Text Sentiment Analysis, A Comparison of Machine Learning Models

Isac Svensson
Dept. of Software Engineering
Blekinge Institution of technology
SE–371 79 Karlskrona, Sweden
isac.m.svensson@gmail.com

Introduction

The amount of data on the internet grows rapidly and according to IDC we can expect the Global Datasphere to grow from 33 to 175 Zettabytes (Zetta = 10^{21}) from 2018 to 2025 [1]. A fraction of all this data is text generated by users of different kinds. Out in the open on the internet, it might be customer reviews, blog posts, news articles, or social media posts. But it might also be private messages, emails, scientific reports, or product requirements in a project. And for most people, it is just that; text. Text with which you might think that the cost of extracting any valuable information from, is too time-consuming and expensive to even consider working with. But with modern processors and the use of text mining, you can easily extract a lot of valuable information from unstructured text. The possibilities and limits to explore the text are set by the data scientist working with it. Some examples of text mining are categorization of text into suitable categories (such as categorizing emails to add a priority), stock market prediction, and sentiment analysis.

Sentiment Analysis

Sentiment analysis (also known as opinion mining or emotion AI) is the use of natural language processing, text analysis, computational linguistics, and biometrics to identify, extract, quantify, and study affective states and subjective information [2]. The technique can be applied to both text, speech, and video. But also data such as what you like on social media combined with where you live and your age [3]. Sentiment Analysis is a sub-field of machine learning (ML), and a sub-field of natural language processing (NLP), and data mining when working with text/speech. See fig. 1.

The most basic use of sentiment analysis is identifying if the content has a positive or negative sentiment such as [4] or [5]. But sentiment analysis is widely used in a range from emotion detection in video, professional writing [6], customer service [7], to marketing [8], and even political influencing [3].

Levels of Sentiment Analysis

Document Level: The task at this level [9] is to determine the sentiment of the full document. Whether it expresses a positive or a negative. For example, given a post from social media, the objective is to determine whether the tone is positive or negative.

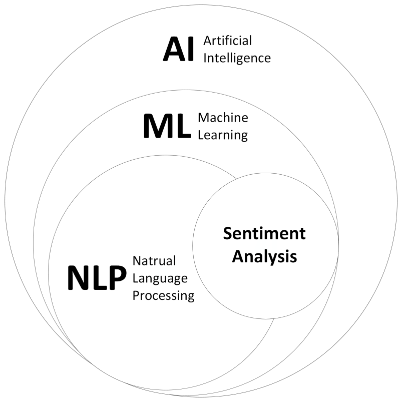


Fig. 1. Relationship between AI, ML, NLP and Sentiment Analysis

Sentence Level: The task at this level goes to the sentences and determines whether each sentence expressed a positive, negative, or neutral opinion [9].

Entity and Aspect Level: Instead of looking at the language construct such as document, paragraph, sentence, aspect level analysis looks at the opinion itself. It bases its idea on that an opinion consists of sentiment (positive or negative) and a target. Considering the sentence "Even though the bass is too low, I think my new headphones are of great quality". It has a clear positive tone, but we cannot say that is completely positive. It was only positive for the quality of the headphones (where the emphasis where), while negative for the lack of bass (not emphasized). The goal of this level is to analyze the sentiment for entities and aspects, such as level of bass and quality.

Approaches to Sentiment Analysis

There are mainly two different approaches to sentiment analysis, lexicon-based, and machine learning. The lexicon-based approach depends on external dictionaries where the words are scored on their sentiment, while the machine learning approach, which this paper emphasizes on the opinions is extracted automatically.

Classifying techniques

There are a wide variety of suitable classifiers for the machine learning approach. In this section, we will look at some of them used for supervised learning.

Probabilistic classifiers: Uses probability to predict an outcome based on observations. E.g. Naïve Bayes

Linear Classifiers: Which separates hyperplanes from a multidimensional space to predict the outcome. E.g. Support Vector Machines (SVM)

Neural Networks: Within this area, in particular convolutional neural networks (CNN), which are often used for processing images, and Recurrent Neural Networks (RNN), which often is used for NLP tasks, have proven to be effective on sentiment analysis.

