# Fashioning Data – A Social Media Perspective on Fast Fashion Brands

**Rupak Chakraborty**
Adobe Systems Inc.
A5, Sector-132, Noida – 201304
rupak97.4@gmail.com

**Senjuti Kundu**
McGill University
506 Pine Avenue, Montreal, QC, H2W1S6
senjuti.kundu@mail.mcgill.ca

**Prakul Agarwal**
University of California, Irvine
Irvine, CA 92697, United States
prakula@uci.edu

## Abstract

In this paper, we study the performance of N-gram language models on classification tasks such as sentiment analysis and spam detection and evaluate the effect of prior probability estimates on the results. Our data is in the form of public online posts pertaining to fast fashion brands, from different social media channels (Twitter and Facebook). We propose a novel ensemble model based on the combination of different N-grams in order to deal with the heteroskedastic nature of data collected from these social media channels. This has been further extended to increase the efficacy of the classification results.

## 1 Introduction

In recent years, the rise of social media channels like Twitter, Facebook and Instagram have opened new avenues for people to express their opinions and generate their own content. Companies such as Simplymeasured (www.simplymeasured.com) and Gnip (www.gnip.com) aggregate data from different networks to help brands form a more complete understanding of how well they engage with users and perform online. Companies such as Metamind, Alchemy and Semantria provide online APIs for sentiment analysis and associated tasks.
Sentiment analysis is a growing area of Natural Language Processing with research ranging from document level classification (Pang and Lee [1]) to learning the polarity of words and phrases (Esuli and Sebastiani [2]). Given the character limitations on tweets, classifying the sentiment of Twitter messages is most similar to sentence-level sentiment analysis. Some researchers have explored the use of part-of-speech features [3] but results remain mixed. Researchers in [4] and [5] have reported different ways of automatically collecting training data by relying on emoticons for defining the sentiment labels in their training data. However as per our observations the presence of an emoticon does not necessarily divulge its sentiment and hence we have taken the approach of manually labeling the training and test data. Da Silva et al. [6] have introduced an approach of using classifier ensembles to determine the sentiment of tweets. However they only consider a binary classification of tweets (i.e. positive and negative) and use a heterogeneous ensemble of classifiers like Multinomial Naïve Bayes, SVM, Logistic Regression and Random Forests.
Agarwal et al [7] propose a method of sentiment analysis using a tree kernel and a set of hand crafted POS features. Twitter hashtags have been extensively used in [8] to train a classifier using the Adaboost algorithm. In recent years several competitions like SemEval 2014 have included sentiment analysis of tweets as a major task. This has led to several state-of-the-art performances like [9] where the authors have used a Deep Learning approach using a combination of sentiment specific semantic word embeddings and hand crafted features. The authors in [10] have enhanced Twitter sentiment classification using contextual information like geolocation, timezones etc.
While a great deal of recent research has focused on sentiment analysis of Twitter data and spam detection (Wang et al [11]) less attention has been devoted to extending these classification tasks to public Facebook posts. Furthermore, while domains such as politics (Bakliwal et al. [12]; Yang et al. [13]) and sports (Hong and Skiena [14]) have received strong coverage, the genre of commercial

fashion brands has not been mined as frequently for predictive and classification tasks

The absence of literature on cross channel sentiment analysis with a special focus on the implications of prior distributions on the classification results has motivated us to undertake the following study. The niche segment of fast fashion brands was chosen because it remains largely unexplored. An attempt has been made to provide a clear comprehensive view of the performance metrics across different channels (Twitter and Facebook in our case) while noting the difference in trends among them.

The major contributions of the present work are as follows: We have collected and manually annotated a dataset** of posts from major social media channels (Twitter and Facebook). Our dataset was based on posts about fast fashion brands, thereby extending the application of sentiment analysis and spam detection to this infrequently-explored genre. The inclusion of more than one social channel provides a cross sectional view of the social media spectrum. It was also helpful in gauging the performance of the same algorithm on channels with disparate content (different in terms of syntactic and semantic structure).

We have extensively analyzed the effect of priors on the associated tasks and compared different N-gram models on many statistical performance metrics like Accuracy, Precision, Recall, Specificity and F1-score. Use of N-grams as features obviates the need for tedious feature engineering which often entails a classification task. The proposed generative ensemble model provides an easily implementable and lightweight framework which can be extended to any classification problem. This is because it does not make any implicit/explicit assumption about the nature or distribution of the data. Thus, we have developed a roadmap for cross channel text analysis and classification, thereby providing a unified and holistic view of any topic or subject (fashion brands in our case).

The key brands identified were fast fashion brands (Zara, Forever 21, H&M etc.) which target young customers in their late teens and early twenties and have a high turnover rate as part of their business strategy. We were particularly interested in studying this demographic since their customers frequently take to social media to express their satisfaction or dissatisfaction with products purchased. Due to the high turnover rate and short shelf-life of most of their products, opinions about their newest items are created every few months. This made data collection easier and more attuned to public opinion.

We divide the paper as follows: In Section 2 we discuss the collection and distribution of data in details, we also include the steps taken for pre-processing the data, in Section 3 the algorithms used for spam detection and sentiment analysis have been detailed along with intuitive explanation of why they work, in Section 4 we present our experiments and observations which includes a detailed analysis of the proposed algorithm for each channel along with a cross channel view of different performance metrics, in Section 5 we conclude the paper.

