CARLETON UNIVERSITY

SCHOOL OF
MATHEMATICS AND STATISTICS

HONOURS PROJECT

TITLE: Multinomial Naïve Bayes Classifiers for Supervised Text Classification: Applications to Sentiment Analysis

AUTHOR: Benjamin Simms

SUPERVISOR: Dr. Shirley Mills

DATE:

CARLETON UNIVERSITY

Abstract

Faculty of Science
Department of Mathematics and Statistics

Bachelor of Mathematics

Multinomial Bayes Classifier for Supervised Text Classification Applications to Sentiment Analysis

by Benjamin Simms

Text classification is the task of assigning predetermined labels to a set of documents, or collections of words. The Multinomial Naive Bayes (NB) Classifier is a simple but powerful learning method rooted in Bayesian theory. In this paper we define the text classification problem, define the Multinomial NB algorithm then apply it two classification tasks: sorting documents by topic, and sorting based on positive or negative opinion on a topic, also known as sentiment analysis.

Acknowledgements

First and foremost, I would like to show my deepest gratitude to my supervisor, Dr. Shirley Mills, an amazing, patient and kind professor who was willing to communicate with me while I was in the middle of nowhere. Without her, this would not have been possible. I am also greatly indebted to all my professors who have taught me in the past four years. I am in debt to Dr Maria Spielauer, Dr Nicolas Berube and Dr Matthew Till from the ESDC. the former for hiring me into the federal government and the latter providing me with the tools to succeed.

Contents

Chapter 1

Introduction

This paper addresses text mining which is the process of applying algorithms to obtain meaningful information from unstructured text. A common application of text mining is to summarize, browse, organize, and classify a large set of documents quickly and efficiently, as outlined by Zhong et al. [8].

One of the simplest yet most powerful text classifiers is the Multinomial Naive Bayes. Our objective is to define, train and test these algorithms on document sorting and sentiment analysis tasks. Multinomial Naive Bayes was chosen because it is both powerful and rooted in simple concepts. For categorical data, we require large feature sets and high-dimensional calculations. In natural language processing, the feature set is a large or complete subset of the vocabulary of words. The Multinomial Naive Bayes is a robust classifier that can process documents without high-order task specific architectures.

1.1 What is Text Classification

With the growing abundance of text available, there is an ever present need to automatically sort and categorise for various tasks. For the purposes of this paper we will discuss the process of assigning a label to each document from a pre-determined set of labels: often referred to as *supervised learning*.

In text classification, we are given a description $d \in \mathcal{X}$ of a document, where $\mathcal{X}$ is the "document space". Descriptions are the representations of the words or terms contained within the document, denoted t_k for $1 \leqslant t_k \leqslant n_d$, where n_d is the length of the document.
In supervised learning we are also given a fixed set of classes $C = \{c_1, c_2, ..., c_n\}$. Classes are

also called ”categories” or ”labels”. Typically, the document space $\mathcal{X}$ is some type of high-dimensional space, and the classes are human defined and specific for each project architecture. [4]

1.2 Text Classifiers

In classification problems, we are given at training set $\mathcal{D}_{train}$ of labelled documents: $\langle d, c \rangle$, where $\langle d, c \rangle \in \mathcal{X} \otimes C$. Using a learning method or learning algorithm, we then wish to learn a classifier or classification function γ that maps documents to classes. That is:

$$\gamma : \mathcal{X} \rightarrow C$$

The supervised learning method denoted Γ, where $\Gamma(\mathcal{D}_{train}) = \gamma$. The learning method Γ takes the training set $\mathcal{D}_{train}$ as input and returns the learned classification function γ.

Once γ is learned it can be applied to the test set $\mathcal{D}_{test}$: which is a set of unlabelled examples: assumed to be similar or identically distributed to the training set $\mathcal{D}_{test}$. Classifier γ achieves optimality when the classifier minimises error on the test set.

Chapter 2

Feature Selection for Text Classification

For the purposes of model construction, a subset or transformation of the data is often necessary for meaningful interpretation. The problem with operating on text is that each document is categorical of variable length.
Feature Selection is the process of choosing which variables in a given dataset are meaningful for the given task. According to [2], the purpose of feature extraction methods are to:

1. Obtain the most relevant information from the original data and represent the information in a lower dimensional space.
2. Constructing combinations of the variables to get reduced number of features while still describing the data with sufficient accuracy.

