Predicting Satisfaction in Customer Support Chat

Opinion Mining as a Binary Classification Problem

Henrik Hedlund
Abstract

The study explores binary classification with Support Vector Machines as means to predict a satisfaction score based on customer surveys in the customer support domain. Standard feature selection methods and their impact on results are evaluated and a feature scoring metric Log Odds Ratio is implemented for addressing asymmetrical class distributions. Results show that the feature selection and scoring methods implemented improve performance significantly. Results also show that it is possible to get decent predictive values on test data based on limited amount of training observations. However mixed results are presented in a real-world application example as there is a significant error rate for discriminating the minority class. We also show the negative effects of using common metrics such as accuracy and f-measure for optimizing models when dealing with high-skew data in a classification context.
Contents

Acknowledgements 5

1 Introduction 6

2 Background 7
   2.1 Feature selection 8
   2.2 Parameter tuning 10
   2.3 K-Fold Cross Validation 10
   2.4 Performance measures 10
      2.4.1 Precision and Recall 11
      2.4.2 F-Measure 11
      2.4.3 Accuracy 11

3 Data 12
   3.1 Data 12
      3.1.1 Token statistics 14

4 Method 16
   4.1 Pre-processing 16
      4.1.1 Tokenization 16
      4.1.2 Ngrams 16
      4.1.3 Class categorization 17
   4.2 Feature selection 17
   4.3 Sampling 18
   4.4 Modeling 18
   4.5 Software 18

5 Experiments 21
   5.1 Distributions 21
   5.2 Corpus experiments 22
   5.3 Feature selection 22
   5.4 Model optimizations 24
      5.4.1 Class weights 25

6 Results 27
   6.1 Test set 27
   6.2 Real-world application 28

7 Discussion 30
   7.1 Future work 31
Acknowledgements

I want to thank Beáta Megyesi for giving me valuable feedback throughout the study and helping me structure this thesis. I also want to thank Joakim Jensen for giving me the opportunity to conduct the study at Tele2.
1 Introduction

Researchers in the Natural Language Processing (NLP) field focus on tasks such as machine translation, speech synthesis, computational semantics and text classification among many other areas. Some of the tasks have the goal of making computers understand and produce spoken and written human language, identify linguistic patterns, correct misspelled words and translate documents from any language to another. With the increasing computational power and vast amount of information available today NLP researchers have turned to statistical methods which originated from the field of Artificial Intelligence.

This thesis will examine the possibilities of applying these statistical methods on real-world problems in the corporate domain. The main work of this thesis is only utilizing the tools and studies which have been the done by previous researchers and now we are putting them to the test on new data to see whether they are applicable to other areas.

The study can be categorized as opinion mining with binary classification where the sole focus of the application is to analyse and categorize chat messages and predict their survey scores using a Support Vector Machine (SVM). The purpose of the experiments is to adapt a SVM to predict sentiment in customer support chat and by doing so testing state-of-the-art algorithms and methods on noisy real-word data.

Part of the case study and thesis work is to develop software with the possibility of near real-time predictions by automating the entire work flow. Since the data is from a commercial system the data and the software code will not be distributed with the thesis.
2 Background

This chapter gives an introduction to the concepts and theory used in the thesis. A quick introduction is given to previous research done on text analysis and other related studies using chat data. A general introduction is given to the field of Machine Learning and classification with Support Vector Machines (SVM).

Text analysis is an area where large collections of text is processed for gaining information, identifying patterns or exploring data for new insights (Hearst, 1999). Typical practices of text analysis include text mining and text categorization. Text mining is the process of extracting knowledge or insights from text collections without any linguistic information. When using linguistic knowledge for extracting information the process involves the field of computational linguistics, both practices often apply machine learning to be able to achieve their respective end goals.

Earlier research focused on analyzing chats texts include studies carried out on IRC channels to investigate linguistic phenomenons such as dialects (Siebenhaar, 2008) and identifying and analysing discourse structure among chat participants (Holmen, 2008; Forsyth, 2007). Other studies focused on classification and data mining tasks such as detecting topics in online chat rooms and the identification of conceptual topics in chat messages (Dong, 2006; Rosa and Ellen, 2009).

Since the early 2000’s, sentiment analysis (Turney, 2002) and opinion mining are two areas that has seen a growing interest. Both areas are concerned with the automatic analysis of evaluative text (Pang and Lee, 2008) however sentiment analysis tends to be more semantically motivated and opinion mining more related to information retrieval. Both include the processing of subjective expressions to get information about peoples opinions by discovering associated sentiment information of events and entities (Liu, 2010). Sentiment analysis done on Swedish data is hard to find, though there is a surge in popularity of Swedish studies on sentiment analysis of social media posts (Lysedal, 2010; Karlsson et al., 2001, 2009). No work has been found addressing customer support chat data in Swedish yet.

With the help of machine learning large quantities of textual data can be processed and used for sentiment analysis and opinion mining tasks. Machine Learning is a popular field in computer science and often used in Computational Linguistics and Artificial Intelligence for its inductive and predictive capabilities. To make predictions a statistical learning method is applied that learns by being trained on either labelled or unlabelled data. The data used for learning consists of a set of observations where each observation has a set of features, which can be seen as its measurable properties, and possibly a label that is the output variable used for prediction. When predicting data where the output variable is unknown in the training data, the category of the learning problem is called unsupervised learning. Unsupervised learning can deal with problems such as finding similarities between observations with the end goal of dividing the observations into unknown categories such as clusters. In contrary
when the label for prediction is already known in the training data the category is called supervised learning. Supervised learning includes an area called classification in which the goal is to classify data into already known categories.

Popular machine learning algorithms that can be applied to classification problems include Naive Bayes, Linear Regression, Logistic Regression, Support Vector Machines and K-Nearest Neighbour. For the curious reader that wants more information on the theory behind statistical learning there is a number of textbooks available online.\[1\] In this thesis the SVM will be used since it has proven to work well with text classification tasks [Joachims, 1998; Wang and Manning, 2012]. The software library LibSVM (Chang and Lin, 2011) has eliminated the need for full implementation of the algorithm which saves a lot of time and effort.

The linear classification problem with Support Vector Classification (SVC) can be formalized as: Let D be a dataset of observations \( D = (x_1, y_1),..., (x_n, y_n) \) where \( x \) is an observation and \( y \) is the class in a binary classification problem \( y = \{0 \text{ if negative}; 1 \text{ if positive} \} \). To discriminate the classes, construct a linear separator in a feature space by maximising the margins between features of different classes. An observation can be expressed as a feature vector \( \vec{x}_i = x_1, ..., x_n \). To find the relationship between \( x \) and \( y \), the prediction is calculated from a score for each feature vector of \( x \) and this score is then used for discriminating classes depending on which side of the separator the observations is on.