## 2 Data Collection and Distribution

We collected data in the form of posts from Facebook and Twitter over a period of 3 months (August to November 2015). The nine selected fashion brands were Forever 21, Mango, Levis, H&M, Guess, Free People, True Religion, Rag & Bone and 7 For All Mankind.

We have used the official Twitter Search API to fetch the data. Tweets containing hashtags and names of the mentioned brands were selected. Apart from these we have sieved tweets containing the official Twitter handles of these brands. For example, searching by "@7fam" retrieves tweets containing the tag @7fam which corresponds to the official Twitter handle of 7 for all mankind.

In case of Facebook, posts were fetched from the official pages of the selected brands using the official Facebook Graph API. Since the brands advertise their latest arrivals through these pages the posts were filtered by the author names, so any self-advertising content has been excluded.

From our observations Twitter is a more fast paced channel (i.e. frequency of posts is more) in comparison to Facebook. This is reflected in the significantly lower number of Facebook posts. On the other hand Facebook posts are more informative and verbose compared to 140 character tweets.

Each post has been hand labelled. For sentiment analysis each post is assigned either of the three class labels positive, negative or neutral depending on the content. In case of spam detection it is assigned a binary label of spam and ham/not spam.

**Dataset can be downloaded from: https://github.com/Senjuti/Dataset-Fashion-Analytics
**Source Code is available at: https://github.com/Senjuti/Fashion-Analytics

Each post has been labelled by two annotators. The average agreement between the two annotators has been observed to be 83.2%

We are not detecting spam in a traditional sense here, it is more of a word sense disambiguation. For example, we need to filter out posts pertaining to Mango the fruit from those relating to the actual denim brand. Similarly we need to distinguish between Levis stadium and Levis the company. As most of the fashion brand names we have selected are proper or common nouns the need to add this additional filter arose, which has been modelled as a spam detector. Additionally posts promoting freebies and advertising one's own fashion collection have been included in the category of spam because they do not contain useful opinion words or sentences and hence they introduce noise in the training data instead of adding value to it.

Each post is passed through a pre-processing pipeline before extracting the N-gram features. The salient steps in the pipeline are as follows – 1. Cleaning the data of URLs, HTML tags, punctuation marks, emoticons and similar noise. 2. Stopword removal (using an English stopword lexicon). 3. Stemming – Reducing each word to its root word using the Snowball Stemmer which is based on Porters Stemmer.

## 2.1 Distribution of the Training Data

**Table 1: Distribution of Train Data for Sentiment Analysis**

| Channel | Positive | Negative | Neutral | Total |
|---|---|---|---|---|
| Facebook | 165 (11%) | 46 (3%) | 1337 (86%) | 1548 |
| Twitter | 987 (27%) | 393 (11%) | 2893 (62%) | 3705 |

**Table 2: Distribution of Train Data for Spam Analysis**

| Channel | Spam | Ham | Total |
|---|---|---|---|
| Facebook | 253 (14%) | 1548 (86%) | 1801 |
| Twitter | 1220 (25%) | 3705 (75%) | 4925 |

Table 1 lists the distribution of training data (posts) for sentiment analysis as can be inferred, the class label distribution is highly skewed, majority of the posts are neutral for both the channels. The data for spam detection is equally biased (i.e. the majority of the posts are not spam) as can be seen in Table 2. This kind of a data distribution closely resembles real-life scenarios where majority of the online posts are likely to belong to a single class. Motivated by this skewed data distribution, an attempt has been made to make the classifier fairly invariant to class label distribution which led to the proposed ensemble model.

## 2.2 Distribution of the Test Data

**Table 3: Distribution of Test Data for Sentiment Analysis**

| Channel | Positive | Negative | Neutral | Total |
|---|---|---|---|---|
| Facebook | 192 (20%) | 67 (7%) | 712 (73%) | 970 |
| Twitter | 369 (23%) | 138 (8%) | 1120 (69%) | 1627 |

**Table 4: Distribution of Test Data for Spam Analysis**

| Channel | Spam | Ham | Total |
|---|---|---|---|
| Facebook | 41 (4%) | 970 (96%) | 1011 |
| Twitter | 613 (25%) | 1627 (75%) | 2440 |

The test data distribution closely tails the training data (as can be seen from tables 3 and 4). This provides a level ground for measuring model performance, though it would be interesting to see the performance on a uniformly distributed dataset. In order to get a clearer picture of classifier performance, metrics like precision and specificity have been included. This skewed distribution makes the effect of priors more prominent thereby enabling us to study them for our specific test conditions.

## 3. Algorithms

Naïve Bayes classifier with N-grams as the features has been used. Algorithm 1 (see section 3.1) is used to combine the output of different N-gram models in order to calculate a single class probability. Each classifier outputs two things: the probability of the most likely class and its corresponding class label. After experimenting with different N-gram models (N = 1 to N = 7), it can be seen that there is no significant gain in performance after N = 5 on the test data. So for the proposed algorithm N-gram models up to only order 5 have been used.