Random forest: Where multiple decision trees are generated from the training data and then the outcome is generated by either majority voting or averaging.

Organization

This paper will focus on text sentiment analysis and compare performance between Naïve Bayes, Random Forest, and Support Vector Machine paired with different ways of preprocessing the data. In section II related work in the field will be discussed, section III will focus on the methodology used in the experiments. Section IV will focus on the results and analysis of those. And finally, section V will conclude the paper.

RELATED WORK

One of the most important contributions on the topic is Lee and Pang's paper [4]. They were the first ones to apply machine learning techniques to determine the sentiment on text. Before that paper, the work on NLP and text mining was more focused on topical analysis of documents and sentiment analysis was only performed with rule-based and lexicon-based approaches.

Joshi and Tekchandani predicted sentiment on tweets using SVM, Naïve Bayes, and Max Entropy in their paper [5]. They approach the problem by having the features extracted as unigrams, bigrams, and hybrid (both uni- and bigrams) and found that no one solution fits all. For SVM and Naïve Bayes, the hybrid features performed best but, for Max Entropy bigrams worked best (which was the worst-performing method in the other two algorithms).

Yenter and Verma published a paper [10] where they apply CNN and Long Short-Term Memory (LSTM) for predicting sentiment with immense success. They were able to outperform all previous attempts on sentiment analysis on the IMDB review dataset with accuracy over 89%.

Brian Li et al. analyzed customer service calls and, in their work, they used both the spoken, transcribed words and the acoustic expression [7]. This decreases his error rate for both positive and negative sentiment and he learns that for positive sentiment the spoken words are more important than acoustic expression. But for negative sentiment, the acoustic sentiment had a bigger role in predicting the sentiment than the spoken words which are consistent with the hypothesis that: customers are happy to explicitly thank and praise helpful agents, but express displeasure indirectly.

Jiang and Suzuki use Logistic Regression (LR), SVM, and Bidirectional RNN (BiRNN) to detect racism and hate speech [11]. Their work archives an accuracy of 96.56% with the best RNN model.

Poria et al. used a deep convolutional neural network with multiple kernels to perform state-of-the-art emotion recognition and sentiment analysis [12]. They used video segments with utterances of about 5 seconds each and feed the audio to an open-source feature extractor called openSmile, the video was fed through a convolutional recurrent neural network and the transcribed audio was fed through a convolutional neural network. The results reached an accuracy of 96.55% compared to 74.66% by previous work [13] by Pérez-Rosas et al.

METHOD

The research method chosen for this work is experiments. Different approaches and models will be tried and evaluated, and an analysis of the dataset will be performed.

Limitations

Limitations for this project was not to exceed datasets of 4 megabytes. That means that the datasets used will be small and thus have a limited vocabulary. Because of the size limit, neutral sentiment was excluded to have more examples for each class.

Environment

Python is a well-known and established interpreted programming language. It is designed to have high readability and less syntactical constructions than other languages. Python also is established in the data science and machine learning world. It has a broad set of available modules for both data analysis and machine learning. Python is used both for data exploration and the development of the models.

In this experiment, multiple python libraries are used for both data exploration and machine learning development. They are all listed here:

- **NLTK** - NLTK is a platform for building Python programs to work with human language data. It provides a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning and wrappers for industrial-strength NLP libraries. [14]
- **pandas** - pandas is a fast, powerful, flexible, and easy-to-use open-source data analysis and manipulation tool. It is perfect for working with structured data and matrixes.
- **NumPy** - NumPy is the fundamental package for scientific computing in Python. It is used for representing higher-dimensional arrays, matrixes and has a lot of computational functionality built-in.
- **Scikit-learn** – Scikit-learn is one of the most used and powerful libraries for machine learning in Python. It provides tools for machine learning, statistical modeling

and has a lot of implemented machine learning models ready to use.