There are a number of variants and extensions of SVC. SVM is a variation of the original algorithm where instead of constructing a linear separator a kernel function \( K \) is introduced. By training the classifier on feature vector \( \vec{x}_i \) is used by a similarity function \( K(x_i, x_{i'}) \) which can be expressed as a dot product in the feature space (Vapnik, 1999). The dot product is the sum between two feature vectors \( (x_i, x_{i'}) \) divided by length of the feature vector \( |\vec{x}_i| \) and is used to make the support vectors easier to compute by reducing the number of computations required (Gareth James and Tibshirani, 2014), this is especially useful in text classification where the amount of features can be very high.

\[ f(x) = \beta + \sum_{i \in C} \alpha_i K(x_i, x_{i'}) \]  
(1)

\[ K(x_i, x_{i'}) = \exp(-\gamma \sum_{i=1}^{n} (x_{ij} - x_{i'j})^2) \]  
(2)

The full non-linear algorithm using a kernel with a radial function is seen in Equation 2.1 and the kernel function is depicted in Equation 2.2.

An important step in SVM is choosing the correct values for the feature vectors. When using the RBF kernel the classes are discriminated based on the euclidean distance of dot product values so therefore it might be favourable to apply a method where the numerical values of each feature lie in a reasonable range. This process of choosing the correct features and assigning a value is called feature selection.

### 2.1 Feature selection

The purpose of feature selection is to make classification more accurate by reducing the amount of irrelevant features and also to make the classifier training phase less
computationally expensive when dealing with large datasets. Applying the correct feature selection process might improve classifier performance but it depends on the classification problem at hand. Studies carried out on feature selection in text classification [Forman, 2003] recommend using various feature selection techniques for filtering rare words based on threshold values combined with feature scoring methods which all depend on what performance measure the goal is to optimize.

Possible feature scoring approaches are Term Frequency ($tf$) and Document Frequency ($df$). We define the Term Frequency as $tf = (t, d)$ (frequency of the term $t$ given the document $d$) and the Document Frequency as $df = (d, t)$ (number of documents $d$ containing the term $t$). The Inverse Document Frequency ($idf$) is a way of standardizing the $df$ relative to the total collection of documents by taking the logarithmic base value $idf = \log(N/d)$ (All documents $N$ given the documents $d$ that contain term $t$).

The Document Frequency ($df$) can also be used to filter out rare and uninformative terms based on threshold values to reduce the feature space and computations required (Yang and Pedersen, 1997). An extensions of $tf$ and $idf$ is the Term Frequency Inverse Document Frequency ($tfidf$) and it is often used in information retrieval to structure document indexes and rank their similarity using cosine similarity. In a classification context it can be used in the same way by filtering out words from a pre-defined threshold value or be used as a feature score. The $tfidf$ value is calculated by multiplying the $tf$ by the following $idf$ equation.

$$idf = \log\frac{N}{n_t} \quad (3)$$

$$tfidf = tf \cdot idf \quad (4)$$

Stop words lists can be used to exclude words that have none or very low information content, for example if they have high frequency across the entire dataset or belong to a closed part-of-speech class. The hypothesis is that they will not contribute as much to the classifiers ability to discriminate classes and can therefore be excluded.

Log Odds Ratio ($oddsratio$) is another feature scoring metric used successfully in studies on cases with skewed class distribution (Mladenic and Grobelnik, 1999). The metric takes into account how many times the term occur in the positive versus the negative class and also the total size of the classes and can therefore be used for easier discrimination of classes based on their size. Taking the logarithmic base normalizes the value for a better range of values which works better with SVM kernels. The definition used here is calculated as (Forman, 2008):

$$oddsratio = \log\frac{df_t}{neg_t \cdot (N_{pos} - pos_t)} \quad (5)$$

where the log value of the Document Frequency for a given term ($df_t$) is divided by the number of negative documents containing term ($neg_t$) multiplied by the number of positive training observations ($N_{pos}$) minus the number of positive observations where the term occurs ($pos_t$).
2.2 Parameter tuning

The SVM algorithm has a number of parameters that need to be set during the training phase that guides the construction of a linear hyperplane or the non-linear boundary of a kernel function depending on which SVM type is used. The SVM has a cost parameter $C$ which regulates the trade-off between training error and margin maximization (Rychetsky, 2001). A lower $C$ value may tolerate larger distances between the boundary and margin to create flexible (variance) models and a higher value may create over-fit (biased) models. Finding the optimal values is done by applying a function which minimizes generalization error (Bergstra and Bengio, 2011). Depending on the classification problem and the nature of the data provided, a number of considerations will have to be accounted for to optimize the parameters so the output model will be optimal in terms of bias and variance.

The best method for parameter optimization is named Grid Search (JA Nelder, 1965; Powell, 1994) and is used to search the optimal values of $C$ and $\gamma$. Grid search searches a range of parameter options for the two parameters where a matrix of available parameter numbers is constructed, traversed and evaluated with sampling techniques where finally the parameters with the least amount of classification errors will be chosen.

2.3 K-Fold Cross Validation

Cross validation is a statistical sampling method for evaluating performance on datasets by randomly separating the dataset into $K$ folds where the first fold is a validation set and the remaining $K-1$ folds are used for training the data. Each fold is evaluated against performance metrics with the goal of observing classifier performance variance caused by eventual spread among features and classes in the training data. Applying $K = 10$ folds can provide a statistically significant measurement of variance across the dataset. Training the model with cross validation can itself tell more about how the model will generalize and results are more predictable on unknown data.

2.4 Performance measures

There are a number of ways of evaluating performance of machine learning algorithms. The choice of metrics for a specific problem may vary depending on the goal of the evaluation. An introduction to standard metrics for a binary classification problem and how they are used in this study is given below. The measurable outcomes of a classifier can be divided into the amount of true positives $tp$ (in this study the correctly classified positive observations), true negatives $tn$ (in this study correctly classified negative observations), false positives $fp$ (in this study negative observations falsely classified as positive) and false negatives $fn$ (in this study positive observations falsely classified as negative).

Table 2.1 illustrates how we define the true or negative condition in relation to our positive and negative classes. The vertical columns represent the predicted class value as outputted by the classifier while the vertical rows represent the real class value, e.g. if an observation from the positive class is predicted as positive, this will be a true positive result.
positive negative
| positive | tp     | fn    |
| negative | fp     | tn    |

**Table 2.1: Prediction table**

### 2.4.1 Precision and Recall

The standard metrics for evaluation of information retrieval systems is precision and recall which measure the effectiveness of document retrieval systems by calculating the amount of relevant retrieved documents given a query. In the classification context, Precision and Recall has another meaning where Precision is a measure of proportions of positive predictions, hence we refer to it as **Positive Predictive Value (ppv)**. We refer to **Recall** as **True Positive Rate (tpr)** since this is a better fit name in the classification context because it depicts how many of the positive classes were correctly classified. On the contrary, the **True Negative Rate (tnr)** measures how many of the negative observations were correctly classified. The **False Positive Rate (fpr)** is used to measure the number of observations falsely classified as positive out of all negative observations.

\[
ppv = \frac{tp}{tp+fp} \quad tpr = \frac{tp}{tp+fn} \quad tnr = \frac{tn}{fn+tn} \quad fpr = \frac{fp}{fp+tn}
\]

### 2.4.2 F-Measure

F-measure (f1) is the harmonic mean between precision and recall:

\[
f1 = 2 \cdot \frac{ppv \cdot tpr}{ppv + tpr}
\] (6)

### 2.4.3 Accuracy

Accuracy is another common metrics used for evaluating classifier performance. In binary classification the accuracy is defined as the total number of correct classifications among the total output of classifications, including false classifications such as false positives and false negatives.

\[
accuracy = \frac{tp+tn}{tp+fp+tn+fn}
\] (7)
3 Data

This chapter gives an introduction to the data used in this study. A quick introduction to the background of the data and a description of its structure is provided. We choose to see the chats as documents which contains all the written texts of both participants i.e the visitor and the helpdesk and the survey answers acts as labels carrying subjective opinion of the visitor.

