Claremont Mckenna College

Department of Mathematical Sciences

Thesis in Data Science

SentiWordNet and VADER: Comparative Analysis of the Efficacy of Hybrid Sentiment Analysis Models

Supervisor
Prof. Mike Izbicki
Claremont Mckenna College

Author
Olivia Renfro

Academic Year
2022-2023

Abstract

Sentiment analysis is widely used in various industries, and both lexicon labeling and machine learning approaches have been extensively compared. While machine learning models have shown higher accuracy, they require manual labeling of training data, which is time-consuming and costly. This study compares the performance of eight hybrid sentiment analysis models on Twitter data using SentiWordNet and VADER polarity lexicons. Different transformation techniques were used to create a numerical feature map and fed into Linear SVM and Random Forest models. SVM with TF-IDF outperformed RF for most hybrid models, while RF performed better than SVM in almost all word-embedding variations. RF performed exceptionally well for both lexicons, even though it is less cited in sentiment analysis literature, and only SVM with TF-IDF transformations were competitive with RF. SVM consistently performed the worst with word-embedding transformations for both polarity lexicons.

Contents

Listing of figures

Listing of acronyms

ML . . . . . Machine Learning

SVM . . . . . Support Vector Machines

RF . . . . . Random Forest

NLP . . . . . Natural Language Processing

TF-IDF . . . . . Term Frequency - Inverse Document Frequency

W2V Word2Vec

POS . . . . . Part-of-Speech

1

Introduction

Sentiment analysis is a field of natural language processing (NLP) that involves analyzing text data containing opinions, behaviors, and sentiments of people towards specific issues, topics, or products [3]. With the growth of interest in sentiment analysis, the field has gained significant attention in recent years, with research now extending to applications in Urdu and other differently structured languages [4].

As the amount of user-generated content continues to increase, sentiment analysis has be-

come an insightful tool for businesses, organizations, and individuals to understand the attitudes, opinions, and perceptions of their desired demographic. Sentiment analysis has numerous applications in areas such as mercantile positions, stock trading, election polling, and more [5]. Social media platforms, particularly Twitter, have emerged as a rich source of public opinion and sentiment data in recent years. The data to analyze public opinion is continually growing as millions of users post their thoughts and feelings frequently. Despite significant progress in sentiment analysis, it remains a challenging task due to the complexity and variability of natural language.

In this thesis, the focus is on sentiment analysis of tweets from Twitter, comparing different techniques for lexicon labeling, feature transformations, and machine learning models. The texts used in this analysis were scraped from Twitter and labeled using a lexicon-based approach, which assigns sentiment scores to words and phrases based on their predefined polarity [6]. Two different dictionaries were used to analyze the performance of transformations and models for each dictionary: SentiWordNet and VADER [7, 8]. VADER is particularly attractive to compare different hybrid models due to its recorded accuracy in comparison to human-labeled data: 0.96 to 0.84 respectively [9]. VADER is also known to label small bodies of text, such as tweets, better than long documents due to the score normalization.[9] SentiWordNet has also shown to do just as well as manual labeling.[1]

However, the lexicon labeling method does not need to be pre-trained on data, which has led to significant popularity due to its ease of implementation. The compilation of polarity-

assigned dictionaries requires intense manual labor and must include cases for part-of-speech influences and implicit meanings. Standard polarity lexicons also do not include misspellings, contractions, colloquialisms, foreign words, and abbreviations and do not account for clause negation [10]. Consequently, the raw texts must be pre-processed through a series of steps before labeling, including contraction expansion, negation handling, tokenizing, POS tagging, and more.

Two main types of feature transformations were applied to the processed and labeled tweets: TF-IDF and word embedding using Word2Vec. TF-IDF, or *Term Frequency-Inverse Document Frequency*, is a commonly used technique in natural language processing that assigns importance to each word based on its frequency in a document. In this analysis, both TF-IDF and TF-IDF with a PCA transformation were considered for each polarity dictionary. Word2Vec, on the other hand, is a more recent word embedding technique that represents words as vectors in a high-dimensional space based on their distributional properties by using a two-layer neural net. Two pre-trained Word2Vec models were used in the analysis against the TF-IDF transformations, Google News 300 and GloVe 200. The Google News 300 model is trained on a large corpus of news articles and commonly used in natural language processing tasks, showing excellent performance in sentiment analysis tasks. The GloVe 200 model is specifically designed to capture the semantics and syntax of language in tweets. Pre-trained word-embedding models are a product of transfer learning in which models are trained on large amounts of data to learn neural network 'weights' that can be extracted and applied to other data.[11] The use of

a pre-trained model has shown to yield better accuracy than organically training a Word2Vec model, particularly with smaller amounts of data.[12]

In this study, we applied two machine learning models, namely Linear SVM (SVM) and Random Forest (RM), to the transformed tweets. Linear SVM is a popular linear classification algorithm that aims to find the optimal hyperplane to separate positive and negative tweets. On the other hand, Random Forest is an ensemble learning algorithm that combines multiple decision trees to improve classification performance. While Naive Bayes (NB) is another commonly used ML algorithm in sentiment analysis, particularly with an independent feature map such as with TF-IDF or word embedding, studies have shown that Random Forest outperforms NB across a variety of data types and applications, partly due to the stability of the algorithm compared to NB.[13][14] Another study also showed Linear SVM outperforming Naive Bayes in sentiment classification tasks.[15] Naive Bayes consistently outperforms other models when the data has a large number of features and observations, partly due to the high processing speed.[16] Literature on sentiment analysis using Random Forest is much more scarce than for Naive Bayes. Machine learning has been shown to lead to better accuracy results than polarity lexicon labeling [15]. However, the traditional approach to using ML in sentiment analysis requires manual labeling of a training data set that is large enough to ensure approximately accurate weights. This labeling process can be expensive, time-consuming, and still susceptible to bias or inaccuracies.

To avoid the difficulties of human-labeled data while still reaching relatively low error rates,

many have suggested the use of a hybrid approach, where data is labeled using a polarity dictionary and then modeled using ML methods. It is important to note that these combined approaches still do not outperform machine learning approaches trained with manually classified data. However, eliminating the need for large-scale manual labeling makes the hybrid model an appealing alternative to other approaches, with only a slight trade-off in performance, according to Fredrik Sommar and Milosz Wielondek [17]. Sommar and Wielondek note that their hybrid lexicon-ML model's assumptions may not apply universally, as the two components were optimized or chosen separately. Tuning certain parameters or introducing more combinations of models may lead to different accuracy results.

The goal of this paper is to compare different hybrid sentiment analysis models for their relative performance on Twitter data. We aim to provide insights into the optimal combination of lexicon labeling, feature transformations, and machine learning models for analyzing public opinion on Twitter, particularly for the SentiWordNet and VADER polarity lexicons. Eliminating manual labeling by optimizing hybrid-model techniques may make sentiment analysis more accessible with minimal loss to accuracy in practice. Our analysis will compare eight hybrid models for each lexicon, SentiWordNet and VADER, using the processes outlined in Figure 1.1. Each lexicon analysis will include accuracy and precision results from applying TF-IDF, TF-IDF with PCA, GloVe200, and Google300 before testing the Linear SVM and Random Forest parameterized models.

