12-2019

Sentiment Analysis, Quantification, and Shift Detection

Kevin Labille
University of Arkansas, Fayetteville

Sentiment Analysis, Quantification, and Shift Detection

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in Computer Science

by

Kevin Labille
IUT Dijon, University of Burgundy
Bachelor of Science in Computer Science, 2010
ESIREM, University of Burgundy
Master of Science in Computer Science, 2013

December 2019
University of Arkansas

This dissertation is approved for recommendation to the Graduate Council

---

Susan Gauch, Ph.D.
Dissertation Director

Khoa Luu, Ph.D.
Committee member

Samantha Robinson, Ph.D.
Committee member

Xintao Wu, Ph.D.
Committee member

Annibal Sodero, Ph.D.
Ex-officio Member

ABSTRACT

This dissertation focuses on event detection within streams of Tweets based on sentiment quantification. Sentiment quantification extends sentiment analysis, the analysis of the sentiment of individual documents, to analyze the sentiment of an aggregated collection of documents. Although the former has been widely researched, the latter has drawn less attention but offers greater potential to enhance current business intelligence systems. Indeed, knowing the proportion of positive and negative Tweets is much more valuable than knowing which individual Tweets are positive or negative. We also extend our sentiment quantification research to analyze the evolution of sentiment over time to automatically detect a shift in sentiment with respect to a topic or entity.

We introduce a probabilistic approach to create a paired sentiment lexicon that models the positivity and the negativity of words separately. We show that such a lexicon can be used to more accurately predict the sentiment features for a Tweet than univalued lexicons. In addition, we show that employing these features with a multivariate Support Vector Machine (SVM) that optimizes the Hellinger Distance improves sentiment quantification accuracy versus other distance metrics. Furthermore, we introduce a mean of representing sentiment over time through sentiment signals built from the aforementioned sentiment quantifier and show that sentiment shift can be detected using geometric change-point detection algorithms. Finally, our evaluation shows that, of the methods implemented, a two-dimensional Euclidean distance measure, analyzed using the first and second order statistical moments, was the most accurate in detecting sentiment shift.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest gratitude to my adviser Dr. Susan Gauch, for her invaluable expertise and guidance provided throughout my doctorate, but also for her immeasurable support whether on the professional or personal level, and without whom this work would have never seen the light.

I would also like to express my sincere and genuine thankfulness to my family who provided constant support throughout my graduate studies: My mom Pascale, my aunt Isabelle, my brothers Jess and Dylan, my sister Sandy, and my step father Emile.

I want to express my deepest and wholehearted thanks to Stephane, Fabrice, Vincent, Adil, and Bri, for their continuous encouragement and motivation.

Last but not least, I would like to thank my friends for providing insight and support throughout my graduate studies: Alex, Nellis, Tobey, Segolene, Mo, Loic, Steve, Danilo, Brice, Marius, Jean Pierre, Israel, Elvis, Manzi, Franck, Michael, Dr. Matt Patitz, Dr. John Gauch, Festus, George, Jhonnie and Bob, Barrett, Honore, and Anne-Lise.

DEDICATION

This dissertation is dedicated to Stephane Makombe who always believed in me and never gave up on me.

EPIGRAPH

It is by logic that we prove, but by intuition that we discover. To know how to criticize is good, to know how to create is better.

Henri Poincaré

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 Introduction

The rapid growth of online commerce, or e-commerce, has allowed online retailers to make it possible for customers to share their opinions about products and items. One issue is that it is hard to define what an opinion is. The difference between an opinion or a fact is very small and people will often disagree on which is which [2, 3]. Despite this, opinions can be useful not only to online e-commerce but also in government intelligence, business intelligence, and other online services [4] that benefit from summarizing and analyzing collective viewpoints.

The number of online reviews has increased tremendously over the years, and it is now possible to read the opinions of thousands of people all over the Internet on movies, restaurants, hotels, books, products, and professionals. The large amount of information available online today allows researchers to study how individuals express opinions and to mine the collections of opinions to identify trends and consensus. Two new research area have arisen from this phenomenon: sentiment analysis and sentiment quantification. Thus, sentiment analysis is the computational analysis of opinions in text, which aims to identify the semantic orientation, or polarity, of such textual data. In contrast, sentiment quantification aims to estimate the proportion of document that belong to each polarity classes.

Sentiment analysis has therefore found its place in various areas, for instance, in politics. Indeed, Wang et al. [5] applied real-time sentiment analysis to Twitter data to analyze public sentiment toward presidential candidates in the U.S elections of 2012. The most prominent and perhaps the most prevalent utilization of sentiment analysis, however, is in business intelligence, since customer's feedback directly reflects their opinion about a

product or service. Sentiment analysis can be used as a concept testing tool when a new product, campaign, or logo is launched. It can be used to improve a company's own performances by analyzing competitor's sentiment data and gain competitive advantage. It can also be applied to gain insight from the opinions of customers to diagnose possible problems and make improvement. Additionally, sentiment analysis can be used to track customer sentiment over time. Although a lot of work has been conducted in sentiment analysis, we believe that the last application aforementioned can be further exploited to open new horizons for businesses and for sentiment analysis research.

Today, although customer's opinions are being tracked over time, they are most of the time saved to be analyzed offline later. Sentiment analysis techniques can then be applied to automatically extract the sentiment of each opinion. By looking at the graph of the number of positive opinions and negative opinions over time, one could potentially identify if and when a shift of sentiment happened. To our knowledge, this task is generally operated manually by a human which cost time and resources. Besides, this approach does not allow one to analyze real-time data, which is an issue when dealing with fast online data sources such as Twitter. In addition, it could have economic consequences in the business world, take for instance the following scenario. A major company releases a new advertisement that customers find scandalous and therefore negatively comment upon on Twitter. If no automatic sentiment shift detection system is in place, the company's reputation could be heavily hurt and the company's sales could also be affected. A fast automatic sentiment shift detection system could identify such phenomenon and trigger an adequate response which would prevent the catastrophe.

One can clearly see that having the ability to promptly detect when customers report negative experiences, poor service, or other complaints is of high importance from a business per-

spective. Similarly, automatically detecting and identifying common causes for dissatisfied customers is crucial to making immediate corrections to improve the customer experience. To the best of our knowledge there are currently no research that focuses on automatically detecting a shift of sentiment. In this document we attempt to provide a solution to the problem of automatically detecting a shift of sentiment with respect to a topic or an entity.

To tackle this problem we will divide our work into 3 goals, each of which will aim at providing an answer to the following questions:

1. How to represent a Tweet in a way such that (i) its representation embeds the sentiment it reflects, (ii) the computer understands, and (iii) computations can be performed by the computer

2. How could we accurately quantify the prevalence of the positive class and the negative class of a collection of tweet given a representation that satisfies (1)

3. How to identify a shift of sentiment after that inquiry (2) was satisfied

To achieve goal 1 we will use a probabilistic approach to create a sentiment lexicon from a collection of Tweets. The sentiment lexicon can then be used to represent a Tweet in the Vector Space Model through a feature vector. We can use the resulting feature vector in goal 2 to perform sentiment quantification in the Vector Space Model through a SVM machine. Finally, a sentiment signal can be built using the quantification resulting from goal 2, and change-point detection methods can be employed to automatically detect a shift.

2 Literature Review

2.1 Sentiment Analysis

2.1.1 Sentiment Analysis

Mining and summarizing online reviews to determine sentiment orientation has become a popular research topic often broken into two main areas of research: opinion summarization and opinion mining. Opinion summarization is the task of identifying and extracting product features from product's reviews in order to summarize them. Hu and Liu proposed [6] a method to find and extract key features and the opinions related to them among several reviews. In contrast, opinion mining consists of analyzing a product's review in order to determine whether or not it reflects a positive or negative sentiment [7]. There are traditionally two ways of doing sentiment analysis, using either supervised learning techniques or unsupervised learning techniques. In the former approach, sentiment classification is often seen as a two-class classification problem and we typically use a naive Bayes classifier or build a Support Vector Machine (SVM) that is trained on a particular dataset[8, 9, 10, 11, 12, 13, 14]. SVM are supervised learning models for classification and regression analysis. Other methods such as hierarchical classification using a Sentiment Ontology Tree [15], have also been used. These approaches generally perform well on the domain for which they are trained. In contrast, unsupervised approaches (also referred to as lexicon-based approaches), consist of computing the semantic orientation of a review from the semantic orientation of each word found in that review. It can be seen as an unsupervised learning method [10, 16, 17, 18, 19, 20]. It is not uncommon to have reviews that are

rated within a range, e.g., from 1 to 5, to express a degree of positiveness or negativeness. Sentiment rating prediction or rating-inference research focuses on the task of predicting the rating rather than the sentiment orientation. Pang and Lee [21] tackled this problem using an SVM regression approach and a SVM multiclass approach. Goldberg and Zhu [22] implemented a graph-based semi-supervised approach and improved upon the previous work.

2.1.2 Sentiment Analysis in Twitter

With over 500 million tweets shared every day (6,000 tweets per second), Twitter has become the fastest growing source of information. Users generally tweet about their feelings or opinions about what's happening around the world. This makes Twitter a valuable source of data for sentiment analysis. However, tweets differ from regular text in many ways: the length of a tweets is restricted to 280 characters and, because they are often posted from cell-phones, the language used contains many spelling mistakes, abbreviations, and slang words. These characteristics make traditional Natural Language Processing techniques, language models, and traditional sentiment analysis tasks trickier to apply. Despite these challenges, numerous projects have investigated Twitter sentiment classification [23, 24, 25, 26, 27]. One of the pioneer works is that of Go et al. [28] wherein they compared a SVM classifier (with feature vectors composed of unigrams, bigrams, or unigrams+bigrams), a Maximum Entropy (MaxEnt) classifier, and a Naive Bayes classifier. Their results suggest that Part Of Speech (POS) tags are not useful in Twitter sentiment classification. They achieved their best accuracy with the MaxEnt classifier and the lowest accuracy with the Naive Bayes classifier. Mohammad et al. [29] and [30] tackled the same problem through a SVM classifier that uses sentiment lexicons as part of the feature vector. They showed that lexicons-related features were valuable features that improved the accuracy of the SVM classifier by more

than 8.5%. More recently, Tang et al. [31] employed a machine learning approach that uses word embedding in different neural networks to capture the sentiment information of sentences as well as the syntactic contexts of words. Although their approach outperformed Mohammad's approach by 1.85%, the computational power and complexity of their model makes this improvement less compelling in practice.