3.1 Data

The chat messages come from an online software-as-a-service (SaaS) platform called LiveChat where customers can get in contact with the company through an embedded pop-up window on their website. On the company website the customers are sent to a chat page where they are asked to select their problem category such as invoices or subscriptions before they get in contact with the support agent. The customer gets in touch with the company agent who will answer any questions or help with any problems the customers might have.

The archived chats were retrieved using the service REST API and consists of 9973 archived customer support chats from February 2016. The chats are stored in a JavaScript Object Notation (JSON) format and contains all the messages between the agents and customers during that particular chat session. The LiveChat data also contains customer surveys which are presented to the customers automatically after each chat session. They include several questions and the answers are then used as labels for the classification task.

A sample message is constructed as illustrated in Figure 3.1, each message is an object contained in an array in the parent chat object. Not shown in the sample structure there are also other data such as problem topic, meta data such as geographical location and time queued before getting into the chat. These will not be addressed because they are not in the scope of this thesis but may be included in the case study analysing the performance from a larger perspective.

"date": "Sat, 02/20/16 02:58:11 pm",
"user_type": "agent",
"agent_id": "agent@company.com",
"text": "Hej Maud, jag ska se varför du inte [...]",
"type": "message",
"timestamp": 1455976691

Figure 3.1: Chat message structure (JSON)
Overall, how satisfied are you with the service received in contact with customer support?

Table 3.1: First survey question

1 Very unsatisfied
2 Pretty unsatisfied
3 Neither unsatisfied/satisfied
4 Pretty satisfied
5 Very satisfied

Table 3.2: Survey answer options

Each chat object has 5 survey questions but they do not always contain data since they are optional. Survey question 1, 4 and 5 are related to the customer satisfaction but for this task only the first question (translated in Table 3.1) is used because it describes the overall experience and therefore could be seen as the definitive answer while the other question address other experiences that regards the service agents abilities in a specific way or the customers previous experiences. The answers then also vary in the form of either a scale, yes/no questions or free text submissions. The numerical scale of 1-5 is based on the translated survey answer options from Table 3.2 where 1 is the least satisfied and 5 is the most satisfied.

Figure 3.2: Survey answer distributions

The majority of the survey questions which have numerical scaling has an average of 82% positive, see the distribution for the first question illustrated in Figure 3.2. As a result there is an uneven distribution of positive and negative classes in the dataset. If looking at the problem from a classification perspective this uneven distribution or skew of data is likely to have negative effect on classifier performance since many features may be biased against the majority class. Generally the content of the chat messages is very different to standard text. The data has large numbers of typos, cases of foreign languages and large number of named entities such as names, addresses, invoice numbers and telephone numbers. This means that if a linguistic study was performed then stricter rules of preprocessing filters, spelling corrections and language identification would have to be made for better processing. In some cases of text classification this might be necessary, however no presumption is made whether it is better or worse to include or exclude such occurrences in this case.
There are a number of standardised responses that agents can use to quickly respond to common customer cases. Some of these responses are invoked by the agents and other are triggered automatically by events such as in the beginning of the chat and at the end of the chat. They are not available in all chats, but most of the chats have an entry message and an exit message depending on events such as whether the agent or customer closed the chat. These standard responses will not be filtered out or given much attention for further preprocessing, feature selection conditions may filter them out in later experiments.

3.1.1 Token statistics

Since there is an automated greeting message in most messages the first 1% of words for every class will be sorted out from the frequency list to get better overview of relevant most frequently occurring tokens. Below is a list of the 20 most common words extracted from a re-sampled dataset where the number of positive and negative case are equal. The tokens are translated except for some unclear cases (e) or those hard to translate (ju).

<table>
<thead>
<tr>
<th>negative</th>
<th>amount</th>
<th>positive</th>
<th>amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>help (hjälpa)</td>
<td>353</td>
<td>over (över)</td>
<td>285</td>
</tr>
<tr>
<td>ticket (ärende)</td>
<td>354</td>
<td>service (service)</td>
<td>286</td>
</tr>
<tr>
<td>since (sedan)</td>
<td>358</td>
<td>understand (förstår)</td>
<td>299</td>
</tr>
<tr>
<td>short (kort)</td>
<td>363</td>
<td>take (ta)</td>
<td>300</td>
</tr>
<tr>
<td>someone (någon)</td>
<td>364</td>
<td>out (ut)</td>
<td>305</td>
</tr>
<tr>
<td>moment (ögonblick)</td>
<td>364</td>
<td>left (kvar)</td>
<td>307</td>
</tr>
<tr>
<td>over (över)</td>
<td>367</td>
<td>new (ny)</td>
<td>308</td>
</tr>
<tr>
<td>do (göra)</td>
<td>371</td>
<td>much (mycket)</td>
<td>312</td>
</tr>
<tr>
<td>send (skicka)</td>
<td>371</td>
<td>e</td>
<td>320</td>
</tr>
<tr>
<td>exists (existerar)</td>
<td>373</td>
<td>them (de)</td>
<td>321</td>
</tr>
<tr>
<td>day(dag)</td>
<td>378</td>
<td>here (här)</td>
<td>322</td>
</tr>
<tr>
<td>answer (svara)</td>
<td>381</td>
<td>moment (ögonblick)</td>
<td>322</td>
</tr>
<tr>
<td>stand (stå)</td>
<td>383</td>
<td>kr</td>
<td>323</td>
</tr>
<tr>
<td>our (våran)</td>
<td>384</td>
<td>absolutely (absolut)</td>
<td>326</td>
</tr>
<tr>
<td>at (hos)</td>
<td>387</td>
<td>good (bra)</td>
<td>344</td>
</tr>
<tr>
<td>here (här)</td>
<td>387</td>
<td>had (hade)</td>
<td>359</td>
</tr>
<tr>
<td>ju</td>
<td>390</td>
<td>wish (önska)</td>
<td>361</td>
</tr>
<tr>
<td>kr</td>
<td>393</td>
<td>,</td>
<td>363</td>
</tr>
<tr>
<td>wrong (fel)</td>
<td>403</td>
<td>me (jag)</td>
<td>363</td>
</tr>
<tr>
<td>had (hade)</td>
<td>411</td>
<td>go (gå)</td>
<td>373</td>
</tr>
</tbody>
</table>

Table 3.3: Top 20 tokens per survey answer

A number of interesting observations can be made from Table 3.3, one being that words like wrong is more frequently used in the negative and good is more frequently used in the positive category. This seems like prototypical words that can be used for discriminating classes. Another seems to be words which may be used in polite ways like absolutely and understand are more frequent in the positive class.

By segmenting the classes by visitor type, we can observe differences in the dataset. Looking at Table 3.4 we can see that the tokens vary in numbers for each
visitor type. Based on the number of tokens observed for each visitor type the agent is probably more likely to use fewer named entities such as phone numbers and invoice numbers than the visitor.