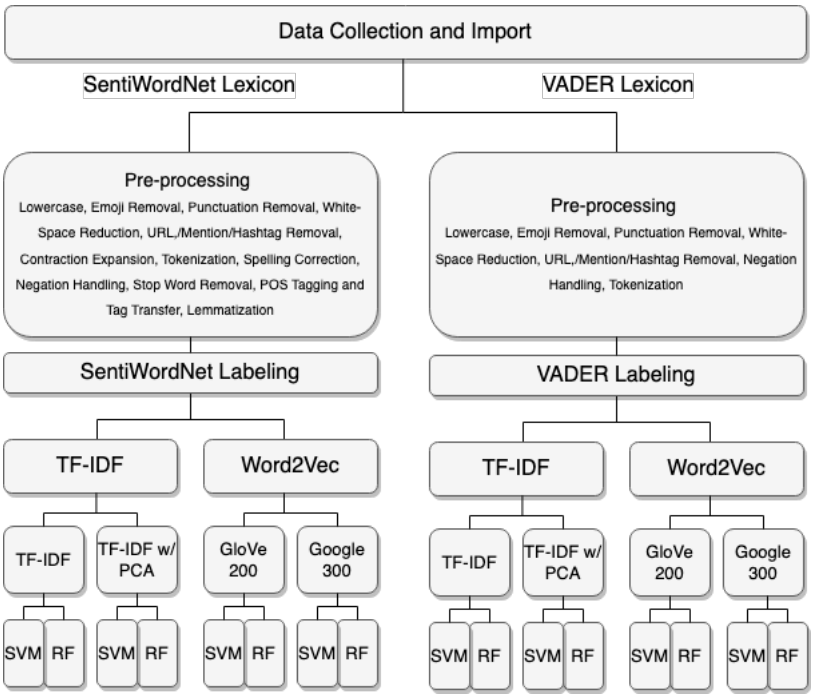


Figure 1.1: Process Overview

2

Data

2.1 Data Collection

The data used for this analysis was collected from the Apify API using two crawlers: the CheerioCrawler, which utilizes simple HTTP requests, and the PuppeteerCrawler, which accesses headless Chrome. These web crawling modules provide a parallel crawling framework for retrieving data from online web pages. Accessing Chrome without the graphical user interface

(GUI) results in better system memory utilization and global access, which increases the efficiency of the scraping tool.[18]

The API scraped data from www.twitter.com using arbitrary search queries on the words "cat," "dog," "#cat," "#dog," and two separate request completions. These requests were imported separately and joined before the data cleaning process. The distribution of sentiment values after labeling (Chapter 3) varied based on the polarity dictionary used, with overall trends indicating more positively tagged tweets than negative or neutral ones (Figure 3.2). This asymmetry is addressed in Section 5.

To ensure that the training and testing sets had enough samples in ratio to the post-feature engineering and selection dimensionality, a minimum of 1200 samples was required. After joining and preliminary cleaning of the data, there were 5295 unique observations and ten columns. The features collected in the first part of this study included 'id', 'conversation_id', 'full_text', 'url', 'view_count', 'favorite_count', 'retweet_count', 'reply_count', 'created_at', and 'hashtags' (from a series of features collapsed into one variable). In this analysis, only the columns 'id', 'full_text', and 'hashtags' were retained from the original data. However, features used in modeling included 'sentinet_tagged_sentiment', 'VADER_tagged_sentiment', and 'lemmatized'. The 'lemmatized' feature was transformed into different numerical feature maps during the model preparation stage (chapter 3).

2.2 Text Pre-processing

The quality of data is a critical determinant in the accurate classification and analysis of sentiments, especially in scenarios where implementing complex deep-learning algorithms is not feasible due to resource limitations and overhead constraints.[19] It has been estimated that a significant portion of the time allocated by a data scientist is dedicated to the data collection and cleaning stage – ranging from 50% to 80%.[20] In the field of sentiment analysis, a widely accepted practice for preparing data before feature selection involves performing operations such as cleaning, part-of-speech (POS) tagging, stemming or lemmatizing, stop-word removal, and tokenization. However, the precise steps involved in this process may vary depending on the specifics of the study, and as of yet, there is no universally acknowledged ”correct” pre-processing methodology available.[21] The standard framework for the process is as follows:

1. Clean
    - Lowercase
    - Emoji Removal/Handling
    - White-Space Reduction
    - URL, Mention and Hashtag Removal
2. Tokenize
3. Stop Word Removal
4. POS Tagging and Lemmatizing
5. Sentiment Labelling

In the context of this study, the original framework for sentiment analysis was expanded to incorporate additional steps, namely contraction expansion, spelling corrections, negation

handling, and a dedicated POS tagging procedure, with the aim of enhancing the precision of lexicon labeling. Some of these procedures were omitted for the VADER labeled data, due to the nature of the polarity assignment. The comprehensive modified procedure encompassed the following:

1. Clean
    - Lowercase
    - Emoji and Punctuation Removal/Handling
    - White-Space Reduction
    - URL, Mention and Hashtag Removal
    - Contraction Expansion
2. Tokenize
3. Spelling Correction
4. Negation Handling
5. Stop Word Removal
6. POS Tagging and Tag Transfer
7. Lemmatizing
8. Sentiment Labelling (in chapter 3)

Most of the pre-processing and labeling in this study was completed using the NLTK toolkit in Python except where otherwise specified.[22]

2.2.1 Cleaning

Meticulous attention is imperative during the cleaning stage of text data, particularly in the case of tweet data, owing to the presence of significant noise and irrelevant components, such as

user mentions, URL advertisements, emojis, misspellings, and other inconsistencies.[21] Furthermore, several words in written language serve only to contextualize feature words without contributing substantially to the sentiment value. Unfortunately, most polarity lexicons do not account for these misspellings, unique word forms, and neutral words, commonly known as "stop-words". Given that the data used in this study was sourced from Twitter, it was vulnerable to all of the aforementioned text-related difficulties. The initial steps taken post-data collection included the elimination of emojis, mentions, URLs, and hashtags, conversion of text to lowercase, expansion of contractions , and more.

After converting all text to lowercase and the removing all emojis with the "cleantext" Python package, a custom function was devised and executed on each tweet to split contractions.[23] For instance, the tweet ["I don't like dogs"] would be transformed to ["I do not like dogs"] after this modification. This operation necessitated the splitting each text string but was carried out before tokenization to prevent the string elements in each array from containing multiple words. Post-contraction expansion, regular expressions and pattern matching were used to eradicate all occurrences of emojis, mentions, URLs, and hashtags, as well as to remove any extraneous white space.[24] At this stage of the procedure, each entry in the table contained a cleaned tweet, but further processing was required to prepare the data for labeling.

2.2.2 Tokenizing

Both polarity lexicons used in this analysis required tokenizing the input text. This process involves splitting a string into an array of strings, with each word serving as an individual element. To accomplish this, the NLTK Tweet Tokenizer module was employed, which parsed the tweets. In our example, the tweet "I do not like dogs" would become ["I", "do", "not", "like", "dogs"].

2.2.3 Negation Handling

Polarity Lexicons do not account for sentence negation, in which a negator (such as "neither," "nor," and "no") may change the sentiment of the subsequent word. Many studies either omit a procedure for managing negation or introduce a system that retroactively alters the sign of the text's sentiment score, often resulting in imprecise or partially accurate outcomes. For the data employed in this research, a user-designed function was employed to manage negation by receiving tokenized text as input and generating a new word list with negation handled. This modification to the pre-processing framework has the potential to enhance lexicon sentiment labeling performance, with the author's research demonstrating that the application of the function with TF-IDF resulted in an labeling accuracy of .87, matching or surpassing processes employing more sophisticated models.[25]

The function utilizes WordNet, a lexical database in the NLTK python library, to generate cognitive synonyms, called 'synsets', for the word following a negation. It then checks for the exis-

tence of antonyms for each synonym and the word itself, and replaces the original word with the antonym with the highest dissimilarity coefficient. The dissimilarity coefficient for each antonym is calculated using 'wup_similarity' function in the following formula: dissimilarity = (1 − word1.wup_similarity(word2)). This results in a shift of the sentence's expected polarity from a word by word calculation. For example, the tweet ["I", "do","not","like","dogs"] would become ["I", "do", "dislike","dogs"], with "dislike" scoring more negatively than the combined polarities of "not" and"like".

2.2.4 Stop Word and Punctuation Removal

To further refine the text data, stop-words were removed from each observation after negation handling. The NLTK stop-word corpus was employed with some exclusions, such as against' and won'. To account for potential negation instances, a set of negation words including no', not', and 'nor' were also excluded. Moreover, tokenized punctuation was removed from the tweet string arrays during this stage.