The use of Naïve Bayes as the classifier of choice has been motivated by the fact that our main intention is to study the effect of priors using N-gram features and Naïve Bayes has proven to be very effective for many text classification tasks, often matching state-of-the-art results obtained by using SVMs and the like. Algorithm 1 proposed aggregates the outputs of all the classifiers (each classifier is trained on a different N-gram model, with values of N ranging from 1 to 5) and predicts the final class label as the one which has highest probability among the five input class labels.
The reason why this approach increases the performance is - N-gram models match the N-grams in the test sample with the probabilities of the N-grams calculated during training, thus a larger posterior probability implies a greater degree of match of the N-grams of the test sample with the training set. Hence, intuitively a greater probability increases the likelihood that the model has seen a similar post/sample during training, so taking the class label of the maximum probable N-gram as the final prediction increases the chances of a correct classification. The algorithms have been presented in the form of a pseudocode where each step has been clearly mentioned, it has been further grouped into separate training and testing phases (with additional hyperparameter adjustment phase wherever necessary). Algorithm 2 (see section 3.2) is our proposed ensemble algorithm which is built on top of Algorithm 1.

## 3.1 Algorithm 1: Combining Different N-grams

**Training Phase:**

**Input:** Training data after being passed through the preprocessing pipeline. Each sample consists of N-gram features and its associated class label.

**Output:** Trained N-gram models (N = 1 to N =5) using the maximum likelihood probability estimate followed by add one smoothing.

**Testing Phase**:

**Input:** An unknown test sample.

**Output***:* The class label (of the most likely class) to which the sample belongs.

**Step 1:** For a given test sample calculate the pair *(Pi, Ci)* where *Pi* is the probability value output of the *ith* N-gram model (in our case *i* = 1 to 5) and *Ci* is its corresponding class label. So we will have five such pairs of *(Pi, Ci)*.

**Step 2:** Find the max value of *Pi* and take its corresponding *Ci* to be the final class label.

**Step 3:** Output the class label calculated in the Step 2

Algorithm 1 proposed cannot effectively deal with the skewness in the data. This is because it is trained on the entire dataset and hence its performance is limited by and equal to the best performing N-gram model (which is N = 5 in most cases across both classification tasks). In order to deal with the inherent single class bias of the dataset the training data has been randomly split into N mutually exclusive and exhaustive parts. N-gram models have been trained on each of these N parts using Algorithm 1 and the weighted output of these N-gram models has been used to predict the final class label. The proposed algorithm for ensemble learning includes a hyper-parameter adjustment phase where the weight of each model is calculated based on its performance on a held out set.

Randomly splitting up the training data into mutually exclusive and exhaustive parts reduces the effect of a single class bias which is prevalent in the data. By training individual models on each of these disjoint parts of the feature space, each model receives the unique ability to learn the final hypothesis differently. This prevents overfitting the data which often plagues individual N-gram models trained on the entire dataset. The weights of the models are tuned as per their performance on a held out set. Hence, the predictions of well performing models are given more weightage in comparison to those with lower performance on the held out set. Taking the final prediction (i.e. final class label) as the modal (most frequently occurring class) class label of the weighted output of the individual models ensures that the predicted class label is the one on which majority of the models agree upon. Thus, in spirit our ensemble model works in the same way as Random Forests [15] which have shown to be better at learning final hypothesis in comparison to individual decision trees.

### 3.2 Algorithm 2: Ensemble of Combined N-Grams

**Training Phase**:

**Input**: Training data after being passed through the preprocessing pipeline. Each sample consists of N-gram features and its associated class label

**Output**: A total of K N-gram models each trained using Algorithm 1.

**Step 1**: Randomly split the training dataset into N mutually exclusive and exhaustive parts.
**Step 2**: For each part $N_i$ train model $K_i$ by Algorithm 1.
**Step 3**: Save these models for testing and later use.

**Hyper parameter Adjustment Phase:**

**Input**: Trained models $K_i$ and held out dataset.

**Output**: Model Weights $W_i$ corresponding to each $K_i$

**Step 1**: For each model $K_i$ calculate the number of correct predictions ($X$i) and the number of incorrect predictions ($Y$i).
**Step 2**: Calculate weight $W_i$ of each model as $e^{xi/(yi+1)}$
**Step 3**: Save the model weights $W_i$

**Testing Phase:**

**Input**: An unknown test sample.

**Output**: The class label (of the most likely class) to which the sample belongs.

**Initialize**: $f = \{\ \}$ (Empty List to hold the class labels)

**Step 1**: For each model $K_i$ calculate the class label $C_i$ using Algorithm 1.
**Step 2**: For each $K_i$ let $Q_i = W_i \, x \, C_i$ (i.e. each class label $C_i$ is counted $W_i$ times)
**Step 3**: For each $K_i$ $f = f \, U \, Q_i$ (i.e. Add the class label computed in Step 2 to the list)
**Step 4**: Output the final predicted class label as the modal class of $f$

## 4. Experiments and Results

N-gram models have been trained for different values of N (N = 1 to N = 7). During the classification phase using the Naïve Bayes classifier, we have experimented under two conditions
1. Calculating the class label probabilities while discounting the prior probabilities learned during the training phase (i.e. assuming equal prior class distribution). 2. Taking the prior probabilities into account while calculating the class label probabilities.
In order to get a clear insight into the effects of priors on different classification tasks across different channels, we have studied each scenario independently. First we present the results of each channel separately on the twin classification tasks then we provide a cross channel view of the effect of priors.