<table>
<thead>
<tr>
<th>data</th>
<th>tokens</th>
</tr>
</thead>
<tbody>
<tr>
<td>agent</td>
<td>6722</td>
</tr>
<tr>
<td>visitor</td>
<td>11328</td>
</tr>
<tr>
<td>all</td>
<td>14694</td>
</tr>
</tbody>
</table>

*Table 3.4: Feature per visitor type*

You could argue that the Agents texts probably are better for creating a generalized model because it is the most consecutive with the same features occurring across different observations. We test this hypothesis in later experiments.
4 Method

This chapter explains the methods for predicting opinion from customer support messages using state-of-the-art methods by the adaptation of SVM. In this study, software written in Java is used to automate work flows for easy iteration and application on new datasets. The work flow in the following sections describes the linear process of the software as illustrated in Figure 4.1. A simplified schematic of the software class structure is presented on the last page of this chapter.

![Figure 4.1: Method overview](image)

4.1 Pre-processing

For this task no pre-processing based on linguistic knowledge such as spelling correction or detection of non-Swedish words are done as part of the purpose is to examine performance on processing noisy data instead of gold-standard corpus material.

4.1.1 Tokenization

We develop a tokenizer for the task of separating words and punctuations into processable units (tokens) which produces better representations of words. Otherwise words would include punctuations when processed by the classifier. The basic operation of the tokenizer is to transform characters to lower-case, separate punctuations including commas, exclamation marks and colons from words and then split the messages on white space characters so each word or punctuation is separated into a single unit.

4.1.2 Ngrams

The software also has functionality to separate $n$ consecutive words or symbols into ngrams. By processing each unit as single tokens and then combining $n+1$ tokens into an ngram we create a new token consisting of several tokens. Following previous studies showing that bigrams leads to increased performance we construct bigrams. (Wang and Manning 2012).
4.1.3 Class categorization

We choose to define the problem as binary classification and therefore the numerical survey answers have to be divided into two classes instead of 5 with a categorization rule. The chosen rule for binary categorization is that if the survey answer is lower than or equal to 3 on the scale it is negative. If the answer has a score above 3 on the scale then the opinion is positive. By applying these rules we divide the total dataset into two classes for the binary classification problem, the resulting dataset is presented in the table below.

<table>
<thead>
<tr>
<th>data</th>
<th>total features</th>
<th>negative</th>
<th>positive</th>
</tr>
</thead>
<tbody>
<tr>
<td>total</td>
<td>4393</td>
<td>43852</td>
<td>735</td>
</tr>
</tbody>
</table>

Table 4.1: Dataset features and binary class distribution

By applying these rules the skewed population remains: around 80% positive and 20% negative. Applying another rule such as removing the neutral opinion would have different effects on the distribution but this option is not implemented in this study.

4.2 Feature selection

All the features are seen as a bag-of-words collection which means that the order and syntactical dependencies are not taken into account. Each feature needs to have numerical value defined as its feature value. Feature scores used as values are Term Frequency ($t_f$), Document Frequency ($d_f$), Inverse Document Frequency ($id_f$) and Log Odds Ratio ($oddsratio$). The feature scores are also used for feature selection by filtering based on a pre-defined threshold value. Experiments include methods for finding the optimal threshold value by traversing a range of values and comparing performance against the baseline model. As previous studies have shown good results using $oddsratio$ feature metric (Mladenic and Grobelnik, 1999; Forman, 2008) we examine its performance in comparison to the other feature scoring metrics. We implement $oddsratio$ as defined in previous studies (Forman, 2008) and described earlier in the background chapter.

Experiments are carried out to evaluate performance with different threshold values for $d_f$. Other feature selection techniques includes filtering using a stop words list. Usually these are based on part-of-speech but since no linguistic information is used in this thesis we construct the stop words collection from the top 100 most frequently occurring words across the total dataset. The stop words collection is created manually based on a feature frequency file and used for filtering before the conversion to the sparse data format. The sparse data format is required for the data to be processed by LibSVM.

As shown in Table 4.2, each line represents an observation and begins with the labelled class followed by indexes of the features and their values. In the first row the

| $tf$ | 1:1 2:2 3:4 4:4 9:1 11:1 12:1 13:1 14:3 15:5 16:1 18:8 20:4 22:5 | ...
| $tfidf$ | 1:0:10904167775683422 2:2.0140391705278344 3:0.3465615 | ...

Table 4.2: Sparse format
feature values after the term index are $tf$. In the second example the feature value is $tfidf$.

### 4.3 Sampling

To address the issue of skewed distribution of classes the corpus is divided into an unbalanced dataset and a balanced dataset by artificial re-sampling. By adjusting the number of observations for each class so that the dataset has a new distribution we examine the effects of the skew, and its potential implications on classifier performance.

<table>
<thead>
<tr>
<th>dataset</th>
<th>training</th>
<th>validation</th>
<th>testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>unbalanced</td>
<td>3110</td>
<td>480</td>
<td>803</td>
</tr>
<tr>
<td>balanced</td>
<td>1100</td>
<td>100</td>
<td>270</td>
</tr>
</tbody>
</table>

*Table 4.3: Data subset sizes*

We split the full data set into a training, validation and test set. The training set is allocated approximately 70% of total observations. 10% is allocated for a development set for tuning parameters. The testing data used for evaluating the performance on unseen data and has 20% of the total data set observations.

### 4.4 Modeling

We create models that discriminate the positive and negative classes based on the features available from training data. The baseline model is created without feature selection and scoring methods and only implement simple parameter optimizations to get decent performance. To create a baseline model we choose values for model parameters $C$ and $\gamma$ based on a limited series of cross validation experiments on training data ($C = 3, 2, \gamma = 0.001$). No experiments are conducted to see whether the optional parameters such as degree and nu values affect the performance. After creating the baseline we extract feature set models for experiments which are then further optimized by applying Grid Search to find the best fit values for $C$ and $\gamma$. Grid Search implementation is provided by the wrapper package grid.py included in the software library LibSVM.

Model creation is done by running LibSVM training module on the training data. The command outputs a model with the support vectors used for prediction which can then be evaluated against the validation or test set. Prediction is done by running the prediction module and the output is then processed by the evaluation class which calculates relevant metrics.

### 4.5 Software

The software described in Figure 4.2 is briefly discussed in this section. Each part of the software is only described in terms of abstract functionality without going into details about what data structures, methods or variables are used.

The Parser uses a HTTP GET command to authenticate against the web server REST API and then saves the JSON response in to an appropriate data structure or file.
After the JSON is saved the Dataset data structures are created. The Processor reads the JSON structure of each chat message which has its texts parsed and tokenized by the Text class. The tokenized texts are then saved in the Dataset ngrams structure and each feature is assigned a feature ID in the features data structure. Other relevant data points such as surveys are saved in their own data structures in the Dataset class.

The tokens can then be processed by a feature selection method which dictates what criteria will be used for filtering. The processed text then has their feature values such as Odds Ratio or TF-IDF calculated before they are inserted and converted to a sparse data format which is then saved as files. After the sparse data files have been created the File class reads each sparse file and splits the file according to some pre-defined class distribution to ensure the same distribution throughout the dataset splits used for training, validation and testing.