2.2.5 Hashtag Additions and Spelling Corrections

After removing stop-words and loose punctuation from each tweet, additional measures were taken to ensure the accuracy of the process. Each hashtag for a particular tweet was appended into the tweet's array of text, and a spelling correction function was built to fix misspellings. The spellchecker Python package contains a correction method with by default finds all possi-

ble words within two characters away by removal, replacement, or insertion, and determines the best fit.[26] Misspellings include improperly spelled words and elongated words such as changing 'looove' to 'love'. One prior study found that fixing elongated words led to higher accuracy for the Linear SVM model than skipping this procedure.[16] Although often overlooked, misspellings can negatively affect the accuracy of sentiment labeling, as the polarity lexicons are limited in range and cannot account for unique spellings. The spell-correcting function checked if each word could be spell corrected, replacing misspelled words with their correctly spelled counterparts when possible. It is important to note that the spellchecker' correction method occasionally returns words that do not match the original but still have better sentiment accuracy than when omitting this stage of text pre-processing.[27]

2.2.6 POS Tagging

The next stage of text pre-processing is part-of-speech (POS) tagging which assigns each word in a sentence a specific part-of-speech tag that corresponds to the grammatical function of a word within a sentence, such as nouns, verbs, adjectives, and adverbs. For instance, in the cleaned and tokenized tweet, ["dislike", "dog"], the word dislike" would be assigned a 'verb' tag and the word dog" would be assigned a 'noun' tag. The universal POS tag set is depicted in Figure 2.1. The POS tagger used in this paper was sourced from General Architecture of Text Engineering (GATE). The "Stanford Tagger" is a CRF-based part-of-speech (POS) tagger specifically designed for English tweets and adapted from a leading algorithm in the

Tag	Meaning	English Examples
ADJ	adjective	*new, good, high, special, big, local*
ADP	adposition	*on, of, at, with, by, into, under*
ADV	adverb	*really, already, still, early, now*
CONJ	conjunction	*and, or, but, if, while, although*
DET	determiner, article	*the, a, some, most, every, no, which*
NOUN	noun	*year, home, costs, time, Africa*
NUM	numeral	*twenty-four, fourth, 1991, 14:24*
PRT	particle	*at, on, out, over per, that, up, with*
PRON	pronoun	*he, their, her, its, my, I, us*
VERB	verb	*is, say, told, given, playing, would*
.	punctuation marks	*. , ; !*
X	other	*ersatz, esprit, dunno, gr8, univeristy*

Figure 2.1: Universal POS Tagset

field.[28] Conditional Random Fields, or CRFs, are highly effective sequence modeling algorithms that enable accurate, precise, and efficient tagging by identifying patterns in speech.[29] The Stanford POS tagger was selected for its consistently superior performance compared to other taggers, as evidenced by its reported accuracy rate of 0.91.[30][28] Implementation of this tagger required the interface download at stanfordnlp.github.io and installation of Java at www.java.com/en/download. The tokenized data was tagged by accessing the Stanford NLP Server in the command line while using Python requests to generate an array of tags.

Although the Stanford POS tagging model achieved the highest reported accuracy in prior research, the Penn Treebank POS tags utilized within the model are not compatible with the NLTK package's required tags.[31] Thus, a conditional function was necessary to convert the tags into WordNet-compatible tags. Figure Figure 2.2 depicts the disparity in tag sets between the two frameworks. Any tags assigned by the Stanford model that were not present in the broader WordNet tag set were substituted with an empty string within the [word, tag] pairing.

Table 2
The Penn Treebank POS tagset.

1.	CC	Coordinating conjunction	25.	TO	*to*
2.	CD	Cardinal number	26.	UH	Interjection
3.	DT	Determiner	27.	VB	Verb, base form
4.	EX	Existential *there*	28.	VBD	Verb, past tense
5.	FW	Foreign word	29.	VBG	Verb, gerund/present participle
6.	IN	Preposition/subordinating conjunction	30.	VBN	Verb, past participle
7.	JJ	Adjective	31.	VBP	Verb, non-3rd ps. sing. present
8.	JJR	Adjective, comparative	32.	VBZ	Verb, 3rd ps. sing. present
9.	JJS	Adjective, superlative	33.	WDT	*wh*-determiner
10.	LS	List item marker	34.	WP	*wh*-pronoun
11.	MD	Modal	35.	WP$	Possessive *wh*-pronoun
12.	NN	Noun, singular or mass	36.	WRB	*wh*-adverb
13.	NNS	Noun, plural	37.	#	Pound sign
14.	NNP	Proper noun, singular	38.	$	Dollar sign
15.	NNPS	Proper noun, plural	39.	.	Sentence-final punctuation
16.	PDT	Predeterminer	40.	,	Comma
17.	POS	Possessive ending	41.	:	Colon, semi-colon
18.	PRP	Personal pronoun	42.	(	Left bracket character
19.	PP$	Possessive pronoun	43.	)	Right bracket character
20.	RB	Adverb	44.	"	Straight double quote
21.	RBR	Adverb, comparative	45.	`	Left open single quote
22.	RBS	Adverb, superlative	46.	``	Left open double quote
23.	RP	Particle	47.	'	Right close single quote
24.	SYM	Symbol (mathematical or scientific)	48.	''	Right close double quote

(a) Stanford POS Tag Set

WordNet Tag	TreeBank Tag
n	NN
a	JJ
s	JJ
r	RB
v	VB

(b) WordNet POS Conversion Tags

Figure 2.2: Tag Sets for Stanford POS and WordNet POS Tagging

2.2.7 Lemmatization

Normal speech may contain many versions of a word outside of its base form. For example, consider the words "are", "am", and "is", which are all forms of the word "be". To ensure consistency in the text data, lemmatization was performed on the tagged data. Lemmatization reduces inflectional forms of words to their base or root form, known as the lemma. This process differs from "stemming" a word which is sometimes used in place of lemmatization. The most popular stemming algorithm, Porter's algorithm, has the following rules: "sses" = "ss", "ies" = "i", "ss" = "ss", "s" = " "[32]. Although stemming has shown to be more effective than lemmatization in language analysis outside of English, the many modifications and updates to the algorithm since it's appearance in 1980 have still left the vast number of irregularly formed and conjugated words in English unaccounted for.[33] The limited rule set in stemming can produce nonsensical words or words that would not match a limited polarity lexicon; for ex-

ample, stemming the word "ponies" using Porter's algorithm would return "poni", which will not be matched by a modern lexicon.

Lemmatization helps to standardize words with the same meaning and reduces the complexity of the text data. The NLTK package was used for lemmatization in this study. NLTK's WordNetLemmatizer utilizes WordNet's built-in morphological analyzer to identify and return the base form of a given word with more fluid constraints than in a stemming algorithm.[22] The accuracy of lemmatization improves by including both elements of the [word, tag] tuples from each tweet, as collected in the last phase of text pre-processing.

3

Model Preperation

3.1 Lexicon Approach Sentiment Labelling

Following data preparation, two polarity lexicons, SentiWordNet and VADER, were used to assign sentiment values to each tweet. The labeling process produced two data sets with identical features but a different set of labels.

Opinion Lexicon	Total Sentiment Bearing Terms
General Inquirer	4216
Subjectivity Clues Lexicon	7650 (out of 8221 terms)
Grefenstette et al	2258
VADER	7500
SentiWordNet	28431 (out of total 86994 WordNet terms)

Figure 3.1: Terms per Polarity Lexicon[1][2]

3.1.1 SentiWordNet Polarity Labelling

SentiWordNet is a polarity lexicon integrated into the Python NLTK package, leveraging the WordNet databases composed of English synonyms clustered into "synsets", which are collections of synonyms that are linked to other synsets by a number of different possible relationships.[15] In contrast to typical polarity lexicons, SentiWordNet accounts for semantic orientation and strength, resulting in a more nuanced analysis of sentiment. [34][35] SentiWordNet is also much larger than other existing polarity lexicons as seen in Figure 3.1.[1][2] For each token, three numerical scores were assigned in the categories of Pos(), Neg(), and Obj(), falling between the values of 0 and 1, denoting the positive, negative, and objective sentiment values. To determine the sentiment of each tweet, the positive sentiment aggregate was subtracted from the negative sentiment aggregate.

$$\text{total sentiment} = \sum_{\text{words in tweet}} \text{positive sentiment}_{\text{word}} - \sum_{\text{words in tweet}} \text{negative sentiment}_{\text{word}}$$

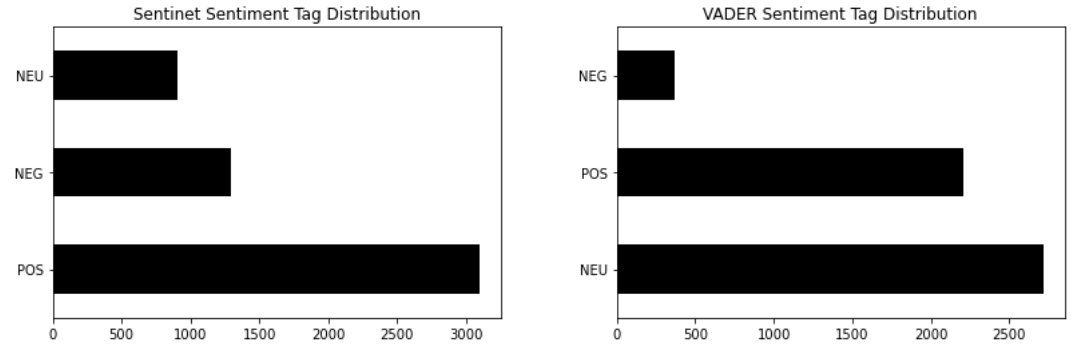
Tweets in the data table were assigned a "POS" label if the total sentiment value was greater than 0 and a "NEG" label if less than 0. If the total sentiment score was approximately zero, then the tweet was labelled as "NEU".