The algorithms used in this experiment are imported from sklearn (Scikit-learn). They are implemented in a way that we can import them and directly train and test them on the dataset. The following classifiers are used:

- Support Vector Machine
- Naïve Bayes
- Random Forest

Dataset

Amazon is the world's biggest company in online shopping. And with that, there comes a lot of data. Amazon has published a dataset consisting of over 130 million customer reviews [17] that are made public for research purposes. The data is split into .tsv-files by category and range from groceries to software products. For this project, a subset of reviews is used.

To avoid getting a language to biased by one specific category six different categories are used when sampling the subset. These categories are babies, groceries, home improvement, jewels, toys, and video games. These datasets were merged into one big dataset, from which, the subset was extracted from. Five different subsets were created with different distributions of negative and positive reviews.

Since the goal is only to predict binary sentiment (positive or negative) and the size of the dataset is limited to a maximum of 4 megabytes only 1- and 5-star reviews were chosen. Where a 1-star review is of negative sentiment and a 5-star review is of positive sentiment.

The standard distribution of the dataset was 87.1%/12.9% (positive/negative), thus that is the distribution of the subset with the standard distribution. But subsets with the distribution of 80/20, 67/33, 60/40, and 50/50 were created as well to examine what role distribution has in training.

To further examine how well the models are performing, a small dataset of 2200 tweets from Kaggle [18] is used as well.

Feature Extraction

The feature extractors used in this experiment are CountVectorizer, TfidfVectorizer, and HashingVectorizer. All from the sklearn.feature_extration package.

CountVectorizer: Implements both tokenization and occurrence counting. It produces a sparse representation of the counted tokens. It can either represent each token with a binary value or frequency.

TfidfVectorizer: is equivalent with CountVectorization with a TF-IDF-transformation performed. TF stands for term frequency and TF-IDF stands for term-frequency times inverse document-frequency. This is a common term weighting scheme in information retrieval, that is of good use in document classification. The goal of using tf-idf instead of the CountVectorizer is to scale down the impact of tokens that occur very frequently in a given corpus and that are hence empirically less informative than features that occur in a small fraction of the training corpus. [15]

HashingVectorizer: is much like CountVectorizer in the way that it also produces a sparse frequency matrix. But instead of letting every word in the given corpus be represented by a feature, a hashing trick is performed to find an integer corresponding with each token.

In the vectorizers, some different additional settings were entered, such as using both unigrams by themselves and together with bigrams, removing stop words defined by sklearn, number of decision trees in the random forest, and using the idf-smoother in the TF-IDF.

In the data exploration, it was found, that especially question marks had a higher frequency in negative reviews than in positive. Therefore, that was added as an extra feature, to examine if it made any difference.

Data Exploration: A simple data exploration will be performed, where the complete corpus will be examined and information about the dataset will be gathered. Some information that will be examined is word frequency in the different sentiments, the number of question marks and exclamation points is used and, the average length of the reviews. This is to see if any valuable insights can be found.

a) Metrics of Evaluation: To evaluate the performance of the models the metrics Accuracy(1) and F1-Score(4) are used. Accuracy is the obvious choice, it simply calculates the rate of correctly predicted instances where the result is 0 if everything is incorrectly predicted and 1 if all instances are correctly predicted.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

F1-score is a harmonic mean of precision(2) and recall(3) [16]. Precision gives us the proportion of positive predictions that are correct, and recall gives us the proportion of positives that are correctly predicted. Thus, F1-score is the mean of correct predictions in the target class and prediction class. F1-score is especially useful in settings where there is an uneven class distribution.

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$F1\text{-}Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4}$$

Implementation

Below the implementation of the machine learning model is visualized. The implementation and fitted 2880 times with different hyper-parameters. Each model is then evaluated against three different test sets; a test set with standard distribution, a test set with the same distribution as the training data and the tweets. Thus, 8640 iterations of predictions are made.