### 4.1 Results for Facebook

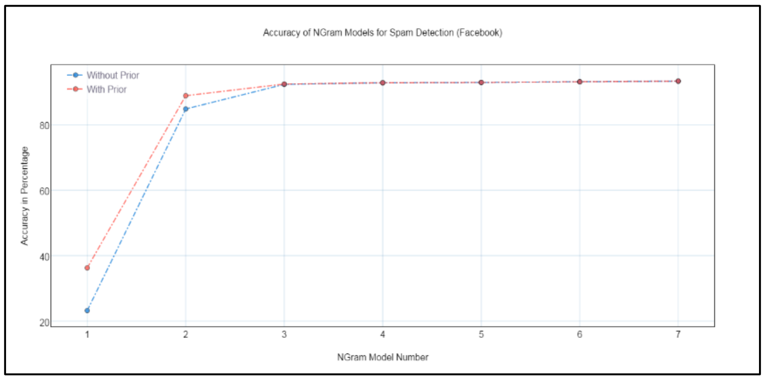

**Figure 1: Comparison of N-gram Models for Spam Detection on Facebook**

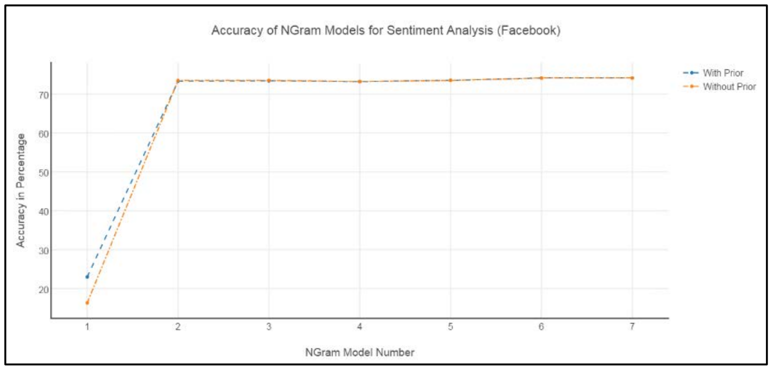

**Figure 2: Comparison of N-gram Models for Sentiment Analysis on Facebook**

Figure 1 illustrates the performance (in terms of accuracy) of different N-gram models for spam detection on Facebook, as can be seen from the figure the effect of priors is more profound on the unigrams and bigrams in comparison to the higher order N-grams. Another notable feature is how the performance almost plateaus after N=4 for both cases (i.e. taking the prior into account and discounting the prior).  Figure 2 juxtaposes the performance of different N-gram models for sentiment analysis on Facebook, it follows the same trend as in figure 1 but the overall classification accuracy is less in this case, the best performing N-gram model (N =  7)  has an accuracy of 74.15% for sentiment analysis while it is 93.28% (for N = 7) in case of spam detection . The obvious reason for this is that spam detection is a two class classification problem (only 2 class labels spam and ham), while sentiment analysis is a three way classification task (positive, negative and neutral class labels). Thus, priors play an important role in the prediction of classes especially if the data is highly biased (as in our case).

**Table 5. Performance Metrics for Sentiment Analysis of Facebook Posts (Without Priors)**

| N-gram Model Number | Accuracy | Precision | Recall | Specificity | F1Score |
|---|---|---|---|---|---|
| 1 | 16.27 | 37.57 | 47.37 | 26.19 | 41.91 |
| 2 | 73.53 | 64.16 | 59.74 | 65.21 | 61.87 |
| 3 | 73.53 | 64.11 | 42.94 | 60.28 | 51.44 |
| 4 | 73.22 | 62.88 | 39.64 | 59.14 | 48.62 |
| 5 | 73.53 | 63.60 | 39.65 | 59.29 | 48.85 |
| 6 | 74.15 | 65.81 | 39.93 | 59.77 | 49.70 |
| 7 | 74.15 | 66.12 | 39.68 | 59.59 | 49.59 |

**Table 6. Performance Metrics for Sentiment Analysis of Facebook Posts (With Priors)**

| N-gram Model Number | Accuracy | Precision | Recall | Specificity | F1Score |
|---|---|---|---|---|---|
| 1 | 22.96 | 42.66 | 51.99 | 32.66 | 46.87 |
| 2 | 73.32 | 65.94 | 49.63 | 62.01 | 56.64 |
| 3 | 73.42 | 64.87 | 42.20 | 59.93 | 51.13 |
| 4 | 73.22 | 62.88 | 39.64 | 59.14 | 48.62 |
| 5 | 73.53 | 63.60 | 39.65 | 59.29 | 48.85 |
| 6 | 74.15 | 65.81 | 39.93 | 59.77 | 49.70 |
| 7 | 74.15 | 66.12 | 39.68 | 59.59 | 49.59 |

Tables 5 and 6 elucidates the performance metrics of N-gram models for sentiment analysis on Facebook, in order to evaluate the effect of priors on different N-gram models, we take the results of table 5 as the baseline. There is a spike in the accuracy of unigrams (41% over the baseline) while for the rest there is marginal (in case of bigrams and trigrams) or no decrease (for N >= 4). The same trend is visible across other performance parameters like precision, recall and specificity. A possible explanation for this behavior is that Facebook posts are mostly highly structured (i.e. in proper English) in comparison to tweets and hence higher order N-grams effectively model the language structure which obviates the effects of prior probabilities as is evident from the results.