3.1.2 VADER Polarity Labelling

The VADER lexicon has shown higher accuracy than even human labelling, particularly with social media data.[9] VADER works to map 'emotional intensity', including though emoticons and punctuation. In contrast to SentiWordNet, VADER requires minimal data cleaning, only necessitating pre-processing steps of lowercase, emoji removal, negation handling, and URL, hashtag, and mention removal. Each word in a tweet is scored by VADER in the range of $[-4, 4]$, where -4 is the most negative and $+4$ is the most positive. The total sentiment score for a tweet is given by VADER as the sum of the sentiments of each of the words in array normalized to a $[-1, 1]$ scale. Both the VADER data set and the SentiWordNet data set showed a higher proportion of tweets with the "POS" label than "NEG" or "NEU" (Figure 3.2). As a precaution against this uneven distribution, the splitting of the data was handled very carefully and analysis results show accuracy per label.

3.2 Training/Testing Data

In order to ensure the ratio of tag types were consistent across the training and testing sets for both the SentiWordNet labeled data and the VADER labeled data, both data sets were first split



(a) Sentinet Tag Dist.

(b) VADER Tag Dist.

Figure 3.2: Sentiment Distributions of Lexicon Labeling

according to the labels "POS", "NEG", or "NEUTRAL". For each lexicon, all three sentiment subsets were split into training and testing data (75%/25%) for both the tweets and labels. Next, the lexicon's training data sets for each sentiment tag were concatenated to produce final training data sets; the lexicon's testing data sets was produced using the same technique, illustrated in **??**.

3.3 Feature Engineering: Extraction and Selection

One assumption of machine learning classification models is that the feature inputs only are of the numerical data type. Therefore, the next stage in the hybrid sentiment analysis is feature mapping. The methods chosen for feature selection and extraction were TF-IDF, TF-IDF with PCA, Word2Vec with GloVe200, and Word2Vec with Google300.

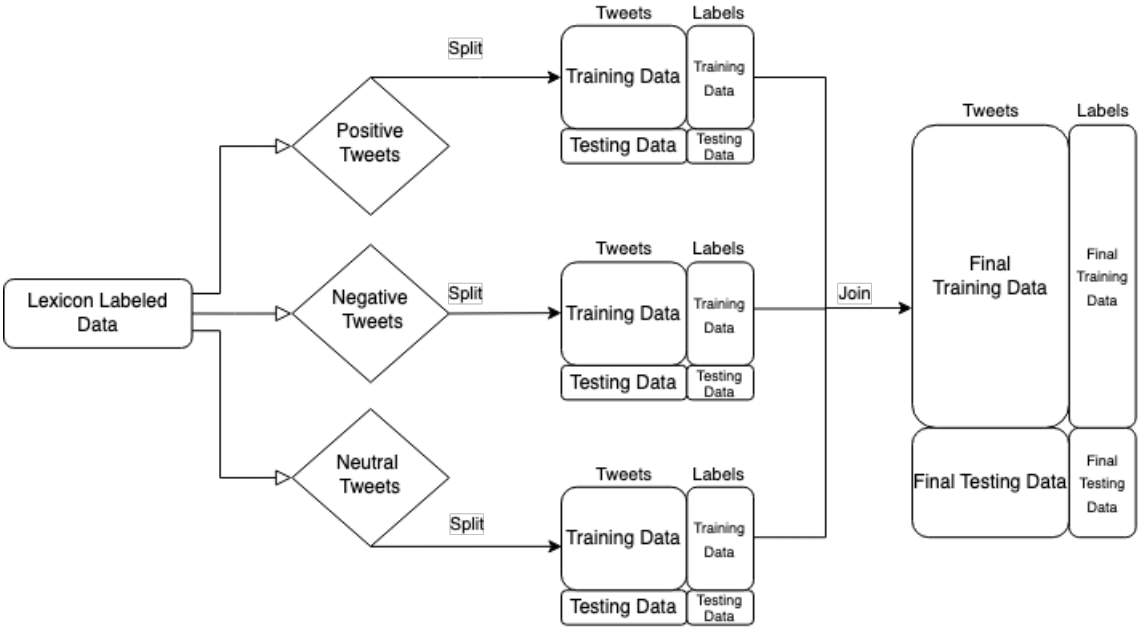


Figure 3.3: Train/Test Tag Split Procedure

3.3.1 TF-IDF

In TF-IDF, or Term Frequency-Inverse Document Frequency, a feature map is created in which keywords with significant value will be weighted heavier than words without. The significance of each word in the data set is found using two main calculations on the word: (1) **Term Frequency**, $TF(t)$ = (Number of times term t appears in a document)/(Total number of terms in the document), and (2) **Inverse Document Frequency**, $IDF(t)$ = log(Total number of documents)/(Number of documents with term t in it).[36][21] The product of $TF(t)$ and $IDF(t)$ gives the TF-IDF weight of that word. TF-IDF was implemented using the scikit-learn Python IF-IDF vectorizer and stored as a sparse matrix.

TF-IDF TruncatedSVD

In order to generate a more robust comparison of models, TF-IDF was studied next to TF-IDF with a Truncated SVD, or singular value decomposition, feature transformation. The singular value decomposition of a matrix S is the factorization of S into the product of three matrices $S = UDV^T$, where the columns of U and V are orthonormal and the matrix D is diagonal with positive real entries.[37] The data sets resulting from the TF-IDF transformation were high in the number of features: 7145 for SentiWordNet and 7079 for VADER (both higher than the number of observations). Dimension reduction is a common practice in modeling as many models may experience performance drop offs as the dimensionality grows, especially as the risk of overfitting grows alongside the VC dimension.[38][16]

The Truncated SVD feature reduction method was chosen because of its ability to handle sparse matrices (unlike traditional PCA) and success rates in accuracy retention. A custom function was implemented that took to the cumulative sum of the variance proportion retained per component, tracking the number of summed components, until a threshold of 0.95 was reached. The final number of features for the VADER analysis and the SentiWordNet analysis were each determined and had values of 1656 and 1668, respectively.

3.3.2 Word2Vec Word Embedding

Word2Vec is a word embedding technique presented in 2014 that represents words as vectors in a high-dimensional space based on their distributional properties using a two-layer neural net.

Transfer learning is a machine learning technique in which the models were previously trained on large amounts of data to be reused for a new yet related model later. This technique saves computational time and improves the accuracy of the word-embedding model.[39] A benefit of word embedding is the ability to extrapolate relationships of similar words, rather than train on direct relationships. Word2Vec is lauded in the NLP field for its ability to capture multiple degrees of similarities[40] Two pre-trained Word2Vec models were used in this analysis: GloVe 200 and Google News 300.

GloVe 200 (Twitter)

The GloVe 200 model is an unsupervised learning algorithm for obtaining vector representations for words in 200 dimensions. Training is performed on aggregated global co-occurrence statistics of word to word pairs from a Twitter corpus, and the resulting representations show the underlying linear substructures of the word vector space.[41] The GloVe 200 tweet vectors were created by finding the word vectors for each token, averaging the value of the word vectors in each tweet, then scaling the results.