The Cross Validation class then reads the appropriate subset such as validation or training and depending on what platform is used the software invokes the LibSVM executable or loads the WEKA java library. If the WEKA library is chosen the data must be read from LibSVM format and converted into the WEKA format. After the dataset has been converted a cross validation method splits the dataset randomly into 10 folds and performs evaluation. After Cross Validation is performed a file is created which holds relevant metrics and statistics for each model or per validation iteration.

When a model is created in the Modeling class, the LibSVM svm-train executable is run with relevant program flags to set parameters or weights. The svm-train executable creates a model file which is then used by svm-predict in the Predict class which also takes a file used for prediction and an output file as arguments which then outputs a prediction results file. To evaluate the prediction results the Evaluation class is run to read the predictions file and compares it to the original file. The Evaluation class then outputs relevant metrics and statistics for the model.
5 Experiments

In this chapter we present experiments conducted to obtain the best performing model. A corpus experiment for investigating distributional skew of classes is evaluated in terms of classifier performance. We investigate feature weights by segmenting the corpus to create sub-corpora based on visitor types. Feature sets created with feature selection methods are compared against each other and different feature scoring metrics are evaluated in combination with the feature selection methods. Further model optimizations are then conducted by parameter tuning using Grid Search.

5.1 Distributions

The evaluation of the distributional variations is illustrated in Receive Operator Characteristic (ROC) space with the True Positive Rate ($tpr$) on Y-axis and False Positive Rate ($fpr$) on the X-axis. A perfect classifier scores a zero $fpr$ and 1 on $tpr$. The grey line across the ROC graph represent performance not different from random outcomes. The purpose is to plot distribution of fold performances so that each point represents a fold outcome. By plotting the results of 10-fold CV it is possible to visualize data variance and to see if a single fold is enough when running the final model on the test data. If there is more spread performance across folds then a 10-fold validation will be necessary when evaluating final model since the test data might be unfavourably distributed for generalization conclusions.

![Figure 5.1: 10-Fold CV on different class distributions](image)

(a) Unbalanced (b) Balanced

Figure 5.1: 10-Fold CV on different class distributions
In Graph 5.1 we plot tpr and fpr for each fold from 10-Fold CV experiments to examine data variance and the effects of skewed class distributions for the positive class (blue) and negative class (orange). Performance across folds in terms of dataset variance and tpr and fpr trade-off is measured by the distance observed in ROC space. As visualized in a trade-off between tpr and fpr the classifier performance is negatively affected for the minority class (orange) for the unbalanced set in graph (a). For the balanced set the negative class is performing almost as good as the positive which is an indicator that the classifier is well fit to discriminate classes on equal grounds but it is by no means evidence for improved generalization capabilities for the testing phase.

5.2 Corpus experiments

Based on data presented during examination of the dataset, major differences in features for visitor type segments were observed. To examine what effects each visitor type segment has on classifier performance we experiment on sub-corpora created from segmenting on the visitor type data. Two corpora are created as listed in the table below and compared to the baseline with the same model parameters. The agent corpus only contains chat messages that the customer support agent has written. The visitor corpus only has text from messages written by customers visiting the customer support chat.

<table>
<thead>
<tr>
<th>SVM</th>
<th>Features</th>
<th>TPR</th>
<th>FPR</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>43852</td>
<td>0.8482</td>
<td>0.7812</td>
<td>84.375</td>
</tr>
<tr>
<td>Agent</td>
<td>16222</td>
<td>0.8354</td>
<td>0.8229</td>
<td>83.5416</td>
</tr>
<tr>
<td>Visitor</td>
<td>34698</td>
<td>0.8333</td>
<td>0.8333</td>
<td>83.3333</td>
</tr>
</tbody>
</table>

Table 5.1: Baseline and sub-corpus performance

Slight differences in performance between the baseline and agent corpora are observed as seen in Table 5.1, however the visitor corpus is performing worse even given its larger number of features. The implications of this seem to be that the agent features carry more weight than the visitor features. We hypothesize that the visitor features are less consecutive, meaning that rare terms are more prevalent and that the variance is greater across observations and therefore less prone to making accurate predictions. However, no further visualisations are made to support this but the positive effect of reducing the feature variations of rare term occurrences is supported by previous studies (Yang and Pedersen, 1997; Forman, 2003).

While it is interesting to examine the differences in data, the benefit of only using agent texts seem to be less than the baseline and therefore the conclusions of the experiment seem to be that including all features is affecting the performance positively. Since visitor and agent corpus is performing worse than the baseline they will not be used for further experimentation.

5.3 Feature selection

We experiment with different feature sets which are constructed on different preprocessing criteria and other conditions for filtering tokens such as threshold values from aggregated feature scoring metrics. The purpose of the experiments is to evaluate
performance of various feature sets against the baseline. First we evaluate unigram tokens with various feature selection techniques by filtering out named entities such as telephone number and invoice numbers (named), applying stop word filter (stop), applying TF-IDF (t\textit{f}id\textit{f}) and Log Odds Ratio (oddsratio) feature scoring metrics. Then we apply the same techniques to bigram features.

<table>
<thead>
<tr>
<th>svm</th>
<th>ppr</th>
<th>tpr</th>
<th>fpr</th>
<th>fl</th>
<th>accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>unigram</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>baseline</td>
<td>0,8684</td>
<td>0,8438</td>
<td>0,7813</td>
<td>0,7815</td>
<td>84,3750</td>
</tr>
<tr>
<td>named</td>
<td><strong>0,8699</strong></td>
<td>0,8458</td>
<td>0,7708</td>
<td>0,7860</td>
<td>84,5833</td>
</tr>
<tr>
<td>stopwords</td>
<td>0,8626</td>
<td>0,8354</td>
<td>0,8229</td>
<td>0,7626</td>
<td>83,5417</td>
</tr>
<tr>
<td>tfidf</td>
<td>0,8626</td>
<td>0,8354</td>
<td>0,8229</td>
<td>0,7626</td>
<td>83,5417</td>
</tr>
<tr>
<td>oddsratio</td>
<td>0,6944</td>
<td>0,8333</td>
<td>0,8333</td>
<td>0,7576</td>
<td>83,3333</td>
</tr>
<tr>
<td>bigram</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>baseline</td>
<td>0,7998</td>
<td>0,8375</td>
<td>0,7525</td>
<td>0,7866</td>
<td>83,7500</td>
</tr>
<tr>
<td>named</td>
<td>0,7932</td>
<td>0,8354</td>
<td>0,7529</td>
<td>0,7853</td>
<td>83,5417</td>
</tr>
<tr>
<td>stopwords</td>
<td>0,6944</td>
<td>0,8333</td>
<td>0,8333</td>
<td>0,7576</td>
<td>83,3333</td>
</tr>
<tr>
<td>tfidf</td>
<td>0,8097</td>
<td>0,8396</td>
<td>0,7620</td>
<td>0,7851</td>
<td>83,9583</td>
</tr>
<tr>
<td>oddsratio</td>
<td>0,8286</td>
<td><strong>0,8479</strong></td>
<td><strong>0,7104</strong></td>
<td><strong>0,8041</strong></td>
<td><strong>84,7917</strong></td>
</tr>
</tbody>
</table>

**Table 5.2: Feature performance**

As shown in Table 5.2 the experiments show that adding feature scoring oddsratio does improve classifier performance significantly compared to the baseline for bigram features but not for unigram features. Even with the amount of noisy features and skew in the dataset the oddsratio shows good capabilities for better discrimination of features belonging to skewed classes.