For text classification tasks, the feature set is a subset of the vocabulary space denoted $\mathcal{V}$: comprised of terms that have power in predicting classes. The combinations of variables that describe the documents are created by vectorization.

2.1 Text Cleaning and Pre-processing

For NLP (Natural Language Processing) tasks, data is naturally very noisy and contains redundant or useless features. There are several techniques to prepare natural language data for training, the objective being to reduce dimensionality and noise while preserving predictive power.

2.1.1 Removing Noise

Before a set of documents are fed into a classifier, they are first cleaned of unnecessary numbers, punctuation, and extra white spaces. Misspellings are corrected by minimising Damerau-Levenshtein bigram distance between each word and its typographically nearest defined word in a pre-defined dictionary [5]. This is equivalent to running each document through a modern spellchecker.
Next terms are stemmed if possible: which is the process of combining variations of a word into one for stronger predictive power, sacrificing individual context. Taking for example the word ”clean” and its superlative ”cleanest”. Stemming would combine the two into simply ”clean”. This reduces the space spanned by the feature set by one dimension.

2.1.2 Stop Words

Training a text classifier requires a vocabulary space to be called and referenced to need be. One of the key assumptions of a Multinomial Naive Bayes is that every term holds predicting power in choosing a class. The inherent problem lies in natural language, where not every term actually holds predicting power.
We are left with *stop words*, terms which provide no predicting power and are omitted by the vocabulary. Stop words are assumed to be identically distributed among the entire corpus. Theorem Stop words: Given Vocabulary space $\mathcal{V}$, and set of pre-determined classes $\mathcal{C}$ = $\{c_1,...,c_J\}$. Let $\phi_{t,c}$ be the distribution of a token t over a particular class $c \in \mathcal{C}$. If the predicted distribution of t is identical over every class: that is,

$$\hat{\phi}_{t,c}\ constant\ \forall\ c \in C$$

Term t is labelled a *stop word*, and removed from the vocabulary space for purposes of feature selection.
$\hat{\phi}_{t,c}$ is an estimate of $\phi_{t,c}$ obtained by separating the training data $\mathcal{D}_{train}$ by class and calculating the proportions of words appearing in a document. The set of stop words is denoted $\mathcal{V}^{'}$. Theoretically, $\mathcal{V}^{'} \cup \mathcal{V}$ should comprise every word in $\mathcal{D}_{train}$.

2.2 Extracting Vocabulary

Given a corpus, the algorithm for text extraction given by [4] is as follows:
$Algorithm : ExtractVocabulary$

Initialise $\mathcal{V}$, Require $\mathcal{V}^{'}$
for each d in $\mathcal{D}$
for each *unique* $t \in d$
if t not in $\mathcal{V}^{'} \cup \mathcal{V}$
$t \in \mathcal{V}$

2.3 Bag-of-Words Model

An easy method to extract features from text for use in machine learning algorithms is the Bag-of-Words (BoW) approach. The only true way to represent categorical data is by tokenizing it: assigning a mathematical representation to each word. The most intuitive way to be to one-hot encode each word.

"One-hot encoding" is a vectorization method where the vocabulary set comprising all words V is formed. Its dimension is equal to the size of the vocabulary set. Each word forms the standard basis: where each word is one "of-itself" and zero elsewhere.

Though technically the word is tokenized, it is still useless in a practical sense. Any attempt to quantify context, semantics and meaning are lost in the process.

The bag of words model attempts to give a more useful representation by leveraging the words position and frequency in the document. This attempts to approximate a word's context without defining it. If the document is long it is split conventionally by sentence or by paragraph. For shorter documents such as reviews the document is intact. The document or subset of said document is converted to a vector the size of the vocabulary. The values of the vector are the counts of words that appear in the document.