Google News 300

The Google News 300 model is another unsupervised learning algorithm that was trained on a large corpus of news articles and commonly used in natural language processing tasks, showing excellent performance in sentiment analysis tasks. The Google 300 tweet vectors were created using the same process as for GloVe 200.

4

Models

Both classification models, Linear SVM and Random Forest, were implemented in python using the scikit learning toolkit.[42] The choice of these models, as elaborated on in chapter 1, was due to their promising applications in the smaller feature space and observation count, high accuracy and popularity in text classification, and model stability.

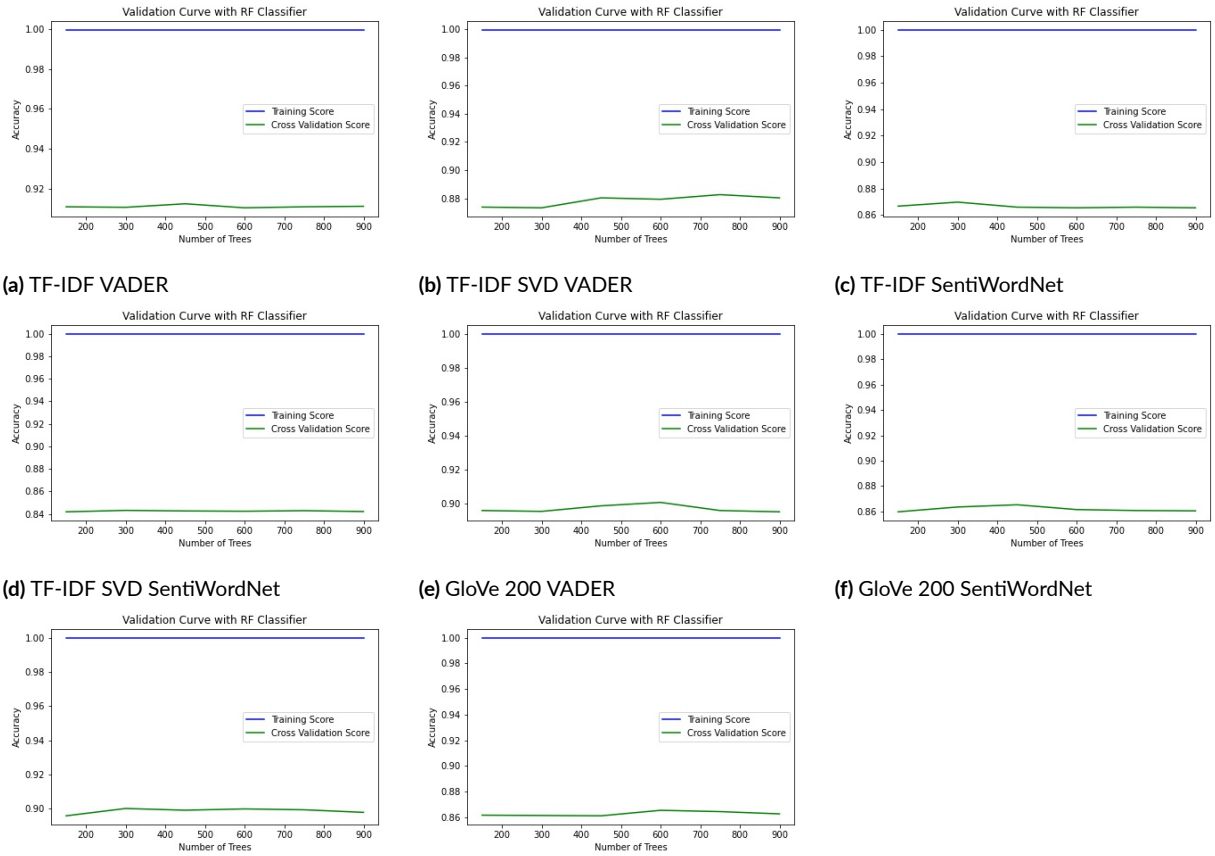
4.0.1 Linear SVM

Linear SVM is a linear classification algorithm that attempts to find the optimal hyperplane to separate positive, negative, and neutral tweets Although normally a binary classifier, Linear SVM can handle more than two classifications in scikit-learn by employing a one-vs-rest method.[42] Setting the 'dual' parameter to 'False' allows data with a larger number of features than observations to be passed modeled, which was necessary for TF-IDF with no dimension reduction. Linear SVM has shown to outperform Naive Bayes, another popular algorithm, in sentiment classification tasks.[15] It is also known to preform better than Logistic Regression, the probabilistic approach as opposed to the statistical approach, on small and complex data sets.[43]

In this study, adjustments to each of the sixteen total SVM models were attempted to optimize parameter selection, however, in all cases, the default parameters yielded the best results.

4.0.2 Random Forest

Random Forest is an ensemble learning algorithm that uses multiple decision trees to improve classification performance. Studies have shown that Random Forest outperforms NB across a variety of data types and applications, partly due to the stability of the algorithm compared to NB.[13][14] The main parameter that affects the performance of a RF model is the number of trees in the forest. The generalization error for a random forest converges asymptotically to a limit as the number of trees in the forest becomes relatively large.[44]



(a) TF-IDF VADER **(b)** TF-IDF SVD VADER **(c)** TF-IDF SentiWordNet

(d) TF-IDF SVD SentiWordNet **(e)** GloVe 200 VADER **(f)** GloVe 200 SentiWordNet

(g) Google 300 VADER **(h)** Google 300 SentiWordNet

Figure 4.1: Validation Curves for Random Forest Model Parameter Tuning

To check if parameter tuning would increase the accuracy of any of the eight models, validation curves were constructed for each unique data set (Figure 4.1). The cross-validation value was set to five randomized validation sets by default. For each of the eight models, the number of trees was set to values from [150, 300, 450, 600, 750, 900], and the default value for 100 was also attempted. All hybrid models showed optimal performance at the default value of 100 trees, except for TF-IDF VADER, TF-IDF with SVD VADER, and Google 300VADER, which were optimized at number of trees: 450 trees, 750 trees, and 300 trees, respectively.

5

Results

The data collected in this analysis was uneven in sentiment classification for both SentiWordNet and VADER. Rather than correcting classification weights to offset this imbalance, the analysis includes data showing the resulting effects, with recommendations on which model performs best given asymmetry. The purpose of omitting re-distribution of classification weights is to simulate real data practices and test the capacity of the different models to account for and adjust to disproportionate data.

As the data is not well balanced, i.e. there are more positive tags than neutral or negative, accuracy is not a good measurement of the models' performances. For each of the sixteen total models, precision, recall, and F-1 scores were computed and stored for (1) the weighted average of the classifications, (2) the positive classifications, (3) the negative classifications, and (4) the neutral classifications. Precision is the measurement of false positives and is calculated for each label as $P = TP/(TP + FP)$, where TP is the number of 'true positives' and FP is the number of 'false positives'. Recall is the measurement of false negatives and is calculated using $R = TP/(TP + FN)$, where FN is the number of false negatives. Both precision and recall fall within the range of $[0, 1]$, with a higher score indicating better model performance. The $F1$ score measures the models' overall performance, balancing between and aggregated from the precision and recall scores; the formula for the $F1$ score is $F1 = \frac{P \cdot R}{P+R}$.

5.1 Weighted Results

5.1.1 SentiWordNet

The SentiWordNet weighted results show promising patterns for identifying an optimized hybrid model. The best performing hybrid model in terms of the $F1$ score was the TF-IDF transformation combined with the SVM ML algorithm, receiving a score of 0.9153 (Figure 5.1). The other SentiWordNet hybrid models with over 90% $F1$ scores were all implemented with the Random Forest ML model, and transformed with the TF-IDF mapping ($F1 = 0.9124$),

the Google 300 embedding ($F1 = 0.9013$), and the GloVe 200 embedding ($F1 = 0.9001$). The TF-IDF transformation with Singular Value Decomposition (SVD) for both ML models recorded $F1$ scores within 0.03 of the standard TF-IDF performances. This suggests that there is a slight performance trade off for easier and quicker data processing which should be considered in future applications. However, if concerned about the dimensionality of TF-IDF transformations because of complexity or storage, one might consider using a different hybrid model, such as RF with word embedding as seen above. The worst performing hybrid model implemented the SVM algorithm and GloVe 200 embedding ($F1 = 0.6832$). The pattern of SVM performing poorly with word embedding will be elaborated on throughout the rest of this section.