RESULT & ANALYSIS

Data exploration

When analyzing the data it was clear that some words were more frequent in each class than others. In the sub-dataset with

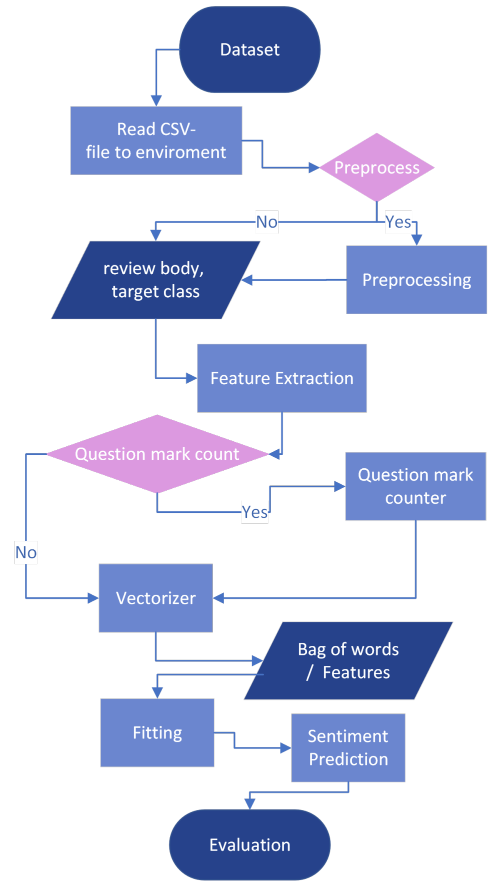


Fig. 2. Implementation of Machine Learning Model.

the distribution of 50% for each class, the top 10 words from reviews are displayed in the left part of table 1.

Pos. review	Neg. review	Pos. tweet	Neg. tweet
great	would	love	sad
love	money	great	miss
loves	disappointed	day	sorry
easy	one	happy	bad
perfect	product	thanks	hate
well	return	great	feel
little	even	nice	im
best	back	mother	work
nice	broke	http	sucks
good	cheap	hope	sick

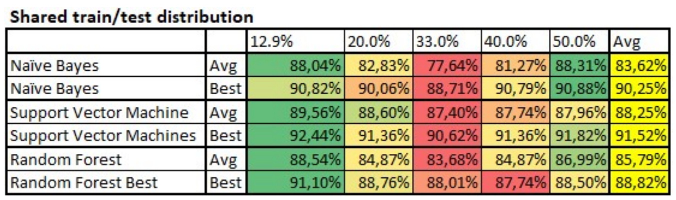
TABLE I
WORDS WITH THE LARGEST DIFFERENCE OF USAGE IN REVIEWS AND TWEETS

Looking at the positive sentiment words they are very similar to the top words of the Twitter dataset (right part of table 1), but the negative sentiment words have a small relationship. This will be enough cause of not being able to reach high accuracy for the tweet dataset.

Looking at the average length of reviews and usage of question marks and exclamation point suggests that there is no significant difference between classes, except for question marks, where the usage is thrice as high for negative sentiment as for positive. This makes sense, because if you are unsatisfied with a product, you will have questions.

Results

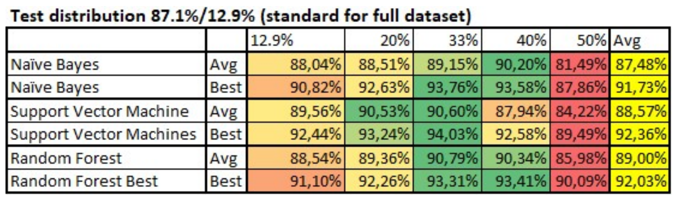
Based on the results of the experiments, a table with accuracy per dataset and algorithm used is shown in Fig. 3, Fig. 4, and Fig. 5. Fig. 3 shows the accuracy when performing tests on a dataset with the same sentiment distribution as the model is trained with. Fig. 4 shows the accuracy when performing tests on a dataset with the same sentiment distribution as the full original dataset with models trained with different distributions. Fig. 5 shows the accuracy when performing tests on the dataset with tweets with models trained with different distributions.