**Table 7. Performance Metrics for Spam Detection of Facebook Posts (Without Priors)**

| N-gram Model Number | Accuracy | Precision | Recall | Specificity | F1Score |
|---|---|---|---|---|---|
| 1 | 23.22 | 48.60 | 43.99 | 44.81 | 46.18 |
| 2 | 84.78 | 52.30 | 56.71 | 57.07 | 54.42 |
| 3 | 92.29 | 52.48 | 52.65 | 52.81 | 52.56 |
| 4 | 92.78 | 53.04 | 52.90 | 53.07 | 52.97 |
| 5 | 92.88 | 52.05 | 51.82 | 51.95 | 51.93 |
| 6 | 93.08 | 52.29 | 51.92 | 52.06 | 52.11 |
| 7 | 93.28 | 52.55 | 52.28 | 52.16 | 52.28 |

Tables 7 and 8 depict the performance of spam detection on Facebook they are exactly are in the same vein as the results for sentiment analysis. There is a massive increase in the accuracy for unigrams (56%) when we include priors into the equation however this has side effects of decreasing the precision and recall.

**Table 8. Performance Metrics for Spam Detection of Facebook Posts (With Priors)**

| N-gram Model Number | Accuracy | Precision | Recall | Specificity | F1Score |
|---|---|---|---|---|---|
| 1 | 36.26 | 48.57 | 41.67 | 42.27 | 44.86 |
| 2 | 88.83 | 54.16 | 58.82 | 59.18 | 56.39 |
| 3 | 92.39 | 52.58 | 52.70 | 52.86 | 52.64 |
| 4 | 92.78 | 53.04 | 52.90 | 53.07 | 52.97 |
| 5 | 92.88 | 52.05 | 51.82 | 51.95 | 51.93 |
| 6 | 93.08 | 52.29 | 51.92 | 52.06 | 52.10 |
| 7 | 93.28 | 52.02 | 52.02 | 52.16 | 52.28 |

**Table 9: Performance Metrics for Sentiment Analysis on Facebook (Proposed Method i.e. Algorithm 2)**

| Ensemble Model | Accuracy | Precision | Recall | Specificity | F1-Score |
|---|---|---|---|---|---|
| Without Prior | 66.69 | 51.94 | 60.95 | 61.65 | 56.09 |
| With Prior | 88.04 | 50.77 | 51.57 | 51.76 | 51.17 |

**Table 10: Performance Metrics for Spam Detection on Facebook (Proposed Method i.e. Algorithm 2)**

| Ensemble Model | Accuracy | Precision | Recall | Specificity | F1-Score |
|---|---|---|---|---|---|
| Without Prior | 56.95 | 49.62 | 65.78 | 58.68 | 56.57 |
| With Prior | 76.10 | 62.61 | 66.73 | 68.66 | 64.61 |

Tables 9 and 10 depict the performance of the proposed method on sentiment analysis and spam detection respectively. The improvements by including priors is measured against the baseline of discounting priors. Though there is an improvement in accuracy for both classification tasks, however the precision, recall and specificity in case of spam detection decreases marginally. There is an increase of 33.62% in accuracy for sentiment analysis whereas for spam detection there is a leap of 32.01%. Precision for sentiment analysis improves by 26.17% whereas for spam detection a decline of 2% is noted. Recall and specificity improvements for sentiment analysis is negligible hovering somewhere around 1%. However in case of spam detection both decrease by 1.5% and 1.6% respectively, this may be attributed to the overwhelming presence of ham labels in the test data (about 75%).Thus, as a general trend the effect of priors becomes more dominant with the increase in the number of classes in data especially if the distribution of class labels is skewed.

Facebook has unique challenges in comparison to Twitter. First the verbosity of the posts in comparison to 140 character tweets is something which needs to be taken into account. It takes a longer time to make predictions because of the greater number of N-grams. Second, the posts are in proper English and the use of emoticons and acronyms is less in comparison to Twitter.