In our work, we will build on the approach of [32] but extend it to take into consideration the nature of Tweets.

2.2 Sentiment Quantification

2.2.1 Quantification

More recently, a new research focus has emerged from automated classification: quantification. In contrast to classification that aims to estimate the class label of individual instances, the purpose of quantification is to evaluate the population or prevalence of the different classes in the dataset. Although the tasks sound similar, a method with a high accuracy on the individual level can be biased and achieve poor performance when estimating the proportion of the different classes, requiring new techniques. We therefore need to tackle the quantification problem with new techniques. Hopkins and King [33] studied the question using a non-parametric statistical model and applied it to estimate the proportion of politically related blog posts in 7 classes. Results suggest that their approach outperforms a traditional SVM classification approach. Gonzales-Castro [34] et al. focused on quantification for a 2-class problem. They implemented a class distribution estimator (named HDy) that minimizes the Hellinger Distance of the predicted distribution with the

validation distribution. They contrasted HDy with several state-of-the-art class distribution estimators (such as CC, AC, MS, and PP) and concluded that it performs best with every classifier they tested. Baranquero et al [35] focused on optimizing classification as well as quantification through a SVM_{perf} machine, and introduced a metric called Q-measure. They concluded that optimizing both classification and quantification (by the mean of the Q-measure) performs better than optimizing quantification only. Similarly, Esuli and Sebastiani [36] focused on text quantification using multivariate SVM, i.e., SVM_{perf}, that uses the Kullback-Leibler divergence as a loss function (KLD is a measure of the divergence between two probability distributions).They found that SVM(KLD) outperform every other linear SVM approach or other quantification method and is therefore a more appropriate choice for text quantification.

2.2.2 Sentiment Quantification in Twitter

Sentiment quantification is simply quantification techniques applied to sentiment data. There have been several projects that apply sentiment quantification to Twitter data. One of the first works to tackle this problem was published by Gao and Sebastiani [1] wherein the authors apply the $SVM(KLD)$ approach from [36] to Twitter data. They compared $SVM(KLD)$ with an SVM classifier that minimizes the well known Hinge Loss function (HL). The Hinge Loss function is a loss function commonly used for maximum-margin classification problems. They concluded that $SVM(KLD)$ outperforms $SVM(HL)$. In 2016, the high-impact conference "International Workshop on Semantic Evaluation", i.e., SemEval, held a track on sentiment quantification. The 2nd best-performing approach was contributed by Vilares et al. [37] who tackled the problem by training a Convolutional Neural Network (CNN) and using its hidden activation values as features to train a $SVM(KLD)$ classifier.

They compared it to a simple "classify and count" approach that employed a regular SVM and concluded that their approach performed better. However, the overall best approach from SemEval 2016 was by Stojanovski et al. [38]. They used a combination of a Convolutional Neural Network (CNN) and a Gated Neural Network (GNN), which was then fed into a softmax layer. They concluded that the combination of the two neural network is well suited for quantification. Their CNN alone achieved a KLD (note that KLD is the most commonly used metric for evaluating quantification) of 0.053 versus 0.045 for the GNN alone. Since the CNN's performance alone is very close to that of the CNN+GNN, the benefit of adding the GNN to the CNN is marginal. In the 2017 edition of the International Workshop on Semantic Evaluation, Mathieu Cliche [39] achieved first place on the sentiment quantification task. He used a deep-learning method that uses both a Convolutional Neural Network (CNN) and a LSTM (Long Short-Term Memory) neural networks that uses word embedding.

In our work, we will build upon the approach of [36] by comparing various statistical distances optimized through a multivariate SVM.

2.3 Event Detection

2.3.1 Event Detection in Text Stream

Event detection aims to automatically detect and identify novel events from a text stream. The question has been heavily studied in the past with one of the first approach published in 1998 by Yang et al. [40]. Event detection is traditionally divided into two approaches: *document-pivot*, wherein the goal is to cluster documents based on the semantic distance of their textual content, and *feature-pivot*, wherein the goal is to study the distri-

bution and frequency of features such as words. We will focus on work that uses the latter approach. Kleinberg [41] detected events using an infinite-state automaton wherein events are modeled as state transitions. In contrast, Fung et al. [42] focus on detecting events by identifying "bursty" words (that is, a word that shows a spike in its frequency) using their distribution and by then grouping similar bursty features. While the previous work focus on analyzing words on the time domain, He et al. [43] considered the frequency domain and employed the Discrete Fourier Transform (DFT) on word's signals. Indeed, burst in the time domain will produce a spike in the frequency domain. They combined this information with a Gaussian Mixture model to identify and locate events in time. In other work, Chen and Roy [44] focused on identifying events in Flickr using images' tags. They employed wavelet transformations which are localized in both time and frequency domain, allowing them to have both temporal and locational distributions of features.

2.3.2 Event Detection in Twitter

Event detection has been applied to Twitter data in recent years. Sakaki et al. [45] focused on detecting earthquake and typhoon events on Twitter using an SVM machine. They modeled the problem as a classification problem. Patrovic et al. [46] tackled the problem of identifying event on Twitter that have never appeared in previous tweets. They used an approach based on the cosine similarity between documents. Ozdikis et al. [47] detected events by clustering tweets that are retrieved through the semantic expansion of their hashtags. They concluded that focusing on hashtags instead of the entire tweet improves the clustering accuracy. Weng and Lee [48] employed a wavelet transformation-based approach to tackle the problem. They built wavelet signals from individual words and modeled the change over time of each word through a sliding window and the H-measure of the signals.

Signals are then clustered with a modularity-based graph partitioning technique and events are represented as subgraphs. Cordeiro [49] also used wavelet transformations. He built signals (wavelets) from hashtags and clustered them into clusters in interval of 5 minutes. The hashtag signals were constructed over time by simply counting the number of occurrence of the hashtag within each interval, resulting in time series. A wavelet transformation is then applied to obtain a time-frequency representation of each signal. After identifying an event, the Latent Dirichlet Allocation (LDA) is applied to reduce the dimensionality of the feature space. More recently, Zhang et al. [50] implemented an event detection system on Twitter that considers the geo-localization of the tweets. Their main intuition is that if a considerable amount of tweets (i.e., a burst) appears and if the semantics of the tweets is significantly different than that of regular tweets at this particular geo-location, then there is an event happening. The candidate events are then identified as true local events through a logistic regression classifier.

Our work differ from the approaches mentioned above as we will be building and analyzing sentiment signals. Our approach consists of analyzing the signal's *trend* over a short period of time using a sliding window w, and comparing that trend with the upcoming sentiment point.

3 Research Plan

The overall goal of this research is to design and build algorithms to automatically detect a shift of sentiment on Twitter with respect to a given subject. Most current research focuses heavily on accurately evaluating the sentiment of textual data, ignoring the problem of automatically detecting when a shift of sentiment happens. In order to detect these sentiment shifts, we will approach the problem as a special case of event detection, algorithms that automatically detect events from signals. Event detection has previously been applied to text streams in which the signal is typically considered to be word frequency. To our knowledge, it has never been used on sentiment signals. We therefore propose to combine sentiment quantification techniques along with event detection methods to achieve our goal. Sentiment quantification is generally measured by estimating the proportion of positive and negative occurrences within a stream of text. In our case, we will focus on stream of tweets from Twitter. The problem that we are tackling rises three important questions that we ought to answer throughout this document:

(i) How to represent a tweet in a way such that (a) its representation embeds the sentiment it reflects, (b) the computer understands, and (c) computations can be performed by the computer

(ii) How can we accurately quantify the prevalence of the positive class and the negative class of a collection of tweet given a representation that satisfies (i)

(iii) How can we identify a shift of sentiment after inquiry (ii) is satisfied

The first goal of this document will answer (i) by exploring sentiment lexicons and feature