Shared train/test distribution

		12.9%	20.0%	33.0%	40.0%	50.0%	Avg
Naïve Bayes	Avg	88,04%	82,83%	77,64%	81,27%	88,31%	83,62%
Naïve Bayes	Best	90,82%	90,06%	88,71%	90,79%	90,88%	90,25%
Support Vector Machine	Avg	89,56%	88,60%	87,40%	87,74%	87,96%	88,25%
Support Vector Machines	Best	92,44%	91,36%	90,62%	91,36%	91,82%	91,52%
Random Forest	Avg	88,54%	84,87%	83,68%	84,87%	86,99%	85,79%
Random Forest Best	Best	91,10%	88,76%	88,01%	87,74%	88,50%	88,82%

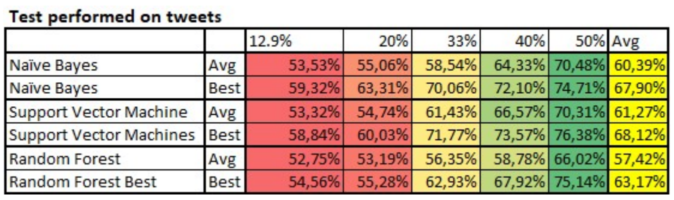
Fig. 3. Accuracy when sharing sentiment distribution on train and test set



Test distribution 87.1%/12.9% (standard for full dataset)

		12.9%	20%	33%	40%	50%	Avg
Naïve Bayes	Avg	88,04%	88,51%	89,15%	90,20%	81,49%	87,48%
Naïve Bayes	Best	90,82%	92,63%	93,76%	93,58%	87,86%	91,73%
Support Vector Machine	Avg	89,56%	90,53%	90,60%	87,94%	84,22%	88,57%
Support Vector Machines	Best	92,44%	93,24%	94,03%	92,58%	89,49%	92,36%
Random Forest	Avg	88,54%	89,36%	90,79%	90,34%	85,98%	89,00%
Random Forest Best	Best	91,10%	92,26%	93,31%	93,41%	90,09%	92,03%

Fig. 4. Accuracy when testing on standard distribution

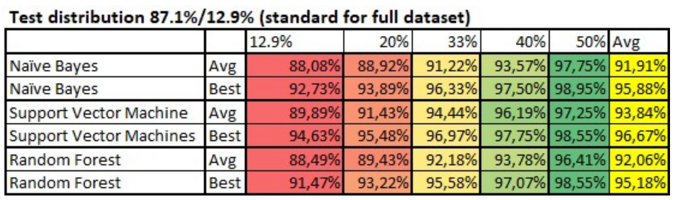


Test performed on tweets

		12.9%	20%	33%	40%	50%	Avg
Naïve Bayes	Avg	53,53%	55,06%	58,54%	64,33%	70,48%	60,39%
Naïve Bayes	Best	59,32%	63,31%	70,06%	72,10%	74,71%	67,90%
Support Vector Machine	Avg	53,32%	54,74%	61,43%	66,57%	70,31%	61,27%
Support Vector Machines	Best	58,84%	60,03%	71,77%	73,57%	76,38%	68,12%
Random Forest	Avg	52,75%	53,19%	56,35%	58,78%	66,02%	57,42%
Random Forest Best	Best	54,56%	55,28%	62,93%	67,92%	75,14%	63,17%

Fig. 5. Accuracy when using tweets as test set

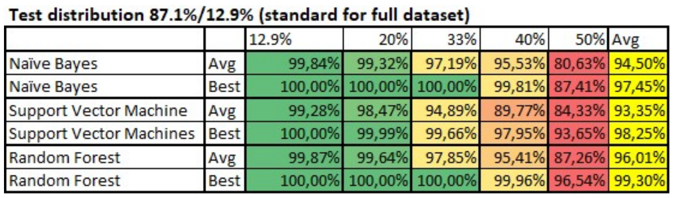
Since the best results are found when using the standard distribution dataset for testing, some more results from those experiments follow here. Fig. 6 displays precision, Fig. 7 recall, and Fig. 8 F-1 score.