$Algorithm : ExtractVocabulary$
Given set of documents $\mathcal{D} \in \mathcal{X}$, Vocabulary Space $\mathcal{V}$
Initialise V_{vocab}: a zero matrix $\in M_{\mathcal{D}\times\mathcal{V}}$
for each d in $\mathcal{D}$
    for each t_k in d: $1 \leqslant k \leqslant n_d$
    if $t_k \in \mathcal{V}$
    $[V_{vocab}]_{d,t_k}$ += 1

2.4 TF-IDF Vectorization

Another method of building a vocabulary space is by tf-idf vectorization: a very popular method in building vocabulary. [1] Leverages the phenomena that ranking words in order of decreasing frequency in a large body of text is logarithmically proportional to the word frequency in the text. **??**

Definition: Inverse Document Frequency

Given corpus $\mathcal{D} \in \mathcal{X}$:
for term t_k

$$\begin{aligned} P(t_k, d) &= P(t_k \text{ occurs in } d) \\ &\approx n_{t_k}/\mathcal{N} \\ idf(t_k, d) &= -log(P(t_k, d)) \end{aligned}$$

Where $\mathcal{N} = |\mathcal{D}|$ is the number of documents in the corpus, and $n_{t_k} = |\{d \in \mathcal{D} : t_k \in d\}|$ is number of documents that term t_k appears in. [6]

Definition: Term Frequency

$$tf(t_k, d) = |\{t_k \in d\}|/|\mathcal{V}|$$

where $|t_k \in d|$ is the word count of a given term for a given document.

2.4.1 TF-IDF Vectorizer

$Algorithm : IF - IDFVectorizer$
Given set of documents $\mathcal{D} \in \mathcal{X}$, Vocabulary Space $\mathcal{V}$
Initialise V_{vocab}: a zero matrix $\in M_{\mathcal{D}\times\mathcal{V}}$
for each $d \in \mathcal{D}$
    for each $t_k \in d$
        $[V_{vocab}]_{d,t_k} = tf(t_k, d) * idf(t_k, d)$

Chapter 3

Multinomial Naive Bayes Classifier

Multinomial Naive Bayes classifiers are generative models, rooted in basic probabilistic assumptions.

3.1 Multinomial Naive Bayes

Before introducing the classifier, it is necessary to make assumptions.

Theorem 3.1. *Conditional Independence Assumption*

For each document $d \in \mathcal{X}$ and class $c \in \mathcal{C}$
For term $t_k \in d : 1 \leqslant k \leqslant n_d$ where n_d is the word count of the document. Then each term is conditionally independent given its class.

$$P(t_1, ..., t_{n_d}|c) = \prod_{1 \leqslant k \leqslant n_d} P(t_k|c)$$

In other terms, Word positioning does not affect priors or conditional probability.

Proof. Proof Omitted.
For natural language the Bag-of-Words Assumption is not necessarily true. The context of words change based on the words preceding it as well as after it. However this assumption is necessary for our model. □

Theorem 3.2. *Positional Independence Assumption*

Assume each feature is conditionally independent given the class.
That is, for document $d \in \mathcal{X}$ and class $c \in \mathcal{C}$

$$P(c|d) \propto P(c) \prod_{1 \leqslant k \leqslant n_d} P(t_k|c)$$

Where n_d is the word count of the given document.

Proof. Proof Omitted.

□

$P(t_k|c)$ is the measure of how much evidence t_k contributes to c being the correct class.

3.2 Maximum A-Priori Class

The measure of algorithmic performance is to maximise or minimise some evaluation metric. Classifiers have concrete metrics for evaluating performance: the percentage or proportional error rate.

Theorem 3.3. *Maximum A-Priori Class*
The "best" class for given document d is the class which maximizes maximum a posteriori (MAP), denoted c_{map}. Where,

$$\begin{aligned} c_{map} &= \operatorname{argmax}_{c \in \mathcal{C}} \hat{P}(c|d) \\ &= \operatorname{argmax}_{c \in \mathcal{C}} \hat{P}(c) \prod_{1 \leqslant k \leqslant n_d} \hat{P}(t_k|c) \end{aligned}$$

Where, $\hat{P}(c)$ and $\hat{P}(t_k|c)$ are the estimates for $P(c)$ and $P(t_k|c)$ obtained training over the training set $\mathcal{D}_{train}$.