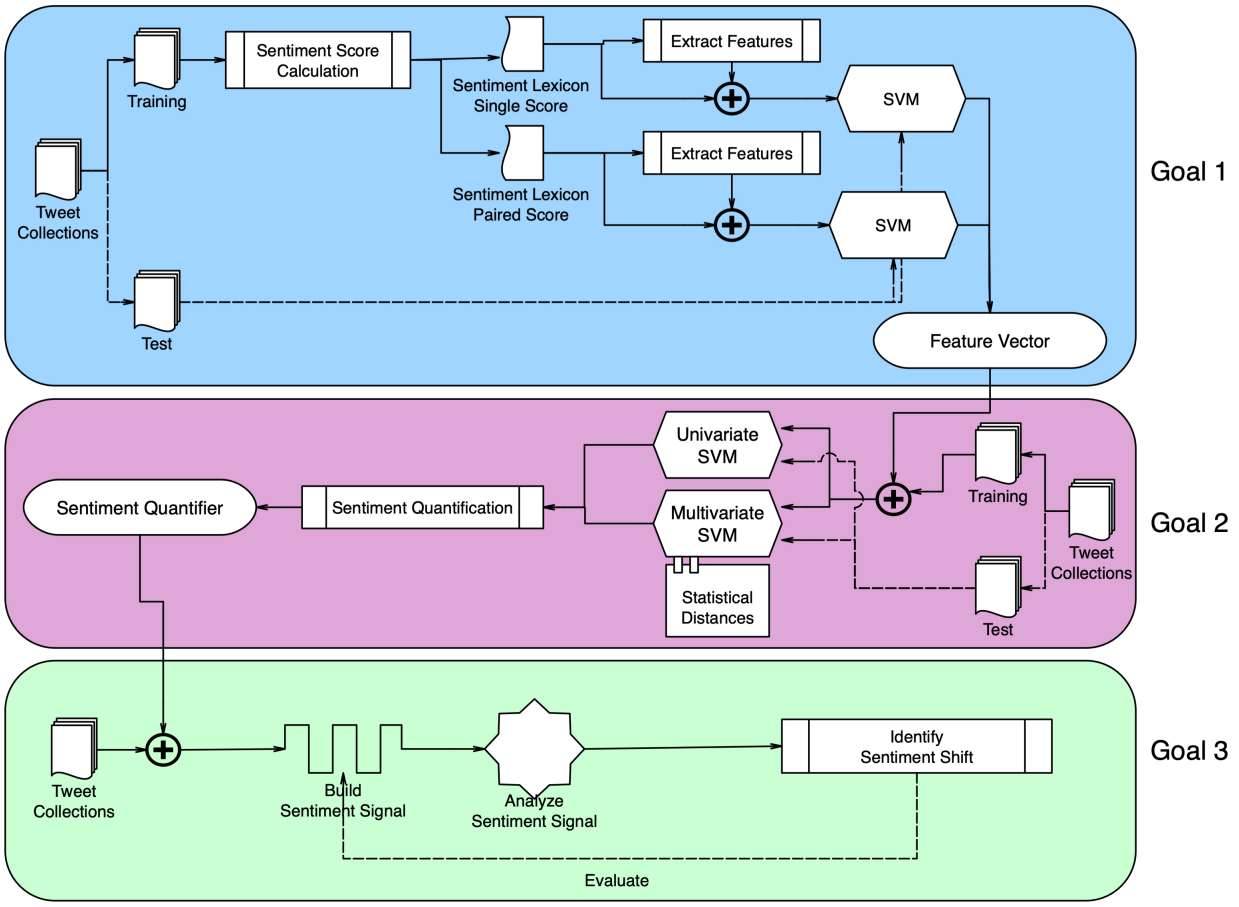

Figure 3.1: Overall architecture of the system

vectors. A vector of real values in the Vector Space Model (VSM) can encode information. Moreover, vectors operations are easy to compute, and the computer can work very efficiently with them. The second goal addresses issue (ii) for which SVM (Support Vector Machine) machines will be explored in combination with statistical distances. Given that the Vector Space Model was chosen to answer (i), Support Vector Machine is a natural choice for trying to answer the sentiment quantification question. The third goal of this document builds on top of (ii), sentiment quantification, and tries to answer (iii) by identifying sentiment shifts. The main idea consists of building sentiment signals from the result of the quantification task, and applying event detection methods to analyze such signals and detect a potential

shift of sentiment. Figure 3.1 depicts the architecture of the system.

3.1 Goal 1: Representing Tweets in the Vector Space Model

From a collection of tweets, we will first build a sentiment lexicon, that is, a list of words with associated sentiment scores. The sentiment scores will be calculated using a probabilistic approach. We will use the resulting sentiment lexicon to represent each tweet in the Vector Space Model (VSM) and use such vectors in a traditional Support Vector Machine (SVM) classifier.

3.1.1 Capturing Word Sentiment

Our method to estimate the sentiment of a word is based on our pilot work [32] in which we estimated the sentiment of a word through a probabilistic approach and an information theoretic approach. Although our formulae worked well for our test set of Amazon product reviews, they might not produce accurate scores with Twitter's data. Indeed, Twitter's language is much different than that of product reviews, and the sentiment score calculation will likely need to be adapted. However, one of the main advantages of the aforementioned approach is that our sentiment scores are accurate whether or not the training dataset is balanced (that is, whether or not there is an equal amount of positive labeled data and negative labeled data in the training dataset).

We believe that this is a very important point to take into consideration and it is the main reason why our new approach is inspired by it. Based on our previous work [32], which produced accurate sentiment scores, we define the score of a word w, $Score(w)$, to be the difference between its positivity, $pos(w)$, and its negativity, $neg(w)$. The positivity of a word is calculated by dividing the positive document frequency of the word with the aggregated

positive document frequency of every word, which basically represents how frequently a word is used in positive tweets. A word w_1 that is used in positive tweets more often than a word w_2 should have a higher positive score. However, this statement is no longer true if w_1 is also used in more negative tweets than w_2. To account for that, the positivity is then normalized by the overall frequency of the word. By doing so, a word that appears in many positive documents and few negative documents will have a higher positive score than a word that appears in the same amount of positive documents but more negative documents. The same calculation is done on the negative documents as well. The score of a word is therefore calculated as follows:

$$Score(w) = pos(w) - neg(w) \tag{1}$$

where:

$$\begin{aligned} pos(w) &= \frac{pdf(w)}{N_{pos}} \times \frac{1}{df(w)} \\ neg(w) &= \frac{ndf(w)}{N_{neg}} \times \frac{1}{df(w)} \end{aligned} \tag{2}$$

and:

$$pdf(w) = \sum_{t \in T_{pos}} x \begin{cases} x = 1 & \text{if } w \in t \\ x = 0 & \text{otherwise} \end{cases}$$

$$ndf(w) = \sum_{t \in T_{neg}} x \begin{cases} x = 1 & \text{if } w \in t \\ x = 0 & \text{otherwise} \end{cases}$$

$$df(w) = \sum_{t \in T} x \begin{cases} x = 1 & \text{if } w \in t \\ x = 0 & \text{otherwise} \end{cases}$$

$$= pdf(w) + ndf(w)$$

$$N_{pos} = \sum_{w \in vocab} pdf(w)$$

$$N_{neg} = \sum_{w \in vocab} ndf(w)$$

We first define three terms: $pdf(w)$, $ndf(w)$, and $df(w)$ where $pdf(w)$ is the positive document frequency of w, i.e., the number of time w occurs in positive tweets from the tweet collection T; $ndf(w)$ is the negative document frequency of w, i.e., the number of time w occurs in negative tweets from the tweets collection T; and $df(w)$ is the total number of occurrences of w in the tweet collection T. Furthermore, N_{pos} is the proportion of positive words in the collection of tweets, i.e., it is the sum of the the positive document frequency pdf of every word in the dictionary; Likewise, N_{neg} is the proportion of negative words in the collection of tweets, i.e., the sum of the negative document frequency $ndf(w)$ of every word in the dictionary.

3.1.2 Building Sentiment Lexicons

The next step is to build a sentiment lexicon. A sentiment lexicon is a list of words associated with, or mapped to, one or several sentiment polarities. There are several approaches to word polarity annotations. *Discrete polarity annotation* label words with a discrete value among *positive*, *negative*, or *neutral*. Such a polarity annotation is used in the MPQA Subjectivity lexicon [51]. *Continuous polarity annotation* assigns words a real value within a range (typically -1.0 to +1.0) that reflects the strength and the orientation of a word. One common polarity annotation is called *fractional polarity annotation* that is defined as a 3-tuple of positive numbers that sums up to 1, wherein each value corresponds to the positivity, negativity and neutrality of the word respectively. The popular SentiWordNet lexicon uses this type of polarity lexicon [52, 53].

The formulae described in the previous section produce real values, so we will use the continuous polarity annotation in our work. After preprocessing the tweets from our training dataset, each unique word will be extracted from the remaining text in order to build a dictionary. Using the formulae from section 3.1.1, we will compute each word's polarity score. We will then compare two scaling techniques in order to get scores in the range [-1, 1]. We will compare the scaling formulae (3) and (4):

$$x' = 2 \times \frac{x - min\_score}{max\_score - min\_score} - 1 \tag{3}$$

$$x" = \frac{x}{max(\mid max\_score \mid, \mid min\_score \mid)} \tag{4}$$

Scaling (3) guaranties that the scores are mapped to the interval [-1, +1] and the minimum score will be -1 whilst the maximum score will be +1. Scaling (4) guaranties that the scores

are within the range [-1, +1] but one of the edge values might not be reached, i.e., if the highest score is +1 the minimum score might be less than or equal to -1 and vice-versa.

We will assess the effect of scaling with both formulae through a straightforward sentiment analysis approach. That is, for each word w in a tweet t we will look up the score in the lexicon l and sum them up. If the resulting score is positive then the tweet is deemed to be positive, if the resulting score is negative the tweet is deemed negative:

$$score(t) = \sum_{w \in t} x \begin{cases} x = score(w) & \text{if } w \in l \\ x = 0 & \text{otherwise} \end{cases}$$

While a single sentiment score gives us information about the polarity strength (its score) and the polarity orientation (its sign) of a word, it does not capture the word's **distribution** across positive and negative occurrences. For example, consider a word w with a score of 0.4, based on formula (1). It could be that its positivity is 0.8 and its negativity is 0.4, or it could be that its positivity is 0.6 and its negativity 0.2, or 0.5 and 0.1. Furthermore, if two words have the same sentiment score, it does not necessarily mean that they have the same positivity and negativity. For instance, if two words have a score of -0.6, they could be the results of $0.11 - 0.71$ or $0.3 - 0.9$. In other words, we are losing information about the word's distribution across the dataset. We will consider two words similar sentiment-wise if their positivity and negativity scores are similar.

We believe that using both the positivity and negativity of words could improve the accuracy of a sentiment analysis machine since it embeds more information than using a single score. Thus, we will build a second lexicon that uses a polarity annotation that is a combination of the *fractional polarity annotation* and the *continuous polarity annotation*.

Using the formula in equation (2) from section 3.1.1, we will build a lexicon that maps each word to a pair of scores $w :< pos, neg >$: its positivity, also referred as $pos(w)$, and its negativity, also referred as $neg(w)$. We will compare the effectiveness and accuracy of each of the single-score lexicon and paired-score lexicon through a sentiment analysis task described later in this document.