Test distribution 87.1%/12.9% (standard for full dataset)

		12.9%	20%	33%	40%	50%	Avg
Naïve Bayes	Avg	88,08%	88,92%	91,22%	93,57%	97,75%	91,91%
Naïve Bayes	Best	92,73%	93,89%	96,33%	97,50%	98,95%	95,88%
Support Vector Machine	Avg	89,89%	91,43%	94,44%	96,19%	97,25%	93,84%
Support Vector Machines	Best	94,63%	95,48%	96,97%	97,75%	98,55%	96,67%
Random Forest	Avg	88,49%	89,43%	92,18%	93,78%	96,41%	92,06%
Random Forest	Best	91,47%	93,22%	95,58%	97,07%	98,55%	95,18%

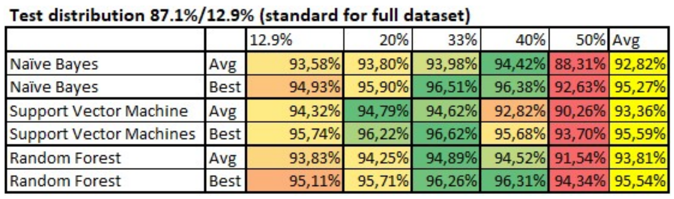
Fig. 6. Precision when testing on standard distribution



Test distribution 87.1%/12.9% (standard for full dataset)

		12.9%	20%	33%	40%	50%	Avg
Naïve Bayes	Avg	99,84%	99,32%	97,19%	95,53%	80,63%	94,50%
Naïve Bayes	Best	100,00%	100,00%	100,00%	99,81%	87,41%	97,45%
Support Vector Machine	Avg	99,28%	98,47%	94,89%	89,77%	84,33%	93,35%
Support Vector Machines	Best	100,00%	99,99%	99,66%	97,95%	93,65%	98,25%
Random Forest	Avg	99,87%	99,64%	97,85%	95,41%	87,26%	96,01%
Random Forest	Best	100,00%	100,00%	100,00%	99,96%	96,54%	99,30%

Fig. 7. Recall when testing on standard distribution

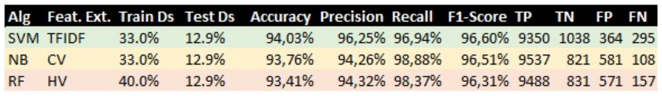


Test distribution 87.1%/12.9% (standard for full dataset)

		12.9%	20%	33%	40%	50%	Avg
Naïve Bayes	Avg	93,58%	93,80%	93,98%	94,42%	88,31%	92,82%
Naïve Bayes	Best	94,93%	95,90%	96,51%	96,38%	92,63%	95,27%
Support Vector Machine	Avg	94,32%	94,79%	94,62%	92,82%	90,26%	93,36%
Support Vector Machines	Best	95,74%	96,22%	96,62%	95,68%	93,70%	95,59%
Random Forest	Avg	93,83%	94,25%	94,89%	94,52%	91,54%	93,81%
Random Forest	Best	95,11%	95,71%	96,26%	96,31%	94,34%	95,54%

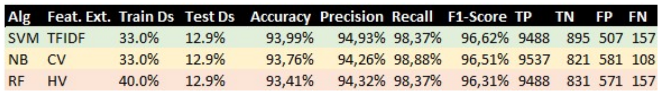
Fig. 8. F1-score when testing on standard distribution

The top-scoring models using Random Forest and Naïve Bayes scored highest on both F-1 score and accuracy within their group. But the best scoring model of SVM was not the same for F1-score and accuracy. The SVM-model in fig. 9 has higher accuracy, but the SVM-model in fig. 10 has only six less correct predictions, but a better-weighted distribution of the false predictions.