## 4.2 Results for Twitter

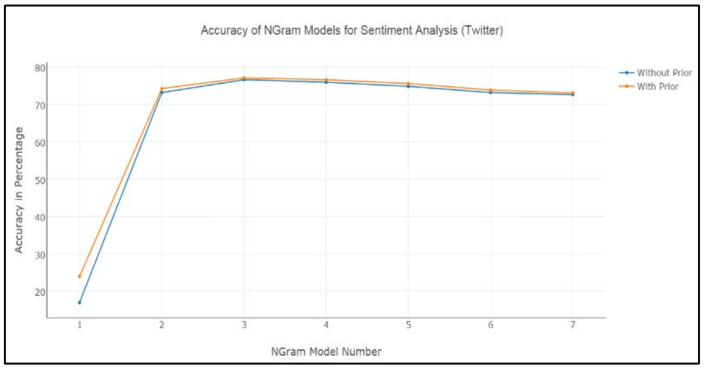

**Figure 3: Comparison of N-gram Models for Sentiment Analysis of Tweets**

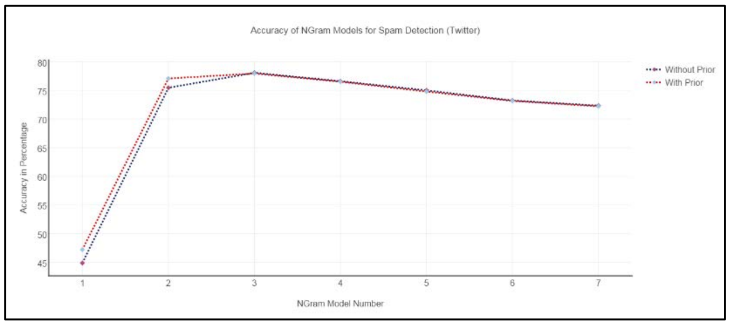

**Figure 4: Comparison of N-gram Models for Spam Detection of Tweets**

Figures 3 and 4 depict the performance of N-gram models for sentiment analysis and spam detection on Twitter data, they follow the same general trend as the N-gram models for Facebook. Similarly the exclusion of prior probabilities (i.e. considering equal prior distribution for all classes) has a deeper impact on the lower order N-grams in comparison to the higher order N-grams (N >= 3). The intuitive reason for this may be that unigrams and bigrams assume a higher degree of conditional independence of the different words in a sentence hence, cannot effectively model the syntactic and semantic level dependencies of the language. The differentiating factor of the Twitter N-gram models is that their performance decreases for higher order N-grams (N >= 3) whereas in case of Facebook it is almost constant. Also the overall accuracy of the Twitter models is less than the Facebook ones both in case of sentiment analysis and spam detection.

**Table 11: Performance Metrics for Sentiment Analysis of Tweets (Without Priors)**

| N-gram Model Number | Accuracy | Precision | Recall | Specificity | F1Score |
|---|---|---|---|---|---|
| 1 | 16.95 | 37.46 | 40.05 | 23.01 | 38.71 |
| 2 | 73.21 | 69.41 | 69.55 | 71.49 | 69.48 |
| 3 | 76.65 | 75.08 | 69.02 | 73.27 | 71.92 |
| 4 | 75.98 | 74.47 | 66.61 | 71.66 | 70.32 |
| 5 | 74.87 | 73.34 | 61.99 | 69.24 | 67.19 |
| 6 | 73.21 | 72.04 | 57.93 | 66.17 | 64.22 |
| 7 | 72.65 | 71.86 | 54.65 | 64.52 | 62.09 |

**Table 12: Performance Metrics for Sentiment Analysis of Tweets (With Priors)**

| N-gram Model Number | Accuracy | Precision | Recall | Specificity | F1Score |
|---|---|---|---|---|---|
| 1 | 23.95 | 34.65 | 40.72 | 28.96 | 37.44 |
| 2 | 74.26 | 71.81 | 67.98 | 71.44 | 69.84 |
| 3 | 77.14 | 75.67 | 68.47 | 73.34 | 71.89 |
| 4 | 76.65 | 75.33 | 66.37 | 71.92 | 70.56 |
| 5 | 75.61 | 74.42 | 61.89 | 69.62 | 67.58 |
| 6 | 73.89 | 73.19 | 57.92 | 66.59 | 64.67 |
| 7 | 73.09 | 72.94 | 54.17 | 64.70 | 62.17 |

**Table 13: Performance Metrics for Spam Detection of Twitter Posts (Without Priors)**

| N-gram Model Number | Accuracy | Precision | Recall | Specificity | F1Score |
|---|---|---|---|---|---|
| 1 | 44.89 | 51.99 | 51.87 | 51.92 | 51.93 |
| 2 | 75.46 | 73.51 | 68.02 | 68.08 | 70.66 |
| 3 | 78.08 | 82.32 | 68.36 | 68.42 | 74.69 |
| 4 | 76.60 | 81.45 | 66.03 | 66.08 | 72.93 |
| 5 | 75.01 | 80.43 | 63.51 | 63.56 | 70.98 |
| 6 | 73.24 | 79.29 | 60.71 | 60.76 | 68.77 |
| 7 | 72.34 | 79.09 | 59.21 | 59.25 | 67.72 |

**Table 14: Performance Metrics for Spam Detection of Twitter Posts (With Priors)**

| N-gram Model Number | Accuracy | Precision | Recall | Specificity | F1Score |
|---|---|---|---|---|---|
| 1 | 47.23 | 50.77 | 50.82 | 50.87 | 50.87 |
| 2 | 77.08 | 77.14 | 68.67 | 68.72 | 72.66 |
| 3 | 77.95 | 82.31 | 68.15 | 68.20 | 74.56 |
| 4 | 76.52 | 81.82 | 65.78 | 65.83 | 72.93 |
| 5 | 74.80 | 80.36 | 63.17 | 63.22 | 70.74 |
| 6 | 73.16 | 79.36 | 60.56 | 60.60 | 68.70 |
| 7 | 72.22 | 78.94 | 59.02 | 59.07 | 67.54 |

For sentiment analysis the best performing Facebook model (N = 7) achieves an accuracy of 74.15% whereas the Twitter N-gram has an accuracy of 76.65%, in case of spam detection the accuracy is 93.28% and 77.95% for Facebook and Twitter respectively. Tables 11 and 12 illustrate the performance of different N-gram models for sentiment analysis, as can be seen the effect of priors is deeper on unigrams and bigrams in comparison to the higher order N-grams. By taking the performance of the N-grams without prior (table 11) as the baseline we notice the following improvements – for unigrams there is a massive improvement in accuracy (41.30%) but at the cost of a decrease in precision of 7.47%. For rest of the N-grams (N >= 2) a marginal increase in accuracy (1.42% to 0.64%) entails an increase in precision and specificity, with an expected dip in recall (2.25% to 0.05%).