These weighted results suggest that when using the SentiWordNet polarity lexicon, the standard TF-IDF transformation with no dimension reduction outperformed the word-embedding transformations. The best performance was seen when standard TF-IDF was used in conjunction with Support Vector Machines, however, the next set of highest $F1$ scores all were attributed to the Random Forest model. The transformations performed better with RF except for TF-IDF SVD and TF-IDF, for which SVM was the better model. There were no major discrepancies between the precision and recall scores for each model, although all models except the two worst performing ones (SVM with both word-embeddings) had a slightly lower recall score. This effect, likely due to the uneven data distribution, will be investigated more closely under the positive, negative, and neutral score reports but shows no obvious major flaws

Model	Transformation	Lexicon	Precision	Recall	F1-Score
RandomForest	GloVe 200	SentiWordNet	0.9117	0.9026	0.9001
RandomForest	Google 300	SentiWordNet	0.9103	0.9042	0.9013
RandomForest	TF-IDF	SentiWordNet	0.9204	0.9147	0.9124
SVM	GloVe 200	SentiWordNet	0.6846	0.6974	0.6832
SVM	Google 300	SentiWordNet	0.7192	0.7291	0.7201
SVM	TF-IDF	SentiWordNet	0.9166	0.9162	0.9153
RandomForest	TF-IDF SVD	SentiWordNet	0.9044	0.8875	0.8837
SVM	TF-IDF SVD	SentiWordNet	0.9002	0.8989	0.8968

Figure 5.1: Weighted Results for SentiWordNet Hybrid Models

in procedure.

5.1.2 VADER

The results for the VADER labeled data showed similar trends with a few key differences. The best performing hybrid model in terms of the *F*1 score was the TF-IDF transformation combined with the RF ML algorithm (0.9318). Note that the SentiWordNet labeled data performed best with SVM while the VADER labeled data performed best with RF (Figure 5.2). The next best performing hybrid models on VADER labeled data were RF with GloVe 200 (0.9286), TF-IDF with SVM (0.9265), and Google 300 with RF (0.9180). For all transformations, RF resulted in higher *F*1 scores than the same transformations with SVM. Most VADER hybrid models received an *F*1 score of over 90%, except for the worst performing models. The worst performing models, as with SentiWordNet, used the word-embedding transformations with the SVM algorithm (0.7834 for Google 300 and 0.7345 for GloVe 200). The TF-IDF transformation with SVD for both ML models again recorded *F*1 scores within 0.03 of the

Model	Transformation	Lexicon	Precision	Recall	F1-Score
RandomForest	GloVe 200	VADER	0.9310	0.9290	0.9286
RandomForest	Google 300	VADER	0.9216	0.9184	0.9180
RandomForest	TF-IDF	VADER	0.9332	0.9320	0.9318
SVM	GloVe 200	VADER	0.7345	0.7349	0.7319
SVM	Google 300	VADER	0.7834	0.7840	0.7831
SVM	TF-IDF	VADER	0.9272	0.9267	0.9265
RandomForest	TF-IDF SVD	VADER	0.9155	0.9116	0.9112
SVM	TF-IDF SVD	VADER	0.9049	0.9033	0.9027

Figure 5.2: Weighted Results for VADER Hybrid Models

standard TF-IDF performances, suggesting the performance trade off for easier/quicker use. These results suggest overall that when using the VADER polarity lexicon, the RF algorithm led to better results and the TF-IDF transformation comprised 2 of the 3 top scoring hybrid models. The weighted average precision and recall scores for each model showed a similar pattern to those in the SentiWordNet, in which all models except the two worst performing ones had a recall score slightly lower than the precision score.

5.2 Positive Results

Although the weighted average scores give an idea of the overall performance of the models, more insight can be gained by looking at scores for each type of classification, particularly with asymmetrical data.

5.2.1 SentiWordNet

Recall that the result of SentiWordNet labeling left a distribution of tags that were positive (0.58), negative (0.25), and neutral (0.17)(Figure 3.2a). The $F1$ scores are once again the most important measure due to this asymmetry, however, precision and accuracy will be evaluated in comparison with ech other.

For positively labeled tweets, TF-IDF with the SVM ML algorithm resulted in the highest $F1$ score (0.9392), with TF-IDF with the RF ML algorithm closely following (0.9385) (Figure 5.3). The positively labeled $F1$ scores begin to deviate from the weighted average ranking slightly, as TF-IDF with the SVD transformation boosted to third best (0.9307). Random Forest was present in the the majority of high-achieving models but not significantly enough to conclude it will perform better for labeling positive tweets than SVM. The set of scores for positive results are generally higher than the weighted results indicating overall better performance in labeling positive tweets. Additionally, the values for positive recall are all higher than values for positive precision which is consistent with the proportions of tag distribution (majority positive tweets in training data $\rightarrow$ more positive classifications made by model $\rightarrow$ more false positives than false negatives).

5.2.2 VADER

The result of VADER labeling left a distribution of tags that were neutral (0.51), positive (0.42), and negative (0.07)(Figure 3.2b). The $F1$ scores in positive tagging aligned with the weighted

Model	Transformation	Lexicon	Precision	Recall	F1-Score
SVM	TF-IDF	SentiWordNet	0.9134	0.9665	0.9392
SVM	GloVe 200	SentiWordNet	0.7368	0.8671	0.7967
SVM	Google 300	SentiWordNet	0.7768	0.8710	0.8212
RandomForest	TF-IDF	SentiWordNet	0.8882	0.9948	0.9385
RandomForest	GloVe 200	SentiWordNet	0.8668	0.9910	0.9247
RandomForest	Google 300	SentiWordNet	0.8767	0.9910	0.9303
SVM	TF-IDF SVD	SentiWordNet	0.8942	0.9703	0.9307
RandomForest	TF-IDF SVD	SentiWordNet	0.8402	0.9974	0.9121

Figure 5.3: Positive Results for SenitWordNet Hybrid Models

Model	Transformation	Lexicon	Precision	Recall	F1-Score
SVM	TF-IDF	VADER	0.9201	0.9384	0.9291
SVM	GloVe 200	VADER	0.7505	0.7192	0.7345
SVM	Google 300	VADER	0.7923	0.7808	0.7865
RandomForest	TF-IDF	VADER	0.9407	0.9203	0.9304
RandomForest	GloVe 200	VADER	0.9474	0.9130	0.9299
RandomForest	Google 300	VADER	0.9479	0.8895	0.9178
SVM	TF-IDF SVD	VADER	0.8938	0.9149	0.9042
RandomForest	TF-IDF SVD	VADER	0.9452	0.8750	0.9087

Figure 5.4: Positive Results for VADER Hybrid Models

results.(Figure 5.4). However, the recall score for the hybrid models was not consistently higher than the precision scores, unlike with SentiWordNet. This is expected as the tag distributions varied across the two models and positive tweets were not the majority proportion. Interestingly, the TF-IDF, both standard and with SVD, when combined with the SVM model produced higher recall scores than precision. This relationship direction is inverse of all other models in the VADER positive results set.

Model	Transformation	Lexicon	Precision	Recall	F1-Score
SVM	TF-IDF	SentiWordNet	0.9109	0.8519	0.8804
SVM	GloVe 200	SentiWordNet	0.5984	0.4691	0.5260
SVM	Google 300	SentiWordNet	0.6399	0.5648	0.6000
RandomForest	TF-IDF	SentiWordNet	0.9767	0.7778	0.8660
RandomForest	GloVe 200	SentiWordNet	0.9650	0.7654	0.8537
RandomForest	Google 300	SentiWordNet	0.9565	0.7469	0.8388
SVM	TF-IDF SVD	SentiWordNet	0.8867	0.8210	0.8526
RandomForest	TF-IDF SVD	SentiWordNet	0.9914	0.7099	0.8273

Figure 5.5: Negative Results for SenitWordNet Hybrid Models

5.3 Negative Results

5.3.1 SentiWordNet

SentiWordNet produced negative tags on 25% of the data, the second leading tag behind positive. Only the two highest and the two lowest performing hybrid models aligned in ranking for negative scores and weighted average scores (Figure 5.5). For negative labeling with SentiWordNet, SVM performed better than RF for all TF-IDF variations, but RF outperformed SVM in all word-embedding variations. This distinction could help other researchers navigate proper hybrid applications in nonuniform text data.