Alg	Feat. Ext.	Train Ds	Test Ds	Accuracy	Precision	Recall	F1-Score	TP	TN	FP	FN
SVM	TFIDF	33.0%	12.9%	94,03%	96,25%	96,94%	96,60%	9350	1038	364	295
NB	CV	33.0%	12.9%	93,76%	94,26%	98,88%	96,51%	9537	821	581	108
RF	HV	40.0%	12.9%	93,41%	94,32%	98,37%	96,31%	9488	831	571	157

Fig. 9. Models with highest accuracy



Alg	Feat. Ext.	Train Ds	Test Ds	Accuracy	Precision	Recall	F1-Score	TP	TN	FP	FN
SVM	TFIDF	33.0%	12.9%	93,99%	94,93%	98,37%	96,62%	9488	895	507	157
NB	CV	33.0%	12.9%	93,76%	94,26%	98,88%	96,51%	9537	821	581	108
RF	HV	40.0%	12.9%	93,41%	94,32%	98,37%	96,31%	9488	831	571	157

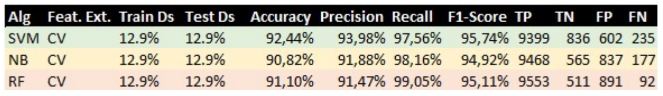
Fig. 10. Models with highest F1-score

Fig. 11 and fig. 12 show the best scoring models that use the same sentiment distribution for both training and evaluation.

Analysis

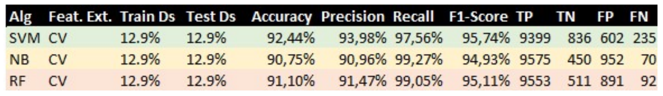
After analyzing the data of the 8640 iterations some valuable information was found. Overall, Support vector machines were the best performing algorithm in all experiments. Another finding is that, although preprocessing and cleaning of the data did not matter that much as expected, it still made a significant difference in accuracy, as we can see in fig. 13. Out of the parameters that were experimented with in this work preprocessing, choice of algorithm and vectorizer, and choice of distribution of sentiment has the most effect on accuracy.

There is no obvious set of settings and parameters to use. In this experiment, Naïve Bayes has performed best together with CountVectorizer and Support Vector Machines together with TF-IDF. Random Forest works better with CountVectorizer on average but did its best iteration with the HashingVectorizer.



Alg	Feat. Ext.	Train Ds	Test Ds	Accuracy	Precision	Recall	F1-Score	TP	TN	FP	FN
SVM	CV	12.9%	12.9%	92,44%	93,98%	97,56%	95,74%	9399	836	602	235
NB	CV	12.9%	12.9%	90,82%	91,88%	98,16%	94,92%	9468	565	837	177
RF	CV	12.9%	12.9%	91,10%	91,47%	99,05%	95,11%	9553	511	891	92

Fig. 11. Models with highest accuracy with the same distribution for train and test



Alg	Feat. Ext.	Train Ds	Test Ds	Accuracy	Precision	Recall	F1-Score	TP	TN	FP	FN
SVM	CV	12.9%	12.9%	92,44%	93,98%	97,56%	95,74%	9399	836	602	235
NB	CV	12.9%	12.9%	90,75%	90,96%	99,27%	94,93%	9575	450	952	70
RF	CV	12.9%	12.9%	91,10%	91,47%	99,05%	95,11%	9553	511	891	92

Fig. 12. Models with highest F1-score with the same distribution for train and test

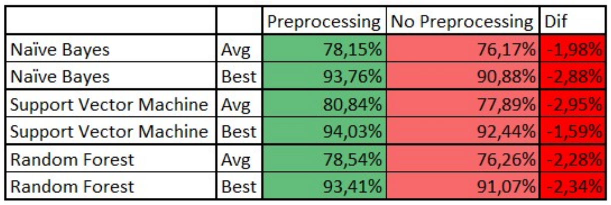
Other parameters such as question mark count as a feature, removing stop words, using bigrams, or using binary or frequency representation all had minor roles (average difference ±1% for accuracy per setting) that together made a bigger impact on the scoring.

F1-Scoring was often in correlation to accuracy as mentioned in the result section. On average the rank is changed ±3 places. Looking at the top 25 placements on accuracy ranking, the same models are in the top 25 ranked when looking at F1-score.

When looking at the results from tests performed with the tweets (fig. 5), we can see that even here is the Support Vector Machine the best performing algorithm. The best model reaches an accuracy of 76.36%, which is over expectations, but not good enough to be useful. Looking closer at the results confirms the hypothesis that the models would have an easier time predicting positive sentiment. But when looking at the top-performing models, results are uniformly distributed. The model with the highest accuracy produced the following result: TP: 803, TN: 804, FP: 197, FN: 300.