3.1.3 Representing Tweets as Feature Vectors

We will investigate and compare two ways of representing a tweet in the Vector Space Model. We will extract each word's sentiment score from the lexicons constructed in the previous section, and derive numerical features from them. A common way of representing documents in the Vector Space Model is through the Bag-Of-Word (BOW) model. In this approach each document is represented by a vector wherein each feature is a word from the dictionary. Therefore, the size of the vector is equal to the size of the vocabulary. We will explore two variations of the BOW model. One wherein each word feature is a binary value (0 or 1) that indicates whether or not the i^{th} word is present in the document. And one wherein each word feature is the computed tf-idf (term-frequency inverse document frequency) value of the i^{th} word within the document. Each feature vector will then be added the sentiment features derived using the lexicon. We will describe these sentiment features below. We do not take into consideration the Part-Of-Speech (POS) of the words based on results from Go et al. [28] that demonstrated that POS is not helpful in Twitter data. We will perform sentiment analysis on a collection of tweets to assess which feature vector is more effective. We will build a sentiment classifier using a Support Vector Machine (SVM). Since it is known to be very effective on text classification [54], a linear kernel will be used for this SVM, based on Mohammad et al. [29] who stated that a linear kernel outperformed other kernels as well

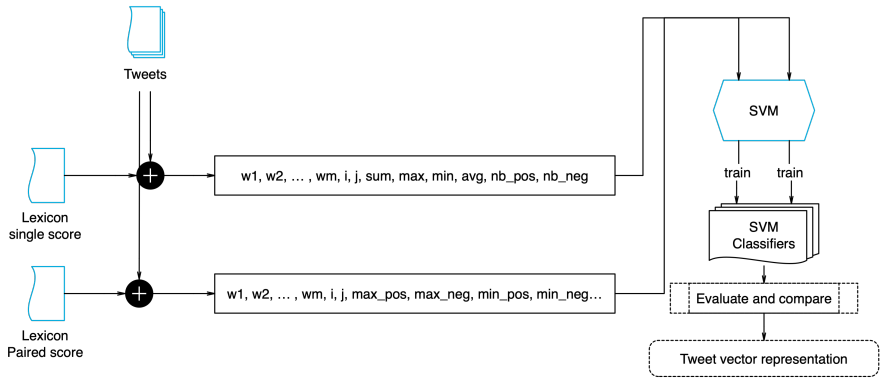

Figure 3.2: Architecture of the SVM classifier for sentiment analysis

as a Maximum Entropy classifier. Figure 3.2 illustrates this process.

3.1.3.1 Feature vector with single score lexicon

The feature vector for the SVM classifier that uses the single-score lexicon will be composed of the following features:

- w_1, w_2, ...,w_m: for each of the m words in the vocabulary If the word is in the tweet then the feature will be equal to 1 or its tf-idf, and 0 otherwise
- i: the number of word that were actually found in the lexicon
- j: the number of token, i.e., the number of word in the tweet
- sum: the overall sentiment score of the tweet calculated by adding up each word's sentiment score
- max: the maximum sentiment score present in the tweet
- min: the minimum sentiment score present in the tweet
- avg: the average sentiment score of the tweet calculated by averaging each word's sentiment score

- *nb pos*: the number of positive token in the tweet
- *nb neg*: the number of negative token in the tweet

3.1.3.2 Feature vector with paired score lexicon

The feature vector for the SVM classifier that uses the paired-score lexicon will be composed of the following features:

- w_1, w_2, ...,w_m: for each of the m words in the vocabulary If the word is in the tweet then the feature will be equal to 1 or its tf-idf, and 0 otherwise
- i: the number of word that were actually found in the lexicon
- j: the number of token, i.e., the number of word in the tweet
- *max pos*: the maximum positive score in the tweet
- *min pos*: the minimum positive score in the tweet
- *max neg*: the maximum negative score in the tweet
- *min neg*: the minimum negative score in the tweet
- *sum pos*: the sum of all positive scores
- *sum neg*: the sum of all negative scores
- *sum*: the overall sentiment score of the tweet calculated by subtracting *neg* from *pos*
- *avg pos*: the average positive sentiment score of the tweet calculated by averaging each word's positive sentiment score

- *avg neg*: the average negative sentiment score of the tweet calculated by averaging each word's negative sentiment score

- *ratio*: the ratio of *avg pos* over *avg neg*

3.1.3.3 Choosing a vector representation

We will study the impact of each feature on the classification performances. Specifically, we will start with the first vector representation, i.e., the bag-of-words model, and measure the performances of the classifier. Then we will add one feature at a time and evaluate the performances of the system after each addition to evaluate the impact of the new feature. The vector features that yields the most accurate results will be used for our tweet vector representation.

3.2 Goal 2: Quantifying Sentiment from Tweets

Given a set of tweets, our goal is to quantify the proportion of tweets that are deemed positive and the proportion of tweets that are deemed negative in the set. We will build several multivariate SVM machines that use the feature vectors resulting from Goal 1, and compare them to the univariate SVM machine from Goal 1 to evaluate which is better suited for tweet quantification.

3.2.1 Classification vs Quantification

A classifier evaluates the class label of individual instances while a quantifier evaluates the class label on the aggregate level (which is the objective of Goal 1). It is important to keep in mind that a perfect classifier is a good quantifier, but the inverse is not true. Also, a good classifier is not necessarily a good quantifier. Let's look at the example in Figure 3.3

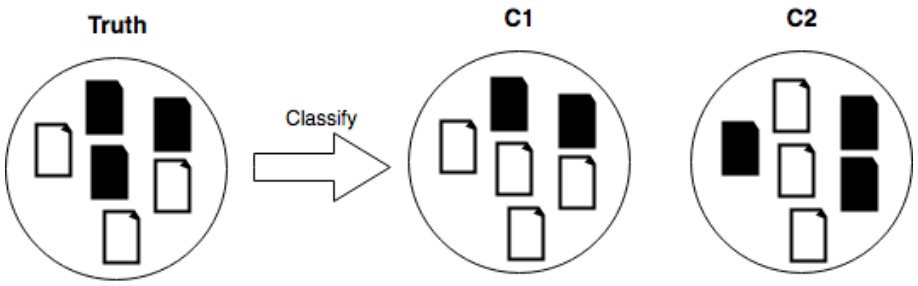

Figure 3.3: Classification example

wherein two classifiers C1 and C2 classified 6 documents. 3 documents are labeled black and 3 are labeled white. The class distribution is therefore 0.5 for the black class and 0.5 for the white class.

The binary classifier C1 has a false positive rate (FPR) that differs from its false negative rate (FNR) but it correctly classified 5 out of 6 document. It is therefore a very good classifier. However, the distribution of the black class in C1 is 0.33 and the distribution of the white class is 0.66. It is therefore a poor quantifier, since the class prevalence was not kept. The binary classifier C2 has a FPR equal to its FNR. It correctly classified 2 documents out of 6 and is thus a terrible classifier. However, the distribution of the black class is 0.5 and the distribution of the white class is 0.5. It is a good quantifier because it did predict the right prevalence of both classes. Since the FPR and FNR are equal in C2, they will compensate each other so that the distribution of the classes is estimated correctly. We call C2 a perfect quantifier. If all misclassifications were toward a particular class, i.e., what we call a statistical bias, it would not affect a classifier but would affect a quantifier.

3.2.2 Sentiment Quantification of Tweets

A traditional SVM machine can be used to quantify by classifying each unlabeled documents and by then counting how many documents belong to the positive class and how

many documents belong to the negative class. We call this approach *univariate SVM* and we will use it as our baseline for Goal 2.

Because a traditional SVM optimizes a univariate loss function, it classifies each item one by one, independently of each other, i.e., an item does not impact how another item is classified. Even if the classifier correctly quantifies the positive and negative class proportions in the training set, there is no guarantee that the proportion of positive documents and negative documents will be the same in the training test set. In fact, we are explicitly expecting a change in the proportion of the positive class and negative class ratios when a shift of sentiment happens. Thus, such a quantifier will most likely suffer from statistical bias.

To overcome this problem, we will use a Support Vector Machine (SVM) for multivariate performance measures. That is, we will use a SVM that optimizes a nonlinear multivariate loss function rather than an univariate loss function. It considers hypotheses $\bar{h} : \bar{\chi} \rightarrow \bar{\gamma}$ that maps an entire set of documents into a set of labels instead of mapping an individual document to an individual label as in $h : \chi \rightarrow \gamma$. So, it allows items to be arranged in structures (graphs of items, trees of items, sets, etc.) in which each item has an attributed label. However, the way an item is labelled is influenced by other items. Thorsten Joachims [55] developed such an SVM machine named SVM^{perf}.

We will perform quantification using SVM^{perf} similarly to our baseline, that is, by classifying each unlabeled documents and then counting how many documents belong to the positive class and how many documents belong to the negative class. Specifically, we will use SVM^{perf} with several *statisticaldistance* metrics and compare them to our baseline univariate SVM.

The Kullback-Leibler Divergence (KLD) is a loss function that measures how one

probability distribution p diverges from a second predicted distribution q. It is defined as follows:

$$KLD(p,q) = \sum_{c_i \in C} p(c_i) \cdot log \frac{p(c_i)}{q(c_i)} \tag{5}$$

We will use the $SVM(KLD)$ from [36] and compare it to our own approaches described below. The idea here is to compare various measures of statistical distance that have different mathematical properties and comparing their performance in a sentiment quantification task. To our knowledge, this work has not yet been investigated.

Our first sentiment quantification machine is a SVM^{perf} that optimizes the Hellinger Distance instead of KLD. The Hellinger Distance is a statistical distance used to measure the similarity between two probability distributions. HD is defined as follows:

$$HD(p,q) = \frac{1}{\sqrt{(2)}} \cdot \|\sqrt{(p)} - \sqrt{q}\|_2 \tag{6}$$

The second SVM^{perf} will optimize the Bhattacharyya distance. The Bhattacharyya distance is another statistical distance that measures how similar two probability distributions are. It is defined as follows:

$$D_B(p,q) = -\ln(BC(p,q)) \tag{7}$$

where:

$$BC(p,q) = \sum_{x \in X} \sqrt{p(x)q(x)} \tag{8}$$

is the Bhattacharyya coefficient.