Tables 13 and 14 describe the performance of the N-gram models for spam detection, the performance gain for unigrams and bigrams is marginal while it decreases for rest of the higher order N-grams when taking the priors into account. This is consistent with the general trend for Twitter as shown in [16]. The decrease in accuracy (for N >= 3) is in tandem with the dip in the precision, recall, specificity and f1-score.Unigrams perform uniquely while including the priors, there is an increase in accuracy of 5.1% while the precision decreases by 2.36% at the same time.

**Table 15: Performance Metrics for Sentiment Analysis of Tweets (Proposed Method i.e. Algorithm 2)**

| Ensemble Model | Accuracy | Precision | Recall | Specificity | F1-Score |
|---|---|---|---|---|---|
| Without Prior | 47.42 | 47.16 | 56.65 | 51.44 | 51.44 |
| With Prior | 62.03 | 52.22 | 54.50 | 59.96 | 53.33 |

**Table 16: Performance Metrics for Spam Detection of Tweets (Proposed Method i.e. Algorithm 2)**

| Ensemble Model | Accuracy | Precision | Recall | Specificity | F1-Score |
|---|---|---|---|---|---|
| Without Prior | 69.76 | 68.61 | 70.82 | 70.88 | 69.69 |
| With Prior | 75.01 | 72.58 | 67.93 | 67.99 | 70.18 |

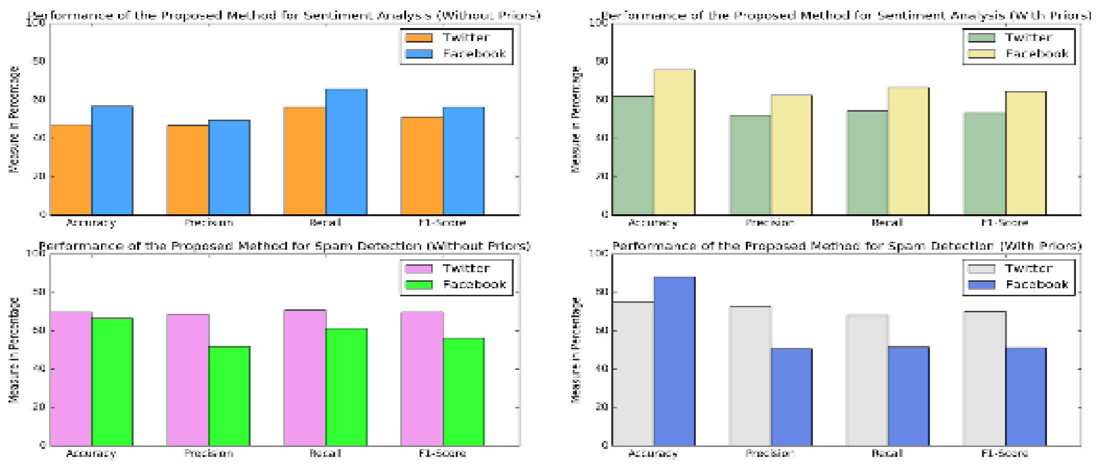

**Figure 5. Compares the relative performance of the proposed method across the channels (Twitter and Facebook) for the twin tasks of sentiment analysis and spam detection. The first row contains results of sentiment analysis under two conditions considering the prior probability and discounting it. The second row juxtaposes the results for spam detection for the same channels under the same conditions. They have been compared across four performance metrics like Accuracy, Precision, Recall and F1-measure**.

Tables 15 and 16 depict the performance of the proposed algorithm for sentiment analysis and spam detection respectively. As stated before, they are compared under two given conditions including the priors and excluding them. In order to compare the effect of priors on the proposed model, the performance of the proposed ensemble algorithm while discounting priors is set as the baseline. Inclusion of prior probabilities in the proposed model has a marked improvement in accuracy and precision, while there is a marginal improvement in the f1 measure. For sentiment analysis there is a leap of 30.80% in the accuracy while for spam detection an improvement of only 7.5% has been observed. Precision for sentiment analysis increases by 10.72% while for spam detection an increase of 5.78% in seen. Recall decreases as the precision improves since they are inversely proportional to each other, the decrease in recall for spam detection is 4.08% while for sentiment analysis it is 3.79%. Another notable thing in case of spam detection is the decrease in specificity by 4.07% which means it classifies some spam tweets as ham, although it is acceptable in our use case since we are not doing spam detection in a traditional sense.F1 measure for sentiment analysis increases by 3.67% while for spam detection is it a meagre 0.7%.

An important thing to note about Twitter is that the proposed algorithm does not perform as well as the naïve N-grams (in terms of accuracy and precision) for both the tasks of spam detection and sentiment analysis.

