A drawing of a cartoon character

Description automatically generated

Jeffrey Levesque

jlevesqu@syr.edu | Compare MNB and SVMs for Sentiment Analysis

IST-736: HW7

PRofessor gates

Table of Contents

[Introduction 2](#_Toc9812292)

[Analysis 2](#_Toc9812293)

[Data Preparation 2](#_Toc9812294)

[Results 2](#_Toc9812295)

[Unigram Models 2](#_Toc9812296)

[Unigram and Bigram Models 2](#_Toc9812297)

[Unigram (100% train) 2](#_Toc9812298)

[Conclusions 2](#_Toc9812299)

# Introduction

# Analysis

The Multinomial Naïve Bayes (MNB), and Support Vector Machines (SVM) were used to classify tweets into a series of sentiment categories. Specifically, tweets related to the HBO series Game of Thrones (GOT) were compiled, then trained against a unigram representation. Kfold validation, and confusion matrices were computed to assess model performance. Furthermore, precision, recall, and the f-ratio measures[[1]](#footnote-1) were also used. The top 25 most indicative words for positive and negative categories were determined by frequency, as well as by values corresponding to term frequency-inverse document frequency (TFIDF)[[2]](#footnote-2). Though the MNB provides the log probabilities[[3]](#footnote-3), which can be tailored to determine indicative positive and negative words, a more general approach was used. This allowed the approach to apply in a broader way, rather than devising an equivalent method for each model type.

Two additional cases were conducted in a similar fashion. However, rather than implementing a unigram representation an ngram\_range=(1,2)[[4]](#footnote-4) was utilized. This allowed models to train using a vectorized combination of both unigrams and bigrams. Moreover, using knowledge of the first two cases, a final model was constructed using the entire train dataset. This eliminated the need for a test condition, with the assumption that the first two cases satisfied initial data exploration. Additionally, since tweets are relatively, with no more than 280 characters[[5]](#footnote-5), the unigram implementation was most suitable.

# Data Preparation

# Results

## Unigram Models

## Unigram and Bigram Models

## Unigram (100% train)

# Conclusions

1. <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html> [↑](#footnote-ref-1)
2. <https://en.wikipedia.org/wiki/Tf%E2%80%93idf> [↑](#footnote-ref-2)
3. <https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html> [↑](#footnote-ref-3)
4. <https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html> [↑](#footnote-ref-4)
5. <https://developer.twitter.com/en/docs/tweets/tweet-updates.html> [↑](#footnote-ref-5)