Proof. From the conditional independence assumption, we have that:

$$P(d|c) = P(t_1, ..., t_{n_d}|c) = \prod_{1 \leqslant k \leqslant n_d} P(t_k|c)$$

By out bag of words assumption, we assume that the position of term $t_k \in$ document d does not affect its predictive power: that is. For t_i and t_j: where $t_i = t_j$ for $1 \leqslant k \leqslant n_d$:

$$P(t_i|c) = P(t_j|c)$$

Which implies that:

$$P(d) = P(t_1, ..., t_{n_d}) = \prod_{1 \leqslant k \leqslant n_d} P(t_k)$$

This is the necessary evidence we need to apply Bayes Theorem.
Recall that for events A and B where $P(A \cap B) = 0$, then:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Then by Bayes Theorem, for $c \in \mathcal{C}$ and $d \in \mathcal{D}$

$$P(c|d) = \frac{P(c)\prod_{1\leqslant k\leqslant n_d} P(t_k|c)}{P(d)} = \frac{P(c)\prod_{1\leqslant k\leqslant n_d} P(t_k|c)}{P(t_1,\ldots,t_{n_d})} \propto P(c)\prod_{1\leqslant k\leqslant n_d} P(t_k|c)$$

By positional Independence, $P(c)$ and $P(t_k|c)$ are independent, so:

$$\hat{P}(c|d) = \hat{P}(c)\prod_{1\leqslant k\leqslant n_d} \hat{P}(t_k|c)$$

$\hat{P}(c|d)$ is maximised when the largest $c \in \mathcal{C}$ is attributed to the given document, that is:

$$\text{argmax}_{c\in\mathcal{C}}\ \hat{P}(c)\prod_{1\leqslant k\leqslant n_d} \hat{P}(t_k|c) = c_{map}$$

□

Due to the inherent sparsity of a given document-term matrix, calculating c_{map} involves repeatedly multiplying small numbers.
Floating Point Underflow is where the result of a calculation is less than floating point precision. The resultant denormal values can compromise the effectiveness of the classifier. Monotonic transformations do not change the positions of maxima. Leveraging this, is computationally more stable to sum logs of probabilities than to multiply them. For this paper we calculate *maxmimum a posteriori* estimate for class c as:

$$c_{map} = \text{argmax}_{c\in\mathcal{C}}(log\hat{P}(c) + \sum_{1\leqslant k\leqslant n_d} log\hat{P}(t_k|c))$$

3.2.1 Approximation of Prior Estimate

Theorem 3.4. *Optimal Prior Estimate for a Multinomial Naive Bayes Classifier.*
Given class $c \in \mathcal{C}$, $\mathcal{D}_{train} \in \mathcal{X}$, the probability that document $d \in \mathcal{D}$, is attributed to class c is modelled by:

$$\hat{P}(c) \approx \frac{\mathcal{N}_c}{\mathcal{N}}$$

where $\mathcal{N}$ is the number of documents in $\mathcal{D}_{train}$, and
$\mathcal{N}_c$ is the number of documents in the training set with class attribute c.

Proof. Given an observation which is document d assigned class c as X_i $1 \leqslant k \leqslant n_d$, then:

$$c_i = P(X_i) = \frac{x_i}{n}$$

Where n is the number of classes.
Then the joint probability is

$$\begin{aligned} L(c) &= \binom{n}{x_1,\ldots,x_n} \prod_{i=1}^{m} c_i^{x_i} \\ &= n! \prod_{i=1}^{m} \frac{c_i^{x_i}}{x_i!} \end{aligned}$$

The log-likelihood is:

$$\begin{aligned} l(\mathbf{c}) = \log L(\mathbf{c}) &= \log \left(n! \prod_{i=1}^{m} \frac{c_i^{x_i}}{x_i!} \right) \\ &= \log n! + \log \prod_{i=1}^{m} \frac{c_i^{x_i}}{x_i!} \\ &= \log n! + \sum_{i=1}^{m} \log \frac{c_i^{x_i}}{x_i!} \\ &= \log n! + \sum_{i=1}^{m} x_i \log c_i - \sum_{i=1}^{m} \log x_i! \end{aligned}$$

We know $\sum_{i=1}^{m} p_i = 1$ so impose Lagrange Multiplier:

$$L(c, \lambda) = l(c) + \lambda \left(1 - \sum_{i=1}^{m} c^i \right)$$

To find the maximum of $L(c, \lambda)$ with respect to c:

$$\begin{aligned} \frac{\partial}{\partial c_i} L(c, \lambda) &= \frac{\partial}{\partial c_i} l(c) + \frac{\partial}{\partial c_i} \lambda \left(1 - \sum_{i=1}^{m} c^i \right) = 0 \\ \frac{\partial}{\partial c_i} \sum_{i=1}^{m} x_i \log c_i - \lambda \frac{\partial}{\partial c_i} \sum_{i=1}^{m} c^i &= 0 \\ \frac{x_i}{p_i} - \lambda &= 0 \\ p_i &= \frac{x_i}{\lambda} \end{aligned}$$