5.3.2 VADER

The VADER tagged tweets only were labeled as negative 7% of the time, indicating significant dis-proportionality. This feature manifests in the scoring results of the negative label, as values

Model	Transformation	Lexicon	Precision	Recall	F1-Score
SVM	TF-IDF	VADER	0.9620	0.8172	0.8837
SVM	GloVe 200	VADER	0.6949	0.4409	0.5395
SVM	Google 300	VADER	0.7273	0.6022	0.6588
RandomForest	TF-IDF	VADER	1.0000	0.8280	0.9059
RandomForest	GloVe 200	VADER	1.0000	0.7849	0.8795
RandomForest	Google 300	VADER	1.0000	0.7849	0.8795
SVM	TF-IDF SVD	VADER	0.9855	0.7312	0.8395
RandomForest	TF-IDF SVD	VADER	1.0000	0.7849	0.8795

Figure 5.6: Negative Results for VADER Hybrid Models

for precision hit 1.0 for all models using RF (Figure 5.6). This means that for RF, all instances of negative tweets were correctly classified. The data classification dis-proportionality also manifests in the comparison of precision to recall; all the precision scores were higher than the recall scores, indicating the models tended to over-assign the negative label. The precision scores for SVM were less uniform, ranging from 0.6949 for GloVe 200 to 0.9855 for TF-IDF with SVD. Generally, SVM and word embedding transformation technique still consistently score lowest. Random Forest algorithm returned the same negative labeling scores for all feature transformations, other than standard TF-IDF, which can happen in cases of label dis-proportionality. TF-IDF with RF had the highest recall, precision, and $F1$ scores (0.8280, 1.0, 0.9059).

Model	Transformation	Lexicon	Precision	Recall	F1-Score
SVM	TF-IDF	SentiWordNet	0.9356	0.8363	0.8832
SVM	GloVe 200	SentiWordNet	0.6289	0.4425	0.5195
SVM	Google 300	SentiWordNet	0.6353	0.4779	0.5455
RandomForest	TF-IDF	SentiWordNet	0.9497	0.8363	0.8894
RandomForest	GloVe 200	SentiWordNet	0.9890	0.7965	0.8824
RandomForest	Google 300	SentiWordNet	0.9592	0.8319	0.8910
SVM	TF-IDF SVD	SentiWordNet	0.9402	0.7655	0.8439
RandomForest	TF-IDF SVD	SentiWordNet	1.0000	0.7655	0.8672

Figure 5.7: Neutral Results for SenitWordNet Hybrid Models

5.4 Neutral Results

5.4.1 SentiWordNet

The neutral classification was expected to yield the weakest results for SentiWordNet, as neutral tweets only comprised 17% of the data. Similar to the smallest tag category in VADER, the precision scores for SentiWordNet neutral tweets were consistently higher than the recall scores. Again, this implies that all models tending to over-classify data as neutral. The ranking of the models are relatively inconsistent when compared against the weighted average results. The only pattern that has been unchanged in almost every collection of scores for every model is the SVM ML algorithm with word embeddings preforming significantly worse than the other hybrid options.

Model	Transformation	Lexicon	Precision	Recall	F1-Score
SVM	TF-IDF	VADER	0.9282	0.9323	0.9302
SVM	GloVe 200	VADER	0.7269	0.7879	0.7562
SVM	Google 300	VADER	0.7838	0.8115	0.7974
RandomForest	TF-IDF	VADER	0.9180	0.9558	0.9365
RandomForest	GloVe 200	VADER	0.9082	0.9617	0.9342
RandomForest	Google 300	VADER	0.8895	0.9602	0.9235
SVM	TF-IDF SVD	VADER	0.9029	0.9175	0.9102
RandomForest	TF-IDF SVD	VADER	0.8797	0.9588	0.9175

Figure 5.8: Neutral Results for VADER Hybrid Models

5.4.2 VADER

Neutral tweets comprised 51% of the all VADER labelled data. Unsurprisingly, *F*1 score ranking of the neutral scores aligns with the weighted average ranking for VADER. The recall scores for each model are higher than the precision scores, indicating that the neutral classifications made by each model are accurate, but the models are less successful classifying every neutral tweet.

6

Conclusion

Overall, this analysis aimed to compare different hybrid sentiment analysis models for their relative performance on Twitter data for two polarity lexicons: SentiWordNet and VADER. Each lexicon analysis included precision, recall, and $F1$ scores from applying TF-IDF, TF-IDF with PCA, GloVe200, and Google300 with Linear SVM and Random Forest parameterized models. This analysis uncovered a few interesting features that arise from combining different techniques.

Before any feature transformations or ML modeling, it was clear that different polarity lexicons can produce very different distributions of labels on the same data set. SentiWordNet and VADER are two of the most common and well-respected opinion dictionaries, yet produced two very different classification results. VADER has a higher threshold for neutrality, for which a body of text must surpass a certain intensity of sentiment. SentiWordNet, on the other hand, tends to assign any level of emotion or sentiment as on the positive/negative scale, resulting in much fewer neutral classifications.

There were plenty of interesting observations in chapter 5 regarding nonuniform data distribution. For instance, negative labeling with the SentiWordNet data set (which had a low proportion of negative labels), SVM performed better than RF for all TF-IDF variations, but RF outperformed SVM in all word-embedding variations. More generally, the hybrid models that used the SVM with word embeddings consistently performed worse than all other procedures. In fact, Random Forests performed exceptionally well for both lexicons even though it is much less cited in sentiment analysis literature; only SVM with TF-IDF transformations were competitive with RF.

6.1 Limitations and Future Work

A major limitation of this report was the size and type of the data. In the future, this work could be expanded to include text data from a collection of sources and of variable size. It would be interesting to study how the dimensionality and number of observations affects the

effectiveness of the hybrid models. Additionally, larger quantities of data may offset the effects of non-uniformity and produce different observations. Although this report was limited to local comparison of hybrid procedures, more dictionaries, transformations, and models could fairly simply be incorporated for a more robust analysis. In particular, Naive Bayes and Logistic Regression could potentially out-preform the models in this study with certain hybrid conditions that have been understudied.

A future expansion on this analysis that is particularly attractive is enhancing the pre-process to account not only for misspellings using a character difference model, but by also incorporating a measure for phonetic spellings. For example, the 'DIFFERENCE' function in SQL can locate words that are phonetically identical within a user chosen degree (0-4) but potentially spelled very differently. In the current framework for spelling corrections, these phonetic similarities are missed.

Another dimension of this analysis that could be explored is introducing different labeling thresholds for the sentiment lexicons. Although this report followed conventional standard, setting a higher threshold for sentiment in SentiWordNet, VADER, or other dictionaries could provide interesting results, particularly as data from human-labeled intensity scales (i.e. the Likert Scale) tend to be concentrated *not* at the polar extremes.

There are countless ways to expand on this analysis, even in aspects of parameter tuning such as bi-gram usage in word embedding or different scaling and standardizing methods.

References

[1] B. Ohana, "Sentiment classification of reviews using sentiwordnet," 2009. [Online]. Available: http://arrow.dit.ie/ittpapnin/13/

[2] N. Kelber and T. Lawless, "Sentiment analysis with vader." [Online]. Available: https://ithaka.github.io/tdm-notebooks/sentiment-analysis-with-vader.html#:~:text=There%20are%20over%207500%20tokens,and%20sensitive%20relationships%20between%20terms.

[3] L. Bing, *Synthesis Lectures on Human Language Technologies*, 2012, vol. Sentiment Analysis and Opinion Mining.