The third statistical distance that we will optimize through SVM^{perf} is the Jensen

Shannon Divergence. The Jensen Shannon Divergence is a smoothed and symmetrized version of the Kullback-Leibler Divergence. It is defined as follows:

$$JSD(p,q) = \frac{1}{2}KLD(p,m) + \frac{1}{2}KLD(q,m) \tag{7}$$

where:

$$m = \frac{1}{2}(p+q)$$

The next statistical distance that we will use in the multivariate SVM machine is the Total Variation Distance. It is yet another metric used to measure the distance between two probability distributions and is defined as follows:

$$TVD(p,q) = \frac{1}{2}\sum_{x \in X} \|p(x) - q(x)\| \tag{9}$$

The last statistical distance that we will use is another symmetrized version of KLD called Resistor-Average Distance introduced by Johnson and Sinanović [56]. It is equal to the harmonic mean of both Kullback-Leibler distances $KLD(p,q)$ and $KLD(q,p)$ and is formally defined as follows:

$$RAD(p,q) = \left[\frac{1}{KLD(p,q)} + \frac{1}{KLD(q,p)}\right]^{-1} \tag{10}$$

We will call these SVM $SVM^{perf}(KLD)$, $SVM^{perf}(HD)$, $SVM^{perf}(BD)$, $SVM^{perf}(JSD)$, $SVM^{perf}(TVD)$, and $SVM^{perf}(RAD)$ respectively.

3.2.3 Notes on Statistical Distances

From a mathematical perspective, a function d in a space χ is said to be a **distance** if for any $x, y, z \in \chi$ the following three axioms are satisfied:

(i) $d(x, y) > 0$ when $x \neq y$ and $d(x, y) = 0$ if and only if $x = y$

(ii) $d(x, y) = d(y, x)$

(iii) $d(x, y) + d(x, z) \geq d(y, z)$

The axiom (i) implies that the distance must be non-negative and respect the identity of indiscernible, axiom (ii) implies that the distance must be symmetric, and axiom (iii) implies that the distance satisfies the triangular inequality.

If a distance measure only satisfies axiom (i) i.e., it is non-negative but not symmetric, it is called a directed divergence measure whereas if a distance measure satisfies both axioms (i) and (ii), i.e., it is non-negative and symmetric, it is called a non-directional divergence measure [57] [58]. A true distance measure can be considered a divergence measure, but a divergence measure does not always satisfies all three axioms and is therefore not a true distance. We warn the reader that the term *distance* must not be taken rigorously when discussing some distance measures throughout this document, e.g., the Bhattacharrya distance, as they might not obey all three fundamental axioms mentioned above, and are therefore not a *distance* in the proper sense. We will now discuss a few properties of all 6 statistical measures mentioned in the previous section.

Kullback-Leibler Divergence

- The Kullback-Leibler Divergence does not satisfy axiom (ii) and (iii), that is, it is not symmetric and it does not satisfy the triangular inequality

- KLD is therefore not a distance but a pseudo-distance or directed divergence measure
- It is considered a measure of divergence because of its ratio $\frac{p(x)}{q(x)}$, that is, the difference in the probability distributions is large when the ratio is far from 1
- $KLD(p, q)$ is undefined if there exists a $q(x) = 0$ for which $p(x) \neq 0$
- $KLD(p, q)$ has no upper bound, that is, KLD's limit goes to $+\infty$ when $q(x)$ is infinitely small

Hellinger Distance

- The Hellinger Distance satisfies all three axioms, it is symmetric, non-negative, and satisfies the triangular inequality
- HD is therefore a true distance
- $HD(p, q)$ is bounded, it has a lower bound of 0 and an upper bound of 1 (due to the $1/\sqrt{2}$ term in the formula)
- $HD(p, q)$ is well defined

Bhattacharrya Distance

- The Bhattacharyya Distance does not satisfy axiom (iii), that is, is does not satisfy the triangular inequality
- The Bhattacharrya Distance is symmetric
- BD is therefore not a distance in the proper sense but a non-directional divergence measure

- As per its definition that employs the natural logarithm, BD is undefined is there exists a $q(x) = 0$ for which $p(x) = 0$
- BD has an upper bound

Jensen-Shannon Divergence

- The Jensen-Shannon Divergence does not satisfy axiom (iii), that is, it does not satisfy the triangular inequality
- JSD is symmetric
- JSD is therefore not a distance in the proper sense but a non-directional divergence measure
- JSD is a smoothed and symmetrized version of the Kullback-Leibler Divergence
- $JSD(p, q)$ is bounded, it has a lower bound of 0 and an upper bound of $\ln(2)$
- Because it uses the KLD, $JSD(p, q)$ is undefined if there exists a $p(x) = 0$ or $q(x) = 0$

Total Variation Distance

- The Total Variation Distance satisfies all three axioms, it is symmetric, non-negative, and satisfies the triangular inequality
- TVD is therefore a true distance
- $TVD(p, q)$ is bounded, it has a lower bound of 0 and an upper bound of 1
- $TVD(p, q)$ is well defined

Resistor-Average Distance

- The Resistor-Average Distance does not satisfy axiom (iii), that is, it does not satisfy the triangular inequality
- RAD is symmetric
- RAD is therefore not a distance in the proper sense but a non-directional divergence measure
- RAD is another symmetrized version of KLD. It is equal to the harmonic mean of both $KLD(p, q)$ and $KLD(q, p)$
- $RAD(p, q)$ is not defined when either $KLD(p, q)$ or $KLD(q, p)$ is equal to 0

Although these are only a few of the numerous distance measures and divergence measures available, for no apparent reason, the Kullback-Leibler Divergence has become the *de facto* standard measure for statistical divergence. Our work will compare other distance measures to see which is best for sentiment quantification.

In this section, we have described how we intend to improve current sentiment quantification methods so as to be able to quantify the class distribution (positive, negative) of a set of tweets. The next section will describe the third goal of this document, i.e., how to automatically detect a shift of sentiment with respect to an entity or subject.

3.3 Goal 3: Detecting Sentiment Shifts

Once we are able to accurately quantify the prevalence positive class instances and negative class instances from a set of tweets, we use our best-performing approach to build a *sentiment signal* over time that we will analyze in order to identify a shift of opinion.

3.3.1 Building a Sentiment Signal

From Goal 2 we are be able to extract the distribution, i.e., the proportion of positive tweets and negative tweets during the time interval from t_{i-1} to t_i from a set T of tweets (where t_{i-1} to t_i is equivalent to one time unit e.g., day, hour, minutes etc...). We can then compute the volume of positive tweets $q(pos)_t$ and the volume of negative tweets $q(neg)_t$ for point t in terms of percentage. Then, similar to [48], we consider this sentiment signal as a time series that is composed of a sequence of discrete-time sentiment points $S(t_i)$ where each $S(t_i)$ is the sentiment towards a given entity during the interval t_{i-1} to t_i, defined as follows:

$$S(t) = [S(t_1), ..., S(t_i), S(t_{i+1}), ..., S(t_N)] \tag{11}$$

A sentiment point is represented through a single value computed from the difference between the volume of positive tweets $q(pos)_t$ and the volume of negative tweets $q(neg)_t$. This way, if the proportion of positive tweets if greater than the proportion of negative tweets, the sentiment point will carry a positive value, similarly, if the proportion of positive tweets is less than the proportion of positive tweets, the sentiment point will carry a negative value. Thus, a sentiment point is defined as follows:

$$S(t_i) = q(pos)_{t_i} - q(neg)_{t_i} \tag{12}$$

3.3.2 Detecting a Sentiment Shift

Thenceforth, we define a sentiment shift as:

Definition 1. *A sentiment shift is a sudden change in the sentiment signal $S(t)$ of a given entity that occurred in the time interval t_{i-1} to t_i.*

Once our sentiment signal is represented as a time series of sentiment points, we can analyze it to detect a possible shift in the sentiment signal. The intuition is that although most sentiment signals will vary gradually over time, occasionally there will be a **sudden** and **dramatic** change. If a sentiment shift happens, the next incoming sentiment point will not follow the previous pattern, or trend, and that is an indicator of a potential sentiment shift. The idea thus consists of analyzing the signal's *trend* over a short period of time using a sliding window and compare that trend with the upcoming sentiment point as soon as it becomes available. In particular, let w be a sliding window of size $|w|$. If the new sentiment point does not fit the trend determined by w, it would indicate that an unexpected sentiment value was detected.

From the perspective aforementioned, our detection algorithms should then have two components:

1. a mathematical mean to try to capture and represent the signal's trend inside the window

2. a mathematical mean of comparing the incoming data point to the window's trend to decide whether or not the point belongs to the trend

We must detect the sentiment shift as soon as the data point becomes available, thus requiring our algorithms to be computable in real time. We will explore several window sizes and compare the detection accuracy to see the impact of having more data points within the window. In addition we will explore the several questions raised by the choice of window size. Particularly, does having a larger window add more noise or value? What happens when the trend window is large enough to encompass a prior sentiment shift, partially, and fully? How would it impact the detection accuracy?

We will investigate several algorithms to detect a sentiment shift and compare their effectiveness. Such algorithms belong to the class of "change point detection" algorithms, these are usually divided into two categories: offline algorithms, and online algorithms. The former contain algorithms that consider all points within the data set and they therefore do not work on streaming data. Thus, we cannot use such algorithms. The latter class, on the other hand, contains algorithms that are real-time and therefore process data points as they become available. Their goal is to detect a change point as soon as they come in. Online algorithms fit our needs, thus we will focus this class.

Online algorithms are further divided into two subcategories: supervised methods, which require sufficient amount and diversity of data for training purposes, and unsupervised methods which work with unlabeled data, and do not require prior training data. Supervised methods include traditional classifier such as Naïve Bayes, Support Vector Machine, Decision Tree, Nearest Neighbor and so forth. We will focus on online unsupervised methods in this work.