$c_i = \frac{x_i}{\lambda}$ implies $\lambda = n$, so:

$$c_i = \frac{x_i}{n}$$

Since n is the number of documents $\mathcal{N}$, and x_i is the number of documents belonging to class c_i i.e. $\mathcal{N}_c$, then the optimal prior estimate is:

$$\frac{\mathcal{N}_c}{\mathcal{N}}$$

□

3.2.2 Approximation of Posterior Estimate

Given class $c \in C$ and in vocabulary term: $t \in \mathcal{V}$, define $T_{t,c}$ as the number of of terms that occur in the subset of the training set $\mathcal{D}_{train}$ labelled class c.

Theorem 3.5. *Optimal Posterior Estimate*

$$\hat{P}(t|c) \approx \frac{T_{t,c}}{\sum_{t \in \mathcal{V}} T_{t,c}}$$

Proof. Like the prior estimate, since $\hat{P}(t|c)$ is a proportion and the conditional independence assumption holds. We argue that the vocabulary distribution

$$\phi_{t,c}$$

Is Multinomial conditioned on class label, where $T_{t,c}$ is the number of observations of t in $\mathcal{D}$ labelled class: c, and $\sum_{t \in \mathcal{V}} T_{t,c}$ is the total number of observed terms in the set of documents. The proof is the same as the prior estimate. □

3.2.3 Laplace Smoothing

For example given a text classifier γ with a vocabulary $\mathcal{V}$, and two classes: C_1 and C_2. We have a test document $d_1 = \{t_1, ..., t_k\}$ which is correctly mapped to C_1.
Now introduce $d_2 = \{t_1, ..., t_k, t_k + 1\}$ which is identical to d_1 except one word which is OoV (Out of Vocabulary). Since this word's count is technically zero:

$$P(t_k + 1|C_1) = P(t_k + 1|C_2) = 0$$

Which implies,

$$P(t_1, ..., t_k, t_k + 1|C_1) = P(t_1, ..., t_k|C_1) * P(t_k + 1|C_1) = 0$$
$$P(t_1, ..., t_k, t_k + 1|C_2) = P(t_1, ..., t_k|C_2) * P(t_k + 1|C_2) = 0$$

In natural language problems it is eventual that OoV words will be encountered by classifiers. Laplace Smoothing introduces small non-zero probabilities to each word which prevents the posterior probabilities from dropping to zero. Our posterior estimate becomes:

$$\hat{P}(t|c) = \frac{T_{t,c}+1}{(\sum_{t \in \mathcal{V}} T_{t,c})+|\mathcal{V}|}$$

3.3 Multinomial Naive Bayes Classifier Algorithm

Two learning algorithms comprise a Multinomial NB classifier: a training algorithm and a testing algorithm. Before training, the corpus is "split" randomly into two sets of documents which are subsets of the corpus: denoted D_{train} and D_{test}.

$Algorithm : TrainMultinomialNB(\mathcal{D}, \mathcal{C})$
$\mathcal{V} \leftarrow ExtractVocabulary(\mathcal{D})$
$\mathcal{N}_c \leftarrow CountDocs(\mathcal{C})$ for $c \in \mathcal{C}$
for each c $\in \mathcal{C}$
    $\mathcal{N}_c \leftarrow CountDocsInClass(\mathcal{D}, c)$
    $\hat{P}(c) \leftarrow \mathcal{N}_c/\mathcal{N}$
    $text_c \leftarrow ConcatenateTextOfAllDocsInAClass(\mathcal{D}, c)$
    for each $t \in \mathcal{V}$
        $T_{t,c} \leftarrow CountTokensOfTerm(t, text_c)$
        $\hat{P}(t|c) \leftarrow T_{t,c} + 1/\sum_{t \in \prime \mathcal{V}}(t_{t,c} + 1)$
        return $\hat{P}(c)$, $\hat{P}(t|c)$, $\mathcal{V}$