The best performing feature filtering method for unigram features is named where it outperforms oddsratio by large margins for all metrics. Filtering based on named entities results show better than baseline performance by a marginal decrease in fpr for unigram features, however the same positive effect is not seen on bigram features. Using bigram features shows great performance for being able to discriminate more negative observations as observed by the decrease in fpr. The increase in number of features seem to have no negative impact on performance.

Feature scoring with tfidf does not seem to add any value in this case. Instead it decreases overall performance compared to baseline for unigrams except for bigrams where the accuracy increases because of an increase in tpr.

Filtering stop words by this method does not generate improvements and instead the general performance is decreased across the board. Using the stop words method to filter out common words for all observations may be the wrong approach and further experiments on filtering the most rare occurrences instead is presented in the following section by using Document Frequency (\textit{df}) instead. We experiment with aggressive feature selection by applying a range of threshold values for exclusion of rare terms in the datasets during pre-processing. \textit{df} is used to filter out the least frequent terms to preserve the terms with the most information value.

In Table 5.3 we can see a trade-off between tpr and fpr and in this case we value better fpr more than tpr. Even though a large number of features are being filtered out the performance seem to be unaffected, the probable causes for this is the limited size of the validation set and how the ratio is calculated in WEKA by weighting the
<table>
<thead>
<tr>
<th>df</th>
<th>features</th>
<th>tpr</th>
<th>fpr</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>43852</td>
<td>0.8482</td>
<td>0.7812</td>
</tr>
<tr>
<td>1</td>
<td>16277</td>
<td>0.8458</td>
<td>0.7708</td>
</tr>
<tr>
<td>2</td>
<td>10991</td>
<td>0.8458</td>
<td>0.7708</td>
</tr>
<tr>
<td>5</td>
<td>6712</td>
<td>0.8458</td>
<td>0.7708</td>
</tr>
<tr>
<td>10</td>
<td>4679</td>
<td>0.8458</td>
<td>0.7708</td>
</tr>
<tr>
<td>20</td>
<td>3324</td>
<td>0.8458</td>
<td>0.7708</td>
</tr>
<tr>
<td>40</td>
<td>2311</td>
<td>0.8438</td>
<td>0.7813</td>
</tr>
<tr>
<td>80</td>
<td>1593</td>
<td>0.8458</td>
<td>0.7708</td>
</tr>
<tr>
<td>160</td>
<td>1046</td>
<td>0.8438</td>
<td>0.7813</td>
</tr>
<tr>
<td>320</td>
<td>601</td>
<td>0.8396</td>
<td>0.8021</td>
</tr>
<tr>
<td>640</td>
<td>363</td>
<td>0.8396</td>
<td>0.8021</td>
</tr>
<tr>
<td>1280</td>
<td>207</td>
<td>0.8333</td>
<td>0.8333</td>
</tr>
</tbody>
</table>

Table 5.3: $df$ threshold value performance

average by class size. Even though cross validation is performed, the skew causes all classes to be predicted as the positive class and still retain the same performance even when $df$ has a large value (>80). In the extreme cases ($df = 1280$) the outcome is not able to find any of the negative class and thus both $fpr$ and $tpr$ is 83%.

Other interesting observations from the experiment is that by filtering out features which occur in a single document ($df = 1$) reduces the feature space by 58%. The probable cause of decrease in feature numbers is typos and named entities such as telephone numbers and invoice numbers which are very local to each chat. This feature selection method can therefore used for reducing computational requirements when optimizing the models by parameter tuning.

5.4 Model optimizations

After each best performing feature selection and scoring method is chosen we combine them and proceed to optimize the models by parameter tuning via the Grid Search function. We present results from experiments on both unigram and bigram features using the oddsratio feature scoring and also combine it with a $df$ threshold value of 1. As the bigram dataset is very large with over 350 000 features and 4300 observations and Grid Search is computationally expensive the process is taking too long on my personal computer running an Intel i5-4590S CPU with 8GB RAM. To make the process faster we run the optimizations on the validation set only.

Table 5.4 shows the best parameters found by the Grid Search optimization function by doing 10-Fold CV with the goal of maximizing the accuracy metric. Grid Search is maximizing accuracy because it is the only output metric for LibSVM CV function. Generally the impact on results from parameter optimization seem to be less important than using oddsratio feature scoring. The $df$-oddsratio has a $df = 1$ and it does not seem to perform any better than just using oddsratio. Using Grid Search can possibly over-fit the model to a specific subset of data but measures were also taken to cross validate against the full training set for the unigram feature set. For bigram features a 40% subset is extracted (for reducing computational requirements) of the training data and used for confirming parameter values and the same parameters are
also found. The parameters resulting from this experiment will then be used in the test phase.

5.4.1 Class weights

The overall performance of the classifier is still not adequate for generalization because of the high \( fpr \). The classifier has problems in discriminating the minority class and to address this we turn to experiments with class weights. The LibSVM package has built-in functionality for addressing class skew by setting a class prediction penalty. The function takes a parameter value for each class which defines a penalty cost for classifying a certain class by multiplying \( C \) with the provided value. Below we present experiments conducted to search for optimal class weight values, either by reducing the \( C \) of the majority class or increasing \( C \) for minority class.

<table>
<thead>
<tr>
<th>w0</th>
<th>w1</th>
<th>tpr</th>
<th>fpr</th>
<th>accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0.8437</td>
<td>0.7812</td>
<td>84.3750</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0.8520</td>
<td>0.7295</td>
<td>85.2083</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0.8520</td>
<td>0.6095</td>
<td>85.2083</td>
</tr>
<tr>
<td>1</td>
<td>0.6</td>
<td>0.8520</td>
<td>0.7295</td>
<td>85.2083</td>
</tr>
<tr>
<td>1</td>
<td>0.5</td>
<td>0.8520</td>
<td>0.6987</td>
<td><strong>85.6250</strong></td>
</tr>
<tr>
<td>1</td>
<td>0.4</td>
<td>0.8541</td>
<td>0.6591</td>
<td>85.4167</td>
</tr>
<tr>
<td>1</td>
<td>0.3</td>
<td>0.8333</td>
<td>0.5233</td>
<td>83.3333</td>
</tr>
</tbody>
</table>

Table 5.5: Class weight performance

In Table 5.5 we compare weights for the negative class (w0) and weight for the positive class (w1) to the baseline which use the default weight values for each class. We use unigram features across the experiment without feature selection methods. We experiment by setting the positive class weight from a range of 0.1 to 0.6 by 1 increments and for the negative a weight of 10 to 20 by increasing the value of 5 per iteration. The values presented in Table 5.5 are the cut-off values where \( tpr \) and \( fpr \) that performed the best and choosing a value below or above did not produce any improvements. By choosing a decimal value lower than 1 for the positive class we decrease the \( C \) parameters so the margin maximization becomes more flexible. By adjusting the weights to minimize the \( fpr \) we prioritize so the classifier will identify larger number of cases from the minority class. The best performing values from Table 5.5 are weights of 10 on the negative class and 0.5 on the positive class. The
results only work independently and if they are combined the performance is dropped drastically.