Unsupervised change point detection methods can be labeled under 6 categories based upon how they work and behave. Although we will not describe all 6 categories in this document, we will discuss why some cannot apply to our situation. Descriptions and examples of the various categories can be found in the work of Aminikhanghahi and Cook [59]. Likelihood ratio methods, subspace model methods, and kernel-based methods require the use of several windows and are therefore qualified as as not complete online (they are $n+k$-real time) and therefore do not fit our requirements. Probabilistic methods are n-real time (requiring n points before the incoming point) but they usually estimate probability distributions and therefore require a large number of data points to be accurate. Graph-based methods are

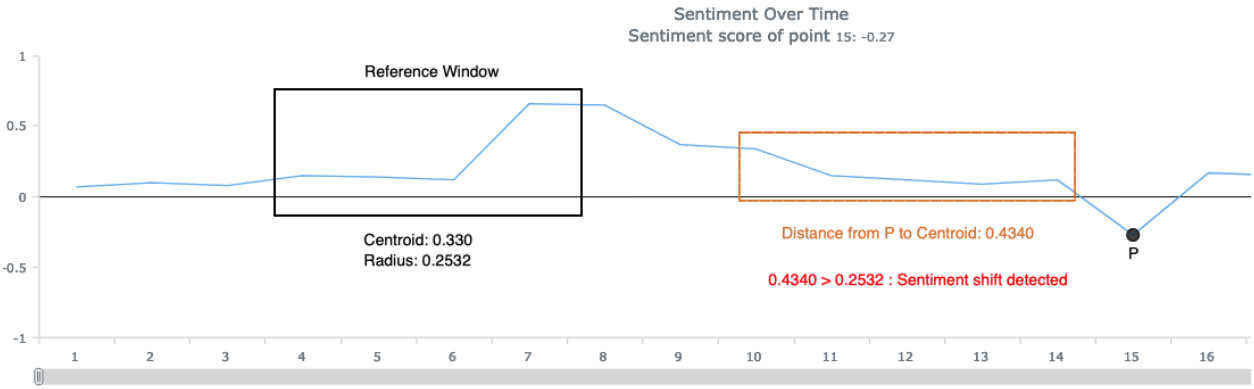

Figure 3.4: Baseline detection illustration

also n-real time but they rely upon the definition of the graph structure rather than the data properties and observations. Finally, while some clustering methods are offline, some are n-real time and only use one single window of size n, which fits our requirements.

There are 3 approaches as to using a window to capture and monitor incoming data points: periodic, incremental, and reactive. In the periodic techniques the model is updated time to time. In the incremental approach the model is updated when the data changes. Finally, in the reactive approach the model is updated when it no longer suits the data (typically when a change point is detected).

We will compare our algorithms to a baseline implemented by Dang-Hoan Tran[60] who investigated change point detection in streaming data using a reactive approach. We adapted his approach to our case.

3.3.2.1 Baseline: Reactive model-fitting approach

This approach consists of having two windows characterized by a Centroid and a Radius both of size n. The reference window is static and is updated when a change point is detected, in our case we will update the window when the incoming data point was is believed to carry a sentiment shift. The second window is a sliding window that slides

over time, this window eventually becomes the reference window upon the detection of a sentiment shift. The distance between the Centroid of the sliding window and the new data point is computed and used as a threshold to determine whether or not the point carries a sentiment shift. The Centroid X_0 of the window, the Radius R of the window, and the distance between the Centroid and the new data point X are computed as follows:

$$X_0 = \frac{\sum_{i=1}^{|w|} X_i}{|w|}$$

$$R(w) = \sqrt{\frac{\sum_{i=1}^{|w|}(X_i - X_0)^2}{|w|}} \tag{13}$$

$$d(X, X_0) = \sqrt{(X_0 - X)^2} = |(X_0 - X)|$$

The decision to make then becomes:

$$\text{Sentiment shift} = d(X, X_0) > R(w) \tag{14}$$

If the distance between the Centroid of the sliding window and the new data point is greater than the Radius of the window, then a shift is detected, and the sliding window becomes the reference window, otherwise the sliding window keeps sliding forward. Figure 3.4 illustrates this process.

3.3.2.2 Our approaches

While the baseline algorithm uses a reactive approach, all of our techniques will use an incremental approach, that is, the trend window will slide one time unit at a time, thereby updating the model upon receiving a data point.

Discrete signal properties: Sig prop

Our first approach consists of using discrete signal properties such as energy E and power P. The energy of the signal is a quantitative measure of its strength while the power measures how the energy is divided over a certain period of time. Additionally, the *instantaneous power* is the energy of a single data point at a fixed instant t. The energy E, power P, and instantaneous power Pi of the sentiment signal over the window w are defined as follows:

$$\begin{aligned} E(w) &= \sum_{i=1}^{|w|} |w(i)|^2 \\ P(w) &= \frac{E(w)}{|w|} = \frac{1}{|w|} \sum_{i=1}^{|w|} |w(i)|^2 \\ Pi(w(i)) &= w(i)^2 \end{aligned} \tag{15}$$

The idea in this approach is to compute the power in the window that contains the new data point upon receiving it. Then we can compare that power to a range of *acceptable* power, derived from the power contained in the trend window at time $t-1$, that is, prior to receiving the new data point.

We first compute the energy of the trend window and derive its power $P(w)_{t-1}$. From $P(w)_{t-1}$ we can derive min_{P_w} and max_{P_w} which are the minimum expected power and maximum expected power respectively. Upon receiving the new data point, the trend

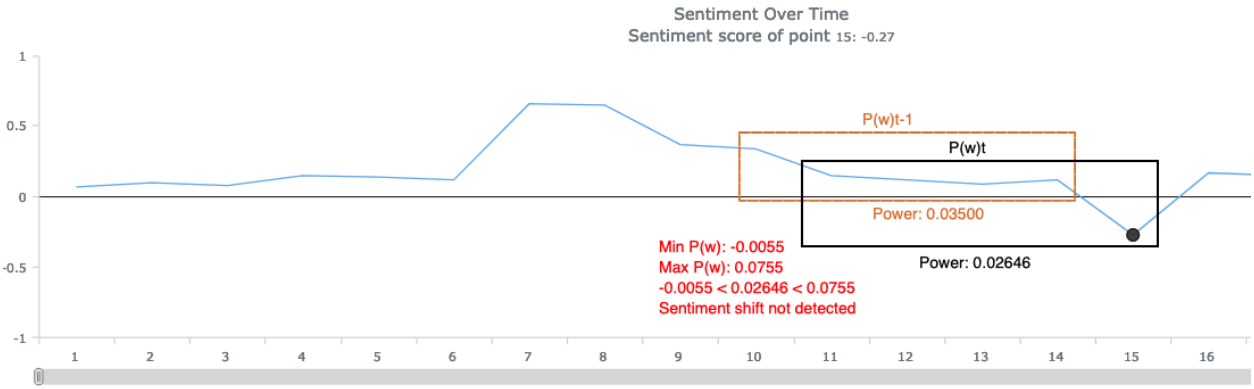

Figure 3.5: Sig prop detection illustration

window slides by one time unit and we can then compute the new power $P(w)_t$. Finally, we compare $P(w)_{t-1}$ to min_{P_w} and max_{P_w}. While we know how to compute both $P(w)_{t-1}$ and $P(w)_t$, we now show how to compute min_{P_w} and max_{P_w}:

$$\mu(w) = \frac{1}{|w|}\sum_{i=1}^{|w|} w(i)$$

$$\sigma(w) = \sqrt{\frac{1}{|w|}\sum_{i=1}^{|w|}(w(i) - \mu(w))^2} \tag{16}$$

$$min_{P_w} = P(w)_{t-1} - \sigma(w)$$

$$max_{P_w} = P(w)_{t-1} + \sigma(w)$$

The decision to make then becomes:

$$\text{Sentiment shift} = (P(w)_t < min_{P_w}) \vee (P(w)_t > max_{P_w}) \tag{17}$$

Intuitively, if the new power falls within the range min_{P_w} - max_{P_w}, the data point was *expected* and is therefore not carrying a sentiment shift. However, if the new power is out-

side the range, i.e., lower than the minimum expected power or greater than the maximum expected power, then it means that the new data point was not an expected value and could therefore indicates a sentiment shift. Figure 3.5 illustrates this process.

Distance point to point: Dist p2p

This approach consists in capturing the signal's trend inside the window by computing the Pythagorean distance point to point between each consecutive point in the window, and derive the average distance needed to travel from one point to another. The distance from the last point inside the window to the new data point is then compared to a threshold value computed from the standard deviation of the distances and their average, since the standard deviation is a measure of how widely a distribution of values is spread out from their average. Although very similar to the baseline method, our approach differs in several ways:

1. Our method uses an incremental approach as opposed to reactive approach. This implies that an incoming data point is compared to the trend that it precedes, as opposed to be compared to the latest trend that encompassed a sentiment shift

2. We use the average of the distances point to point to capture the trend as opposed to using the average values within the window

3. Any distances computed in our model is computed through the Pythagorean theorem (also known as two-dimension Euclidean distance) which uses a two dimensions Euclidean plane as opposed to using a one-dimension plane.

Note: Although the baseline algorithm was designed for multiple dimensions, we adapted it for a single dimension to fit our data.

The Pythagorean distance d between two sentiment points s1(v1, t1) and s1(v2, t2), the standard deviation σ of the distances, and the threshold $Threshold_{p2p}$ are defined as follows:

$$
\begin{aligned}
d(s1, s2) &= \sqrt{(v1 - v2)^2 + (t1 - t2)^2} \\
\mu(w) &= \frac{1}{|w| - 1} \sum_{i=1}^{|w|-1} d(w(i), w(i+1)) \\
\sigma(w) &= \sqrt{\frac{1}{|w| - 1} \sum_{i=1}^{|w|-1} (d(w(i), w(i+1)) - \mu(w))^2} \\
Threshold_{p2p} &= \mu(w) + \sigma(w)
\end{aligned}
\tag{18}
$$

The decision to make then becomes:

$$
\text{Sentiment shift} = d(\text{last point in w}, \text{new point}) > Threshold_{p2p} \tag{19}
$$

Similar to the previous approach, if the distance d between the last point in the trend window and the new data point is greater than the threshold distance $Threshold_{p2p}$, the point is deemed to carry a sentiment shift, otherwise the point is deemed to belong to the trend. Figure 3.6 illustrates this process.