$Algorithm : TestMultinomialNB(\mathcal{C}, \mathcal{V}, \hat{P}(c), \hat{P}(t|c))$
$W \leftarrow ExtractTokensFromDoc(\mathcal{V}, \mathcal{D}_{test})$
for each $c \in \mathcal{C}$
    $score[c] \leftarrow log(\hat{P}(c))$
    for each $t \in W$
        $score[c] += log(\hat{P}(t|c))$
        return $\text{argmax}_{c \in \mathcal{C}}\, score[c]$

3.4 Evaluating Classification Accuracy

Definition 3.6. Confusion Matrix
Given a set of defined classes C = {c_1,...,c_n}, and a classifier
$\gamma : \mathcal{X} \rightarrow \mathcal{C}$.
Define confusion matrix: $\mathcal{C}_{i,j}$,
where,
$i : 1 \leqslant i \leqslant n$, is the class assigned to by γ,
$j : 1 \leqslant j \leqslant n$ is the "correct" class label.

For the purposes of this paper we use sklearn's confusion matrix [1].

Definition 3.7. Recall

For given class c_i, *recall* is the proportion of documents that are classified correctly, that is:

$$Recall_{c_i} = c_{ii} / \sum_j c_{ij}$$

Recall is a class attribute. The Recall Score of classifier γ is the macro average of class recall.

Definition 3.8. Precision

For given class c_i, *precision* is the proportion of documents that actually about the given class:

$$Precision_{c_i} = c_{ii} / \sum_j c_{ji}$$

Precision is a class attribute. The Precision of classifier γ is the macro average of class precision. While recall is a measure of how many relevant items are selected. Precision is a measure of how many selected items are relevant.

Definition 3.9. Accuracy

Commonly accuracy is defined as (1 - Error Rate). Contrary to precision and recall, accuracy is the proportion of $\mathcal{D}_{test}$ that are correctly classified.

That is,

$$Accuracy_\gamma = \sum_i c_{ii} / \sum_j \sum_i c_{ij}$$

Accuracy is an attribute of the classifier γ itself.

Definition 3.10. f^1-Score

f^1-score is the harmonic mean of precision and recall. Perfect classification is achieved when f^1 score is 1.

That is,

$$f^1_{c_i} = 2(\frac{Precision_{c_i} Recall_{c_i}}{Precision_{c_i} + Recall_{c_i}})$$

f^1 is a class attribute, the f^1 score of a classifier γ is the weighted mean of each class' f^1 score [3].

[1] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion$_matrix.html$

Chapter 4

Document Classifier Example

Data

The 20 newsgroups dataset, downloaded from Kaggle[1], contains documents from twenty distinct newsgroups. Each document contains approximatly 100 words of natural language. Th rest of the document is littred with syntax and subject information which will be thrown out by the classifier.
This allows us to select data with meaningful information and messages with sufficient lengths to apply to the Multinomial Naive Bayes. In addition, each document is correctly given a class label by reference to the document id number and the newsgroup it is associated with. This enables us to tune our model to hopefully minimise our error rate.

Methodology

To split the train and test data we used sklearn's train test split function [2]. After parameter tuning, we decide the proportion of documents allocated to the test set to be $\rho = 0.3$.

Text Pre-processing

Pre-processing includes removal of numbers, punctuations, and extra white spaces. Each message also includes a subject line which is A list of common words was taken from NLTK corpus and used as our stop words.
In addition, words that appear in the corpus below a certain frequency are removed from the

[1] https://www.kaggle.com/crawford/20-newsgroups
[2] https://scikit-learn.org/stable/modules/generated/sklearn.model$_s election.train_t est_s plit.html$