As a final experiment the weights are tested in combination with the best performing methods from the feature selection experiments. We apply bigram features with Log Odds Ratio (oddsratio) feature scoring and the best parameter values from tuning experiments in combination with the top class weight values.

<table>
<thead>
<tr>
<th>svm</th>
<th>ppv</th>
<th>tpr</th>
<th>fpr</th>
<th>f1</th>
<th>accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>0.8684</td>
<td>0.8438</td>
<td>0.7813</td>
<td>0.7815</td>
<td>84.3750</td>
</tr>
<tr>
<td>bigram-oddsratio-w10</td>
<td>0.8555</td>
<td>0.8666</td>
<td>0.4466</td>
<td>0.8587</td>
<td>86.6666</td>
</tr>
<tr>
<td>bigram-oddsratio-w05</td>
<td>0.8162</td>
<td>0.8416</td>
<td>0.7516</td>
<td>0.7893</td>
<td>84.1666</td>
</tr>
</tbody>
</table>

Table 5.6: Combined model performance

In Table 5.6 we combine the best results from previous experiments to create the best model. As shown the bigram features with oddsratio feature scaling with parameters $C = 4$ and $\gamma = 0.000488281$ using a positive class weight of 1 and negative class weight of 10 ($w10$) produce major improvements compared to the baseline. The fpr goes down significantly without reducing the tpr which means that the classifier now is able to discriminate the minority class relatively well in comparison to the original model.

We end our experiments with this experiment because we have achieved satisfactory results with current methods. And we also try not to over-generate so we avoid the model over-fitting and in the end not being able to generalize well on unseen data. The model could be further improved by reducing error rate using other subsets of training data. For this we could reproduce experiments with CV on training data. We settle with this model because it is a major improvement from baseline and is on the same level in terms of f1 and accuracy as other classifiers from previous studies (Forman, 2003).
6 Results

6.1 Test set

In this chapter the final evaluation of the models are presented to see how they generalise on previously unseen data. Below we present the results from applying the best performing feature selection and scoring metrics together with the best performing parameter values and class weights found in the experiments. We also compare the results to what we can expect based on previous studies. We choose to present the results with the same structure as in feature experiments and in the process include the different feature sets from the experiments. For all models we use the best parameters found in experiments, for named, stop and tfidf the best parameters for the baseline model is used. All other sets have their respective values found in (5.4).

<table>
<thead>
<tr>
<th></th>
<th>svm</th>
<th>ppv</th>
<th>tpr</th>
<th>fpr</th>
<th>fl</th>
<th>accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>unigram</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>baseline</td>
<td>0.8911</td>
<td>0.9613</td>
<td>0.781</td>
<td>0.9249</td>
<td>86.43</td>
<td></td>
</tr>
<tr>
<td>named</td>
<td>0.8919</td>
<td>0.9456</td>
<td>0.7619</td>
<td>0.9179</td>
<td>85.3051</td>
<td></td>
</tr>
<tr>
<td>stop</td>
<td>0.8851</td>
<td>0.9599</td>
<td>0.8286</td>
<td>0.921</td>
<td>85.6787</td>
<td></td>
</tr>
<tr>
<td>tfidf</td>
<td>0.8897</td>
<td>0.9126</td>
<td>0.7524</td>
<td>0.901</td>
<td>82.5654</td>
<td></td>
</tr>
<tr>
<td>oddsratio</td>
<td>0.903</td>
<td>0.9198</td>
<td>0.6571</td>
<td>0.9113</td>
<td>84.4334</td>
<td></td>
</tr>
<tr>
<td>df-oddsratio</td>
<td>0.9017</td>
<td>0.9728</td>
<td>0.7048</td>
<td>0.9359</td>
<td>88.4184</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bigram</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>baseline</td>
<td>0.8962</td>
<td>0.9398</td>
<td>0.7238</td>
<td>0.9175</td>
<td>85.3051</td>
<td></td>
</tr>
<tr>
<td>named</td>
<td>0.8957</td>
<td>0.9355</td>
<td>0.7238</td>
<td>0.9152</td>
<td>84.9315</td>
<td></td>
</tr>
<tr>
<td>stop</td>
<td>0.8734</td>
<td>0.9885</td>
<td>0.9524</td>
<td>0.9274</td>
<td>86.5504</td>
<td></td>
</tr>
<tr>
<td>tfidf</td>
<td>0.87</td>
<td><strong>0.9971</strong></td>
<td>0.9905</td>
<td>0.9292</td>
<td>86.7995</td>
<td></td>
</tr>
<tr>
<td>oddsratio</td>
<td><strong>0.9219</strong></td>
<td>0.9814</td>
<td><strong>0.5524</strong></td>
<td><strong>0.9507</strong></td>
<td><strong>91.1582</strong></td>
<td></td>
</tr>
<tr>
<td>df-oddsratio</td>
<td>0.9001</td>
<td>0.9556</td>
<td>0.7048</td>
<td>0.927</td>
<td>86.924</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.1: Results on test data

In Table 6.1 we present the results from the test phase. The test prediction is a single fold where we use unweighed metrics in contrast to previous experiments. The best performing classifier (oddsratio) uses bigram features with Log Odds Ratio feature scoring and has assigned the best parameter values $C = 4$ and $\gamma = 0.000488281$ together with a negative class weight of 10. The best performing oddsratio model achieves the best fpr by far and shows that using Log Odds Ratio scoring together with class weights is a successful way to address the problem of skewed distributions. Using tfidf-based feature scoring seem to create bias towards the majority class, probably because it does not take the amount of positive or negative cases containing
the terms into account. Since many terms probably occur more times in the majority class and the feature value does not discriminate for being in the minority class the kernel function will not have large enough distances for accurate discrimination. This is however may be solved by instead of taking the total number of documents that the term occur in, take the number of documents the term occur in given the class which is the basic idea of the \textit{oddsratio} feature metric.

Our best performing model is made by using bigrams with Log Odds Ratio feature scaling and using best parameters ($C = 4$ and $\gamma = 0.000488281$) from Grid Search using a class weight on the minority class of 10 (w10). Significant improvements are made but the results may not be satisfactory for a real-world application since an expected error rate or 55% is expected from the minority class, in the next section we put this to the test.

### 6.2 Real-world application

We evaluate the prediction capabilities of the classifier based on large numbers of unseen and unlabelled data. The purpose of the evaluation is to see how a classifier performs after the testing phase in a real-world situation. From the test phase we can see that there needs to be some further work to address the class skew. However it is interesting to see if the classifier can generalize as well on the unlabelled data as in the test phase. From the total number of observations of 9973 there are 5581 unlabelled observations that will be used for predictive analysis by applying the best classifier model. Since these observations does not have surveys answered, no evaluation is possible neither will the results of the predictions be manually evaluated for performance but instead the predicted class distribution will be compared to the original corpus distribution as an indicator for making qualified guesses about performance.

Returning to the original purpose of this thesis we assess the classifiers performance in predicting a customer satisfaction score (C-SAT) (Andreasen, 1976). The satisfaction score is defined as how many customers that are satisfied from the total population. The original satisfaction score can be mined out of the original surveys by counting the number of surveys that have answered 4 or 5 on the numerical scale of question 1.