For sentiment analysis the proposed method with priors has an accuracy of 62% while for N-gram it has an accuracy of 76.60% (for N = 3).in case of spam detection the difference is marginal
in terms of accuracy but more profound for precision, 72% versus 83.31% (for trigrams).

### 4.3 Cross Channel View of Results

Figure 5 illustrates a cross channel view of the proposed method under two experimental conditions (with and without considering the prior probability distributions) and across a set of four performance metrics - accuracy, precision, recall and f1-measure. As is evident from the figure, for sentiment analysis the Facebook models outperform the Twitter models by a sizable margin however the tables are turned in case of spam detection where the Twitter models are the clear winner across the different metrics. The only metric where they lag behind Facebook is in the case of accuracy (while taking priors into account).

An important conclusion to draw from the given results is that priors affect not only accuracy but also precision and recall – which are perhaps more important metrics for a classifier. Priors also affect different channels differently: the effects are more significant for Facebook than Twitter (in terms of increase of accuracy precision and specificity. For small datasets the priors have been found to be more effective than large uniform datasets.)

In the proposed algorithm since we are training each model on a random subset of the dataset the prior probability distribution is different for each model hence their impact is much more significant in comparison to vanilla N-grams which are trained on the entire dataset.

## 5 Conclusion and Future Work

In the future we plan to extend our work to include other social media channels like Instagram and Reddit in order to study the effect of the proposed algorithm on other datasets, thereby providing a more comprehensive view of the performance of our classification strategy across the social media spectrum. It will be interesting to see the performance of the proposed algorithm on a larger dataset and validate if the results reported here are consistent with the increase in data size.
The data preprocessing pipeline can be enhanced by the addition of emoticon detectors, acronym lexicons and spell checkers. In place of the currently used naïve Add One smoothing, other sophisticated smoothing techniques such as Good-Turing, Witten-Bell and modified Keysner smoothing could be used.

The accuracy rate can also be improved by augmenting the feature set using POS tags, word polarity and punctuation marks. The effect of including hashtags, in case of Twitter, could also be studied. The proposed ensemble model can be further improved by adjusting the hyper-parameters of each individual model to reflect the accuracy on a per class basis (not using average accuracy as is presently done), thereby enabling each model to respond to different class labels differently. Additionally we would also like to explore a smarter way of combining the output of the ensemble by using a neural gating network as is often done.

## Acknowledgments

The authors would like to thank their family, friends and colleagues for their constant help and support.

## References

[1] Pang, B., and Lee, L. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval* 2(1-2):1–135.

[2] Esuli, A., and Sebastiani, F. 2006. SentiWordNet: A publicly available lexical resource for opinion mining. In *Proceedings* of LREC.

[3] Barbosa, L., and Feng, J. 2010. Robust sentiment detection on Twitter from biased and noisy data. In *Proc. of Coling*.

[4] Bifet, A., and Frank, E. 2010. Sentiment knowledge discovery in Twitter streaming data. In *Proc. of 13th International Conference on Discovery Science*.

[5] Pak, A., and Paroubek, P. 2010. Twitter as a corpus for sentiment analysis and opinion mining. *In Proc. of LREC*

[6] Da Silva, Nadia FF, Eduardo R. Hruschka, and Estevam R. Hruschka. "Tweet sentiment analysis with classifier ensembles." *Decision Support Systems* 66 (2014): 170-179.

[7] Agarwal, A., Xie, B., Vovsha, I., Rambow, O., and Passonneau, R. 2011. Sentiment analysis of Twitter data. Proceedings of the Workshop on Languages in Social Media (LSM '11). *Association for Computational Linguistics*, Stroudsburg, PA, USA, 30-38.

[8] Kouloumpis, Efthymios, Theresa Wilson, and Johanna Moore. "Twitter sentiment analysis: The good the bad and the omg!." *Icwsm 11* (2011): 538-541.

[9] Tang, Duyu, et al. "Coooolll: A deep learning system for Twitter sentiment classification." *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. 2014.

[10] Vosoughi, Soroush, Helen Zhou, and Deb Roy. "Enhanced Twitter sentiment classification using contextual information." *Association for Computational Linguistics*, 2015.

[11] Wang, Alex Hai. "Don't follow me: Spam detection in Twitter." Security and Cryptography (SECRYPT), Proceedings of the 2010 *International Conference on. IEEE*, 2010.

[12] Bakliwal, Akshat, et al. "Sentiment analysis of political tweets: Towards an accurate classifier." *Association for Computational Linguistics*, 2013.

[13] Yu, Yang, and Xiao Wang. "World Cup 2014 in the Twitter World: A big data analysis of sentiments in US sports fans' tweets." *Computers in Human Behavior* 48 (2015): 392-400.

[14] Hong, Yancheng, and Steven Skiena. "The wisdom of bookies? sentiment analysis vs. the NFL point spread." *Proceedings of the international conference on Weblogs and Social media (icWSm-2010)*. 2010.

[15] Breiman, Leo. "Random forests." *Machine learning* 45.1 (2001): 5-32

[16] Martínez-Cámara, Eugenio, et al. "Sentiment analysis in Twitter." *Natural Language Engineering* 20.01 (2014): 1-28.