In order to see the impact of the window size on the detection accuracy, we introduce Dist p2p* from this method, where the only difference resides in the threshold calculation

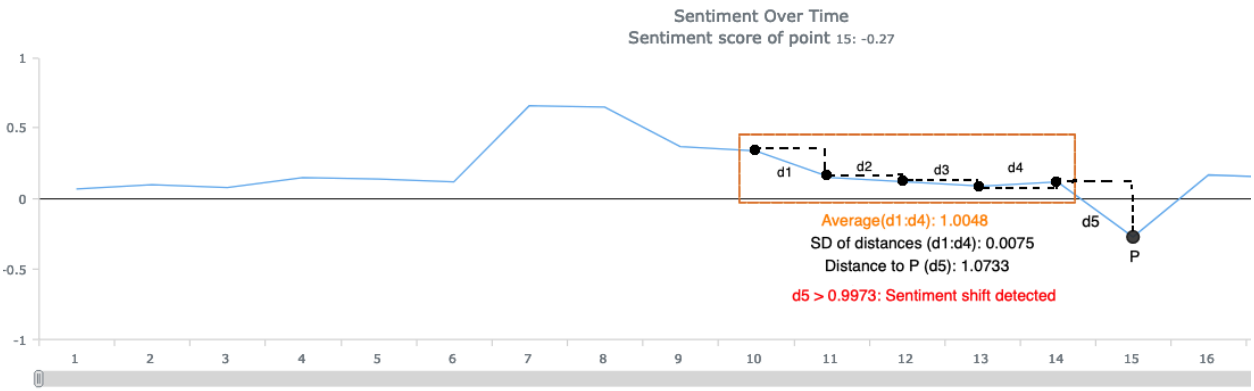

Figure 3.6: Dist p2p detection illustration

which becomes:

$$Threshold_{p2p^*} = \mu(w) + |w| \times \sigma(w) \tag{20}$$

Distance point to vector: Dist p2v

This approach follows the same process as the Distance point to point (Dist p2p) approach, but differs in the way the distances are calculated. Indeed, rather than calculating the distance from one point to the next, the idea here is to consider the distance between the point at t_i to the vector that connects both points prior to it: t_{i-2} and t_{i-1}. Thus, we are no longer using the Pythagorean distance but the distance d between a point $P_0(x_0, y_0)$ and a line that is defined by two points $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$. We then compute the average

distance and the standard deviation similarly to equation 17:

$$d(P_1, P_2, P_0) = \frac{|(y_2 - y_1)x_0 - (x_2 - x_1)y_0 + x_2y_1 - y_2x_1|}{\sqrt{(y_2 - y_1)^1 + (x_2 - x_1)^2}}$$

$$\mu(w) = \frac{1}{|w| - 2} \sum_{i=1}^{|w|-2} d(w(i), w(i+1), w(i+2)) \tag{21}$$

$$\sigma(w) = \sqrt{\frac{1}{|w| - 2} \sum_{i=1}^{|w|-2} (d(w(i), w(i+1), w(i+2)) - \mu(w))^2}$$

$$Threshold_{p2v} = \mu(w) + \sigma(w)$$

The decision to make then becomes:

$$\text{Sentiment shift} = d(\text{last vector}, \text{new point}) > Threshold_{p2v} \tag{22}$$

and follows the same idea as Dist p2p. Figure 3.7 illustrates this process.

Equivalently to Dist p2p*, we introduce Dist p2v* from Dist p2v, where the only difference resides in the threshold calculation which becomes:

$$Threshold_{p2v^*} = \mu(w) + |w| \times \sigma(w) \tag{23}$$

Distance point to aggregated vector: Dist p2av

Our last approach resembles Dist p2v with the major difference that the computed

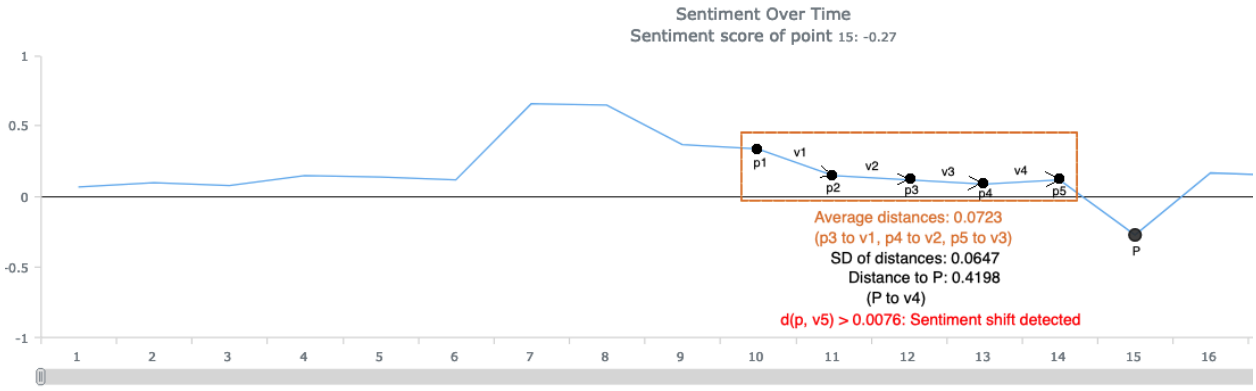

Figure 3.7: Dist p2v detection illustration

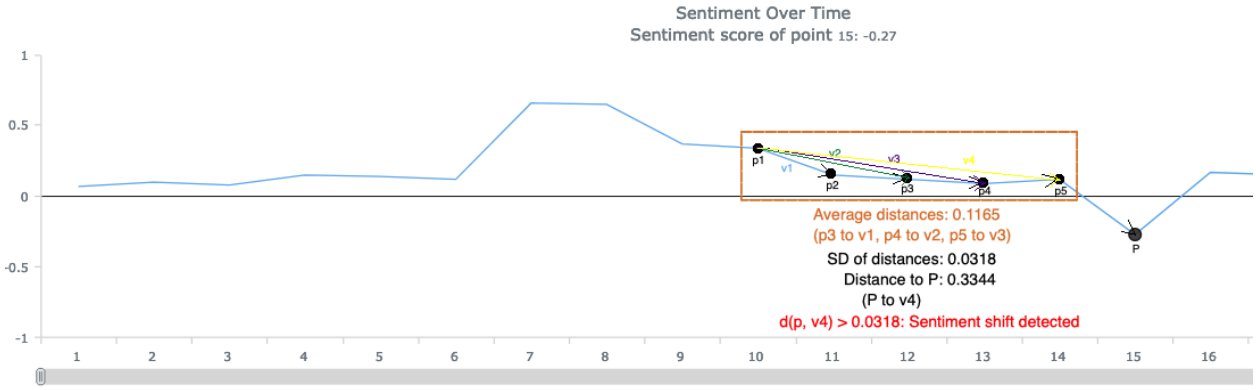

Figure 3.8: Dist p2av detection illustration

distance is the distance from a point to the aggregated vectors prior to the point within the window. For the n^{th} point in the trend window, we compute the distance from the point to a vector which is the sum of all vector from 0 to $n-1$ inside the window that precede the point. The average distance remains the average of all distances computed inside the window. The computation of the distance d, and the standard deviation remain the same as

equation 21 while the average $\mu(w)$ changes:

$$d(P_1, P_2, P_0) = \frac{|(y_2 - y_1)x_0 - (x_2 - x_1)y_0 + x_2y_1 - y_2x_1|}{\sqrt{(y_2 - y_1)^1 + (x_2 - x_1)^2}}$$

$$\mu(w) = \frac{1}{|w| - 2} \sum_{i=2, j=i+1}^{|w|-2} d(w(1), w(i), w(j)) \tag{24}$$

$$\sigma(w) = \sqrt{\frac{1}{|w| - 2} \sum_{i=2, j=i+1}^{|w|-2} (d(w(1), w(i), w(j)) - \mu(w))^2}$$

$$Threshold_{p2av} = \mu(w) + \sigma(w)$$

The decision to make then becomes:

$$\text{Sentiment shift} = d(\text{sum of all vectors prior to point}, \text{new point}) > Threshold_{p2av} \tag{25}$$

and follows the same idea as Dist p2p. Figure 3.8 illustrates this process. Equivalently to Dist p2v*, we introduce Dist p2av* from Dist p2av, where the only difference resides in the threshold calculation which becomes:

$$Threshold_{p2av^*} = \mu(w) + |w| \times \sigma(w) \tag{26}$$

In this chapter, we have discussed our three goals which answer the questions of (1) how to represent a Tweet as a vector of features which captures its sentiment, (2) how to accurately quantify the proportion of positive Tweets and negative Tweets from a set of

Tweets, and (3) how to create a sentiment signal over time of a given topic which is then analyzed to detect when a shift of sentiment happen. We have also provided a description of the methods that we developed to address each of the aforementioned goals. In the next chapter, we will describe the datasets being used as well as the methods employed to evaluate each of our goals.

4 Experimental Evaluation

4.1 Evaluation of Goal 1

4.1.1 Dataset

We will evaluate our approach over several commonly used datasets. The datasets that we will use are sets of tweets annotated with a class label chosen from *positive, negative*, or *neutral*. Because we are dealing with 2-class sentiment analysis and 2-class sentiment quantification, we will ignore tweets that are labeled *neutral* for both training and testing.

4.1.1.1 Sentiment Scores

To evaluate the sentiment scores, we plan to use the datasets from the International Workshop on Semantic Evaluation Task 4 A: Sentiment Analysis in Twitter, from 2013 through 2016 (namely SemEval2013-task_A, SemEval2014-task_A, SemEval2015-task_A, and SemEval2016-task_A). We will also use the Sentiment Strength Twitter (SST) dataset created by [64] and modified by [65] to have the *positive, negative*, or *neutral* classes. Additionally, we will use the Sanders dataset. Table 4.1 depicts the size of each of these datasets. The datasets are publicly available on the Internet and the tweets contained in each of them were annotated manually to ensure a high quality of class labels.

Although the SemEval-task_A (2013-2016) datasets are partitioned into three subsets (training, dev, test), the Sanders and the SST datasets are not. We will split those into two subsets with 80% used for the training set and 20% reserved for the testing set. For the SemEval datasets, the training and dev subsets will be combined and used for training while

the test sets will be used for testing.

4.1.1.2 Feature Vector - Univariate SVM

To evaluate which feature vector representation is best, we will use the datasets from the International Workshop on Semantic Evaluation Task 4 D: Sentiment Analysis in Twitter 2016 and 2017 (namely SemEval2016-task_D and SemEval2017-task_D).