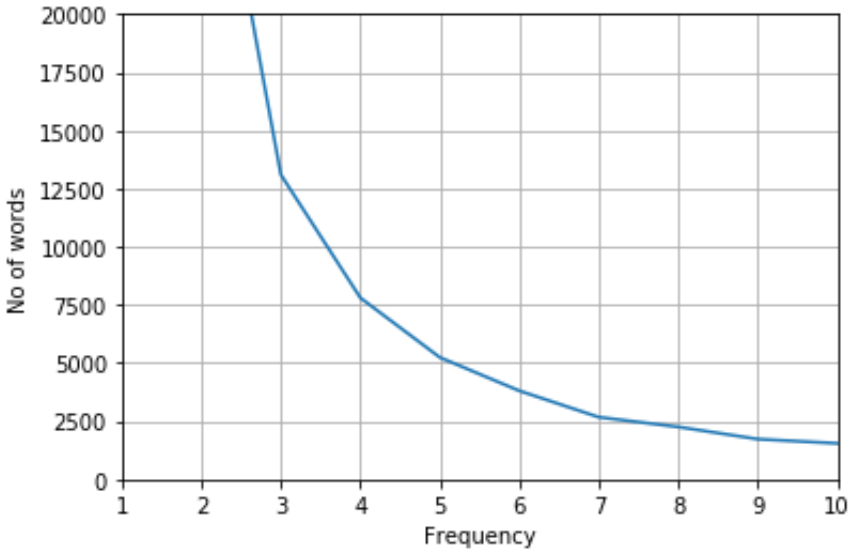

FIGURE 4.1: Term Frequency from each unique term in the training corpus

vocabulary. This removes words with insufficient evidence to support predicting power. The other advantage of this is to remove typos and other Out-of-Vocabulary (OoV) terms. In this example our cut-off frequency is set to 80, giving us a feature set of size 4180.

	Message Sample
Before pre-processing	I was really impressed when I came here. I was seen right away with no appointment. James is seriously thee best. He was friendly, efficient, super helpful, and really went above and beyond. Eyeglass World is lucky to have him. He answered all my questions and made the process of getting my first pair of reading glasses easy. The place is really clean, they have a really big selection, and their prices were the best from every other place I called.
After pre-proccessing	realli impress came seen right away appoint jame serious thee best friend effici super help realli went beyond eyeglass world lucki answer question made process get first pair read glass easi place realli clean realli big select price best everi place call ill definit back frame

TABLE 4.1: Impact of text pre-processing

Analysis

A corpus needs to be converted into a document-term matrix before applying training. The term frequencies for a corpus can be found by summing row-wise. For this dataset we trained two classifiers, one with a standard Bag of Words Approach, the other using TF-IDF. The classifiers were trained on the same train-test split.

	Bag-of-Words Model	TF-IDF Vectorizer
Accuracy	0.86	0.89
Precision	0.86	0.86
Recall	0.86	0.89
F-1 Score	0.86	0.90

TABLE 4.2: Classification Model Performance on the Test Set

From our pre-defined metrics, the TF-IDF classifier did marginally better. More optimal tuning of hyper-parameters would further increase these metrics. [7]

	precision	recall	f1-score	support
alt.atheism	0.70	0.86	0.78	238
comp.graphics	0.83	0.81	0.82	241
comp.os.ms-windows.misc	0.82	0.85	0.84	253
comp.sys.ibm.pc.hardware	0.85	0.84	0.85	251
comp.sys.mac.hardware	0.85	0.93	0.89	226
comp.windows.x	0.88	0.83	0.85	239
misc.forsale	0.80	0.92	0.86	251
rec.autos	0.90	0.91	0.90	274
rec.motorcycles	0.89	0.95	0.92	238
rec.sport.baseball	0.98	0.96	0.97	257
rec.sport.hockey	0.98	0.98	0.98	268
sci.crypt	0.96	0.91	0.93	269
sci.electronics	0.87	0.87	0.87	247
sci.med	0.94	0.90	0.92	238
sci.space	0.90	0.89	0.90	242
soc.religion.christian	0.97	0.99	0.98	260
talk.politics.guns	0.73	0.90	0.81	250
talk.politics.mideast	0.96	0.86	0.91	240
talk.politics.misc	0.78	0.63	0.70	269
talk.religion.misc	0.71	0.49	0.58	249
Test Set				5000

TABLE 4.3: Bag-of-Words Model Performance Metrics

Chapter 5

Sentiment Analysis Example

Sentiment analysis

Sentiment analysis or *Opinion Mining* concerns the attitude of a writer or collection of writers of an event, document, product or idea. Sentiment can be a wide array of emotions or convictions, but for simplicity can be just binary opinion. A *Binary Sentiment Analysis Classifier* is a classifier with two classes: positive or negative. Classifiers are typically trained on labelled reviews. This is done since the sentiment score is labelled by the user, and the subject of the review is present outside the document, typically in its metadata.

Data

The dataset is personally scraped from Indeed, contains about 210,000 business reviews from unique Canadian firms. The dataset is avaliable at [1] Each review is given a score ranging from 1 to 5 stars. For binary classification, reviews with scores 1-3 are labelled negative, while reviews with 4-5 stars are labelled positive.
Each review is approximately 100 words with a high degree of variance. In addition the review contains separate text fields for ”pros” and ”cons” which are appended to the review. This allows us to select data with meaningful information and messages with sufficient lengths to apply to the Multinomial Naive Bayes.