Conclusion

Being able to analyze your data as a company is important and with the immense amounts of data accumulating each day, this becomes harder to do by hand. Therefore it is important to find methods of finding valuable information automatically, and one way of this is text mining. This paper has experimented with different ways of preprocessing data, extracting features, and different algorithms for predicting sentiment. It has also examined if the models produced could be useful with text from different environments, in this case from Twitter, which it could not.



		Preprocessing	No Preprocessing	Dif
Naïve Bayes	Avg	78,15%	76,17%	-1,98%
Naïve Bayes	Best	93,76%	90,88%	-2,88%
Support Vector Machine	Avg	80,84%	77,89%	-2,95%
Support Vector Machine	Best	94,03%	92,44%	-1,59%
Random Forest	Avg	78,54%	76,26%	-2,28%
Random Forest	Best	93,41%	91,07%	-2,34%

Fig. 13. Accuracy difference for cleaning and preprocessing the reviews.

References

[1] IDC, 2018. The Digitization of the World; From Edge to Core. [online] p.3. Available at: https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf [Accessed 28 December 2021].

[2] En.wikipedia.org. 2022. Sentiment analysis - Wikipedia. [online] Available at: https://en.wikipedia.org/wiki/Sentiment_analysis [Accessed 28 December 2021].

[3] Chan, R., 2022. The Cambridge Analytica whistleblower explains how the firm used Facebook data to sway elections. Business Insider, [online] Available at: https://www.businessinsider.com/cambridge-analytica-whistleblower-christopher-wylie-facebook-data-2019-10?r=US&IR=T [Accessed 28 December 2021].

[4] B. Pang and L. Lee, ”Thumbs up? Sentiment Classification using Machine Learning Techniques”, 2002.

[5] R. Joshi and R. Tekchandani, ”Comparative analysis of Twitter data using supervised classifiers”, 2016.

[6] ”Meet Grammarly’s Tone Detector, the Ultimate Tone Analyzer”, Grammarly, 2020. [Online]. Available: https://www.grammarly.com/blog/tone-detector/. [Accessed: 17- Dec- 2021].

[7] B. Li, D. Dimitriandis and A. Stolcke, ”Acoustic and Lexical Sentiment Analysis for Customer Service Calls”, 2019.

[8] M. Rambocas and J. Gama, ”Marketing Research: The Role Of Sentiment Analysis”, 2013.

[9] B. Liu, Sentiment analysis and opinion mining. San Rafael, Calif.: Morgan & Claypool, 2012, pp. 10-12.

[10] A. Yenter and A. Verma, ”Deep CNN-LSTM with combined kernels from multiple branches for IMDb review sentiment analysis”, 2017.

[11] L. Jiang and Y. Suzuki, ”Detecting hate speech from tweets for sentiment analysis”, 2019.

[12] S. Poria, I. Chaturvedi, E. Cambria and A. Hussain, ”Convolutional MKL Based Multimodal Emotion Recognition and Sentiment Analysis”, 2016.

[13] V. Pérez-Rosa, R. Mihalcea and L. Morency, ”Utterance-Level Multimodal Sentiment Analysis”, 2022.

[14] ”NLTK :: Natural Language Toolkit”, Nltk.org, 2022. [Online]. Available: https://www.nltk.org/. [Accessed: 26- Dec- 2021].

[15] ”sklearn.feature_extraction.text.TfidfTransformer”, scikit-learn. [Online]. Available: https://scikit-learn.org/stable/modules/generated/

[16] P. Flach, Machine learning. Cambridge: Cambridge University Press, 2017, p. 300.

[17] Aws. 2018. Amazon Reviews Index. [online] Available at: https://s3.amazonaws.com/amazon-reviews-pds/tsv/index.txt [Accessed 15 December 2021].

[18] Kaggle. 2020. Sentiment Analysis Dataset. [online] Available at: https://www.kaggle.com/abhi8923shriv/sentiment-analysis-dataset [Accessed 15 December 2021].