[4] M. Liaqat, M. Awais Hassan, M. Shoaib, S. Khurshid, and M. Shamseldin, "Sentiment analysis techniques, challenges, and opportunities: Urdu language-based analytical study," pp. 79–86, Aug. 2022. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9454799/#ref-4

[5] M. Devika, C. Sunitha, and A. Ganesh, "Sentiment analysis: A comparative study on different approaches," *Procedia Computer Science*, vol. 87, pp. 44–49, 2016, fourth

International Conference on Recent Trends in Computer Science Engineering (ICRTCSE 2016). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S187705091630463X

[6] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede, "Lexicon-Based Methods for Sentiment Analysis," *Computational Linguistics*, vol. 37, no. 2, pp. 267–307, 06 2011. [Online]. Available: https://doi.org/10.1162/COLI_a_00049

[7] M. Devaraj, R. Piryani, and V. K. Singh, "Lexicon ensemble and lexicon pooling for sentiment polarity detection," *IETE Technical Review*, vol. 33, no. 3, pp. 332–340, 2016. [Online]. Available: https://doi.org/10.1080/02564602.2015.1073572

[8] A. Esuli, "Sentiwordnet," https://github.com/aesuli/SentiWordNet, 2022.

[9] C. Hutto and E. Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 8, no. 1, p. 216–225, 2014.

[10] C. Dhaoui, C. Webster, and L. Tan, "Social media sentiment analysis: lexicon versus machine learning," *Journal of Consumer Marketing*, vol. 34, no. 6, pp. 480–488, 2017.

[11] dishashree26, "Transfer learning and the art of using pre-trained models in deep learning," Apr 2023. [Online]. Available: https://www.analyticsvidhya.com/blog/2017/06/transfer-learning-the-art-of-fine-tuning-a-pre-trained-model/?utm_source=blog&utm_medium=pretrained-word-embeddings-nlp

[12] A. Pai, "An essential guide to pretrained word embeddings for nlp practitioners," Jun 2022. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/03/pretrained-word-embeddings-nlp/

[13] A. Singh, M. N. Halgamuge, and R. Lakshmiganthan, "Impact of different data types on classifier performance of random forest, naïve bayes, and k-nearest neighbors algorithms," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 12, 2017. [Online]. Available: http://dx.doi.org/10.14569/IJACSA.2017.081201

[14] D. Jollyta, G. Gusrianty, and D. Sukrianto, "Analysis of slow moving goods classification technique: Random forest and naïve bayes," *Khazanah Informatika : Jurnal Ilmu Komputer dan Informatika*, vol. 5, no. 2, pp. 134–139, Dec. 2019. [Online]. Available: https://doi.org/10.23917/khif.v5i2.8263

[15] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? sentiment classification using machine learning techniques," in *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*. Association for Computational Linguistics, Jul. 2002, pp. 79–86. [Online]. Available: https://aclanthology.org/W02-1011

[16] D. Effrosynidis, S. Symeonidis, and A. Arampatzis, "A comparison of pre-processing techniques for twitter sentiment analysis," in *Research and Advanced Technology for Digital Libraries*, J. Kamps, G. Tsakonas, Y. Manolopoulos, L. Iliadis, and I. Karydis, Eds. Cham: Springer International Publishing, 2017, pp. 394–406.

[17] F. Sommar and M. Wielondek, "Combining lexicon- and learning-based approaches for improved performance and convenience in sentiment classification," 2015.

[18] DataDome, "Headless chrome: What it is amp; how to detect it," Jun 2022. [Online]. Available: https://datadome.co/learning-center/what-is-headless-chrome-how-to-detect-it/

[19] I. Hemalatha, G. S. Varma, and A. Govardhan, "Preprocessing the informal text for efficient sentiment analysis," *International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)*, vol. 1, no. 2, pp. 58–61, 2012.

[20] M. Lenzerini, "Managing data through the lens of an ontology," *AI Magazine*, vol. 39, no. 2, pp. 65–74, Jul. 2018. [Online]. Available: https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/2802

[21] E. Haddi, X. Liu, and Y. Shi, "The role of text pre-processing in sentiment analysis," *Procedia Computer Science*, vol. 17, pp. 26–32, 2013, first International Conference on Information Technology and Quantitative Management. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050913001385

[22] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*. O'Reilly Media, 2009.

[23] "Cleantext." [Online]. Available: https://pypi.org/project/cleantext/

[24] "Regular expressions." [Online]. Available: https://docs.python.org/3/howto/regex.html

[25] U. Ambudkar, "Negation handling," https://github.com/UtkarshRedd/Negation*handling*, 2022.

[26] "Spellchecker." [Online]. Available: https://pypi.org/project/pyspellchecker/

[27] T. Mullen and R. Malouf, "A preliminary investigation into sentiment analysis of informal political discourse." 01 2006, pp. 159–162.

[28] L. Derczynski, A. Ritter, S. Clark, and K. Bontcheva, "Twitter part-of-speech tagging for all: Overcoming sparse and noisy data," in *Proceedings of the International Conference on Recent Advances in Natural Language Processing*. Association for Computational Linguistics, 2013.

[29] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proceedings of the Eighteenth International Conference on Machine Learning*, ser. ICML '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, p. 282–289.

[30] T. T. Goh, N. A. A. Jamaludin, H. Mohamed, M. N. Ismail, and H. S. Chua, "A comparative study on part-of-speech taggers' performance on examination questions classification according to bloom's taxonomy," *Journal of Physics: Conference Series*, vol. 2224, no. 1, p. 012001, apr 2022. [Online]. Available: https://dx.doi.org/10.1088/1742-6596/2224/1/012001

[31] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz, "Building a large annotated corpus of English: The Penn Treebank," *Computational Linguistics*, vol. 19, no. 2, pp. 313–330, 1993. [Online]. Available: https://aclanthology.org/J93-2004

[32] M. F. Porter, "An algorithm for suffix stripping," vol. 14, no. 3, pp. 130–137, 2023/04/22 1980. [Online]. Available: https://doi.org/10.1108/eb046814

[33] Rianto, A. B. Mutiara, E. P. Wibowo, and P. I. Santosa, "Improving the accuracy of text classification using stemming method, a case of non-formal indonesian conversation," *Journal of Big Data*, vol. 8, no. 1, Jan. 2021. [Online]. Available: https://doi.org/10.1186/s40537-021-00413-1

[34] M. Husnain, M. M. S. Missen, N. Akhtar, M. Coustaty, S. Mumtaz, and V. B. S. Prasath, "A systematic study on the role of sentiwordnet in opinion mining," *Frontiers of Computer Science*, vol. 15, no. 4, p. 154614, 2021. [Online]. Available: https://doi.org/10.1007/s11704-019-9094-0

[35] A. Esuli and F. Sebastiani, "Sentiwordnet: A publicly available lexical resource for opinion mining," in *In Proceedings of the 5th Conference on Language Resources and Evaluation (LREC'06*, 2006, pp. 417–422. [Online]. Available: SENTIWORDNET: Apubliclyavailablelexicalresourceforopinionmining

[36] "Tf-idf : A single-page tutorial - information retrieval and text mining." [Online]. Available: https://tfidf.com/

[37] "Singular value decomposition." [Online]. Available: https://www.cs.cmu.edu/~venkatg/teaching/CStheory-infoage/book-chapter-4.pdf

[38] J. Howbert. [Online]. Available: http://courses.washington.edu/css581/lecture_slides/17_dimensionality_reduction.pdf

[39] "Transfer learning." [Online]. Available: https://www.intel.com/content/www/us/en/developer/topic-technology/artificial-intelligence/training/transfer-learning.html#gs.w5kxpt

[40] J. Mu, "Word 2 vec : What and why," 2016.

[41] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[42] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[43] A. Saini, "Support vector machine(svm): A complete guide," Nov 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/

[44] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324

Acknowledgments

I would like to express my deepest appreciation to my advisor, Mike Izbicki, for his invaluable guidance, encouragement, and support throughout my academic journey. His expertise, patience, and dedication have been instrumental in shaping my education at CMC. I would also like to thank to my parents for their unconditional love, unwavering support, and constant encouragement. Their belief in me has been a constant source of strength and inspiration. I would like to acknowledge the Dean of Students, Dianna Graves, for her encouragement and advice which have been critical in my success. Finally, I would like to thank the numerous friends and colleagues who have supported and encouraged me throughout my academic journey. Their assistance and encouragement have been invaluable, and I am forever grateful for their kindness and support.