With reviews, the lengths are often varied greatly, as opinion can be conveyed in two words as easily as two-hundred words. Reviews are cut off after 100 terms to make vectorization easier and more accurate. In addition reviews with low word counts are removed. Since predictive

[1] https://github.com/bennysimms/interns$_a$*nalysis*|

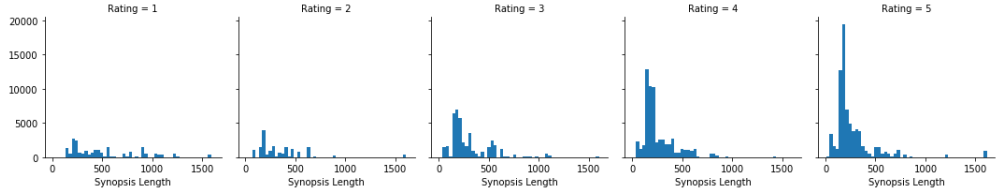

FIGURE 5.1: Synopsis Length for Reviews Separated by Overall Score

power is calculated using the term count of the document, this limits high prediction scores from saturating the model.

Methodology

To split the train and test data use sklearn's train test split function [2]. After parameter tuning, we decide the proportion of documents allocated to the test set to be $\rho = 0.3$. The same training set was then fed through both the BoW (Bag-of-Words) Vectorizer and the TF-IDF Vectorizer, and these were used to train our models.

Analysis

	Bag-of-Words Model	TF-IDF Vectorizer
Accuracy	0.99	0.99
Precision	0.99	0.99
Recall	0.99	0.99
F-1 Score	0.98	0.99

TABLE 5.1: Classification Model Performance on the Test Set

Model performance on the test set was nearly perfect for both models. Compared to the document classification example, classification was easier since:

1. There were only two categories. A random guess would still be accurate half of the time compared to one-twentieth for 20 newsgroups.

2. The intent of the review from a writer's perspective was to convey sentiment. As such, stronger language was used.

To help identify which terms have the highest predictive power, a WordCloud is a useful tool to illustrate that. The python Wordcloud package [3] was used as it can combine n-grams that are often used together in visualisation. For example *stay* doesn't seem to hold much sentiment

FIGURE 5.2: WordCloud for Negative Reviews

FIGURE 5.3: WordCloud for Positive Reviews

until paired with *away*.

Contrary to a negative workplace, what determines a positive workplace from an employee's perspective appears to be less tangible. Work-Life Balance, culture and environment are much more nebulous concepts than time, hours and direct management.

[2] https://scikit-learn.org/stable/modules/generated/sklearn.model$_s election.train_t est_s plit.html$

[3] https://peekaboo-vision.blogspot.com/2012/11/a-wordcloud-in-python.html

Bibliography

[1] Beel, J., B. Gipp, S. Langer, and C. Breitinger (2015, 07). Research-paper recommender systems: A literature survey. *International Journal on Digital Libraries*, 1–34.

[2] Chandrashekar, G. and F. Sahin (2014, January). A survey on feature selection methods. *Comput. Electr. Eng. 40*(1), 16–28.

[3] D.M.W.Powers (2011). Evaluation: From precision, recall and f-measure to roc, informedness, markedness correlation. *Journal of Machine Learning Technologies 2*, 37–63.

[4] Manning, C. D., P. Raghavan, and H. Schutze (2008). *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press.

[5] Mawardi, V., Rudy, and D. Naga (2018, 04). Fast and accurate spelling correction using trie and damerau-levenshtein distance bigram. *Telkomnika (Telecommunication Computing Electronics and Control) 16*, 827–833.

[6] Robertson, S. E. (2004). Understanding inverse document frequency: on theoretical arguments for idf. *Journal of Documentation 60*, 503–520.

[7] Schneider, K.-M. (2005). Techniques for improving the performance of naive bayes for text classification. *CICLing 2005: Computational Linguistics and Intelligent Text Processing*, 682–693.

[8] Zhong, N., Y. Li, and S. Wu (2012, Jan). Effective pattern discovery for text mining. *IEEE Transactions on Knowledge and Data Engineering 24*(1), 30–44.