<table>
<thead>
<tr>
<th>data</th>
<th>svm</th>
<th>total</th>
<th>negative</th>
<th>positive</th>
<th>C-SAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>labelled</td>
<td>none</td>
<td>4393</td>
<td>735</td>
<td>3658</td>
<td>83,26</td>
</tr>
<tr>
<td>unlabelled</td>
<td>unigram</td>
<td>5581</td>
<td>892</td>
<td>4688</td>
<td>83,99</td>
</tr>
<tr>
<td>unlabelled</td>
<td>bigram</td>
<td>5581</td>
<td>869</td>
<td>4711</td>
<td>84,41</td>
</tr>
<tr>
<td>unlabelled</td>
<td>bigram-oddsratio</td>
<td>5581</td>
<td>580</td>
<td>5000</td>
<td>89,58</td>
</tr>
</tbody>
</table>

Table 6.2: Predictions on unlabelled data

In Table 6.2 we calculate C-SAT based on positive or negative predictions regardless of their truth-condition, this means that we do not account for classifier error which is available in test performance. The results from the final prediction on large amounts of unseen data did have a positive outcome in terms of statistical distribution to the original C-SAT data. But this may be due to the fit of the model to making it biased against the same distribution as the original. If compared in this way the unigram baseline model is performing the best but considering the error rates ($fpr$) in
the test results it makes it unlikely that the performance can be that good. It is none
the less interesting that the classifier seem to identify a reasonable amount of positive
predictions making the final C-SAT predictions relatively close to the confirmed
C-SAT score. Considering that the best performing classifier is supposed be better at
identifying the negative class than the baseline models given the lower \( fpr \) it seems
odd that the classifier is finding less negative observations. An hypothesis may be that
there actually is a larger number of positive observations in the unlabelled data but no
further investigation of that hypothesis will be made in this study.
7 Discussion

Our results compare relatively well to other studies. We confirm that using bigram features is better for text classification with SVM as previously presented in other studies (Wang and Manning, 2012). Our experiments also show that implementation of feature scoring metric Log Odds Ratio improve performance for datasets with divergent class distributions as presented in previous studies (Mladenic and Grobelnik, 1999; Forman, 2008) even though it was designed for use on standard corpora with the Naive Bayes classifier.

We debate whether the Precision ($ppv$), Recall ($tpr$) and F-Measure ($f1$) metrics are fit for binary classification with imprecise class skew. $f1$ may not be ideal in classification tasks with skewed data since the high $f1$ results does not accurately portray the generalization performance. $ppv$ and $tpr$ are inherently biased against the positive class which means that when there is significant skew towards the positive class, the magnitude of the negative class error rate will be minimized. The best example to support this can be found during experiments with class weights in Table 5.5 where the best performing weights measured by $tpr$ and accuracy ($w05$) does not translate as well into the combined model in Table 5.6 as compared to the model identified with the best $fpr$ from Table 5.5 ($w10$).

Even though our results show that the best possible classifier can be identified using $f1$ and accuracy in the final results on test data we can take an example to show where the metrics perform worse. Looking at the runner-up classifier for bigram features ($t fid f$) in Table 6.1, it is very well fit for the majority class with a $tpr$ of 99%, however the $fpr$ is also 99% which is not reflected at all in the $f1$. This problem can be applied to accuracy as well, as proposed by other studies the accuracy metric is also inadequate when comparing performance (Provost and Fawcett, 1997) in classification tasks with skewed distributions. From these examples we may conclude that the $f1$ and accuracy is not very well fit for binary classification tasks with class skew. And to address this issue we propose focusing on error rates which makes class skew more apparent and depending on what class is skewed it might be better to focus on the ratio of false negatives ($fnr$) instead of $fpr$. This is supported by other studies on high-skew data (Provost and Fawcett, 1997).

To avoid making false conclusions about generalisation performance when comparing to other studies we consider this assumption since the data may not share the same distribution. Previous studies using Odds Ratio (Forman, 2008) feature scoring achieved best $f1$ performance of around 90% with a 1:5 class ratio as in this case but the study was using another feature scaling method called Bi-Normal Separation (BNS). BNS was not implemented in this study because of time limitations and lack of available implementations of the Normal Cumulative Distribution Function (CDF) in conjunction with LibSVM and Java. BNS is however reported to perform better in cases with high skew, but our results are much better (95%) in terms of $f1$ which may
be another indication that it is not the best metric for evaluating binary classifiers in high-skew situations.

7.1 Future work

This pilot study was done to see if the SVM classifier can be adapted to automatically predict customer satisfaction based on textual chat message data. There lies several opportunities for further developing the methods used to achieve better results. An example is to conduct a comparative study on performance between different classifiers such as Naive Bayes on similar data. Newer methods using multi-layered SVMs as presented in a recent study (Marco A Weiring, 2014) which have shown performance improvements may also be included in the comparative study. There also are possibilities to include linguistic information by analysing part-of-speech tags and aggregating a document scoring index based on word-level sentiment data by semantic indexing techniques as opposed to binary classification.

More research can be done on performance of classifier in relation to class distribution in high-skew data set situations. Using feature Bi-normal Separation (BNS) feature scaling as presented in (Forman, 2008) may be used instead of Log Odds Ratio, which should improve performance even more. Also the Grid Search algorithm can be implemented to minimize \textit{fpr} or \textit{fnr} instead of maximizing accuracy to address the irregular distribution. There also lies an opportunity to confirm the increased satisfaction score from our real-world application where the hypothesis is that people who do not answer the survey has a higher satisfaction score in general.
8 Conclusion

This study was set out to explore adaptation of the machine learning algorithm SVM for the task of automatically predicting a customer satisfaction score based on chat message data. For this task software was developed to automate the parsing of chat data, creation and evaluation of datasets and models to achieve near real-time predictions. During the adaptation process we sought to achieve the best possible classifier performance by running a series of experiments on different feature selection techniques and scaling metrics. In our feature selection experiments we implemented aggressive non-linguistic techniques such as excluding stop words, named entities and filtering based on Document Frequency to investigate ways to improve feature vector quality. To address the problem of asymmetrical class distributions in the dataset we implemented Log Odds Ratio feature scaling metric and also experimented with class weighting to penalize the majority class. We further optimized the classifier by running Grid Search algorithm on the best performing models to obtain optimal parameter values.

We achieved significant improvements compared to the baseline model by using bigrams features with Log Odds Ratio feature scaling metric and tuned parameters. The Log Odds Ratio is successfully implemented for discriminating feature class bias but the classifier required further reduction in bias by adding weight on the negative (minority) class in LibSVM to achieve even better performance. The resulting model achieved a F-Measure of 95% on test data. However, we point out the issue of using common metrics such as F-Measure in the context of binary classification with high-skew data and instead we chose to focus on the False Positive Rate which is less prone to positive class bias. By examining the False Positive Rate we showed that even the best performing model might not be satisfactory for real-world application since an expected False Positive Rate of 55% should be expected from the negative class.

The classifiers were also applied in a real-world scenario by running them on large amounts of unlabelled data to further investigate generalization capabilities. The results showed that the predicted distribution using baseline models were closest to the confirmed distribution. Interestingly the best performing classifier with lowest False Positive Rate identified less negative observations in the unlabelled data than the worse performing baseline models which might be an indication of fewer negatives in the unlabelled data.
Bibliography