The SemEval2016-task_D and SemEval2017-task_D) datasets are split into 4 subsets: train, dev, devtest, and test. We will combine the train, dev, and devtest subsets for training and use the remaining test subset for testing. In addition, both the SemEval2016-task_D and SemEval2017-task_D datasets are composed of tweets that belong to a particular topic (that is the twitter query), and are labeled either positive or negative. While the topic is ignored during the training part, i.e., all tweets are combined and used for training, the topic will actually be used during the testing phase. We will use each topic from the test set as a separate test subset during testing. Table 4.2 depicts the size of each of these datasets.

Due to terms imposed by Twitter, the datasets available online cannot contain the tweets themselves, rather they contain the tweet IDs. Scripts are provided to download the actual tweets from Twitter. Since some tweets become unavailable over time, our datasets will consequently be a subset of the collection of IDs.

Table 4.1: Twitter sentiment analysis datasets

	Train			Dev			Test		
	# positive	# negative	Total	# positive	# negative	Total	# positive	# negative	Total
SemEval 2013-task_A	3,640	1,458	5,098	575	340	915	1,475	559	2,034
SemEval 2014-task_A	3,640	1,458	5,098	575	340	915	982	202	1,184
SemEval 2015-task_A	3,640	1,458	5,098	575	340	915	1,038	365	1,403
SemEval 2016-task_A	3,094	863	3,957	843	391	1,234	7,059	3,231	10,290
	Train (80%)			Test (20%)			Train + Test		
	# positive	# negative	Total	# positive	# negative	Total	# positive	# negative	Total
SST	989	842	1,831	263	195	458	1,252	1,037	2,289
Sanders	418	54	872	101	118	219	519	572	1,091

Table 4.2: Twitter sentiment quantification dataset

	Topics	Positive	Negative	**Total**
SemEval2016-train-task_D	60	2,841	582	3,423
SemEval2016-dev-task_D	20	778	279	1,057
SemEval2016-devtest-task_D	20	893	216	1,109
SemEval2016-test-task_D	100	8,212	2,339	10,551
SemEval2017-train-task_D	100	8,212	2,339	10,551
SemEval2017-test-task_D	125	2,463	3,722	6,185

4.1.2 Method

4.1.2.1 Evaluation of the sentiment scores

Commonly used metrics will be used to assess the effectiveness and accuracy of the normalized scores obtained from formulae (3) and (4). Specifically, we will use the following metrics:

$$Accuracy = \frac{\#\text{of correctly labeled tweets}}{\#\text{of tweets in the dataset}}$$

$$Precision = \frac{\sum \text{True positive}}{\sum \text{predicted conditon positive}}$$

$$Recall = \frac{\sum \text{True positive}}{\sum \text{conditon positive}}$$

$$\text{F1-Score} = 2 \times \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

From this point forward, $score(w)$ will use the scores that yields the best results according to these metrics.

4.1.2.2 Evaluation of the univariate SVM

The performance of the lexicons will be evaluated through sentiment quantification using a traditional univariate SVM classifier. We will train a SVM classifier with a linear kernel using the training data to create the lexicons as stated in section 3.1. The effectiveness of the sentiment lexicons will be evaluated using the testing data.

Commonly used metrics will be used to evaluate the univariate quantifier: the Kullback-Leibler Divergence (KLD), the Mean Absolute Error (MAE), and the Relative Absolute Error (RAE). The Kullback-Leibler Divergence measures how one probability distribution p

diverges from a second predicted distribution $\hat{p}$ and is defined as follows:

$$KLD(\hat{p}, p) = \sum_{c_i \in C} p(c_i) \cdot log \frac{p(c_i)}{\hat{p}(c_i)}$$

The Mean Absolute Error and the Relative Absolute Error are the absolute error between the class prevalence of two quantities and are defined as follows:

$$MAE(\hat{p}, p) = \frac{1}{|C|} \sum_{c \in C} |\hat{p}(c) - p(c)|$$

$$RAE(\hat{p}, p) = \frac{1}{|C|} \sum_{c \in C} \frac{|\hat{p}(c) - p(c)|}{p(c)}$$

We will calculate the macro average KLD, MAE, and RAE, the harmonic means of each metric. For instance, the macro average KLD will be defined as follows:

$$\text{Macro Average KLD} = \frac{\sum_{i=1}^{n} KLD_i}{n}$$

where n is the number of instances, i.e., the number of datasets in our case.

As discussed in Chapter 3, KLD is not defined in some special cases, namely when the predicted prevalence $\hat{p}$ is zero. To circumvent this problem we smooth both prevalence p and $\hat{p}$ through additive smoothing, that is, $p(c_i)$ becomes:

$$p^s(c_i) = \frac{p(c_i) + \epsilon}{1 + \epsilon * 2}$$

where ϵ is the smoothing factor and is defined as follows:

$$\epsilon) = 12 * |dataset|$$

$\hat{p}$ will be smoothed similarly. We will use the smoothed KLD throughout the rest of the document. We will use the most accurate univariate SVM classifier as a baseline for Goal 2, through a "classify and count" approach.

4.2 Evaluation of Goal 2

4.2.1 Dataset

To evaluate the performances of the various multivariate SVM, we will use the same datasets that we have used in Goal 1.

4.2.2 Method

We will evaluate the performance of a quantifier using commonly used metrics: the Kullback-Leibler Divergence (KLD), the Mean Absolute Error (MAE), and the Relative Absolute Error (RAE) that we described previously.

We will train the multivariate SVM using the train subsets and test on the test collection. Additionally, when using both SemEval2016-task_D and SemEval2017-task_D datasets we will train each quantifier on each individual topic (that is a total of 100 topics when combining train, dev, devtest) and test each quantifier on each of the 100 topics for SemEval2016-test-task_D, and each of the 125 topics for SemEval2017-test-task_D . The metrics will be computed for each run and will then be averaged to yield the final score.

Table 4.3: Sentiment shift detection dataset

	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
Total points	20	20	20	20	20	20	20	20	20	20
# True label	2	2	2	2	0	0	0	0	0	2
# False label	18	18	18	18	20	20	20	20	20	18

4.3 Evaluation of Goal 3

4.3.1 Dataset

The dataset to evaluate Goal 3 is composed of 10 synthetic sentiment signals that were manually created. The sentiment signals were hand-build so as to be able to emulate various situations that could happen with real data that we might not have been able to collect through real Twitter streams. Each signal consists in a total of 20 points where each individual data point either carries a sentiment shift or not. A data point that carries a sentiment shift is therefore labeled as "true" whereas a data point that does not carry a sentiment shift is labeled as "false". Table 4.3 summarizes the statistics of each signal. As shown in Table 4.3, there are 5 signals that each contain 2 sentiments shifts while the remaining 5 signals do not contain any sentiment shift. Although the dataset seems to be biased and unbalanced, it actually reflects what would be expected from real sentiment signals, that is, a much higher number of data points that do not carry a sentiment shift. Indeed, sentiment shifts are expected to be sparse and infrequent in real life applications, so we replicated this behavior in our synthetic data. All signals were built so as to cover various situations that could potentially occur with real sentiment signals. Thus, our signals were designed to account for the following factors:

- Nature of signal: positive signal or negative signal

- Duration of shift: brief (1 data point) or long (4 data points)
- Direction of shift: positive shift or negative shift
- Amplitude of shift: small amplitude vs large amplitude
- Number of shifts: If more than one shift occurs in a signal, several cases are to be considered that could affect the accuracy of the detection algorithms: (a) The trend window does not encompass the former shift, (b) the trend window encompasses the former shift partially, and (c) the trend window encompasses the entire former shift
- Brevity of shift: As opposed to the duration, the brevity of the shift indicates the number of data point necessary for the shift to occur. That is, if the shift takes several data points to occur, it is not considered sudden and does not fit the definition of a sentiment shift. Consequently, it shall not be considered as an actual sentiment shift.

Figure 4.1 through 4.10 show the shape of each sentiment signals. Table A.1 (Appendix 1) provides a detailed description of each individual signal.

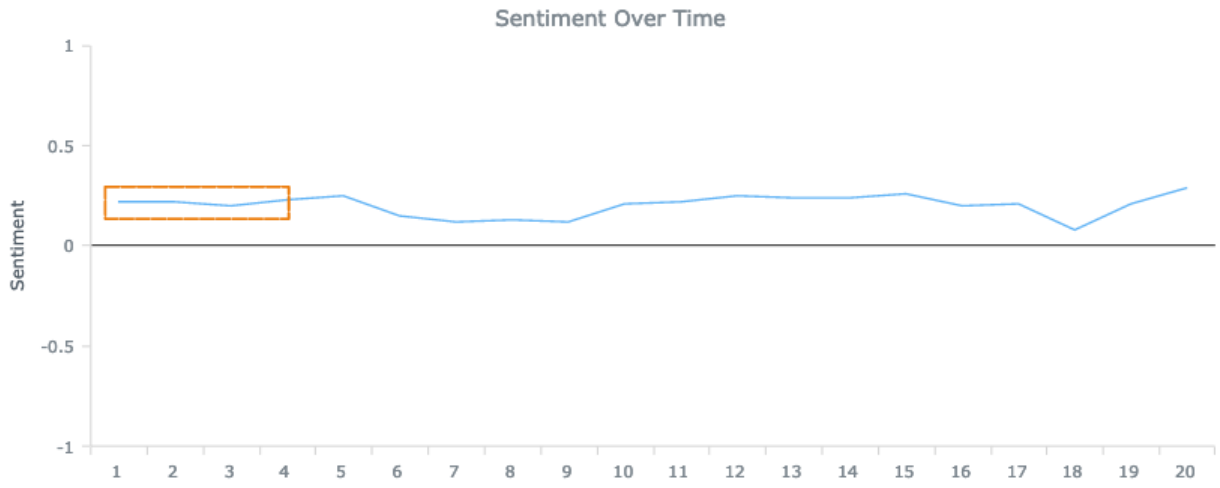

Figure 4.1: Sentiment Signal 1

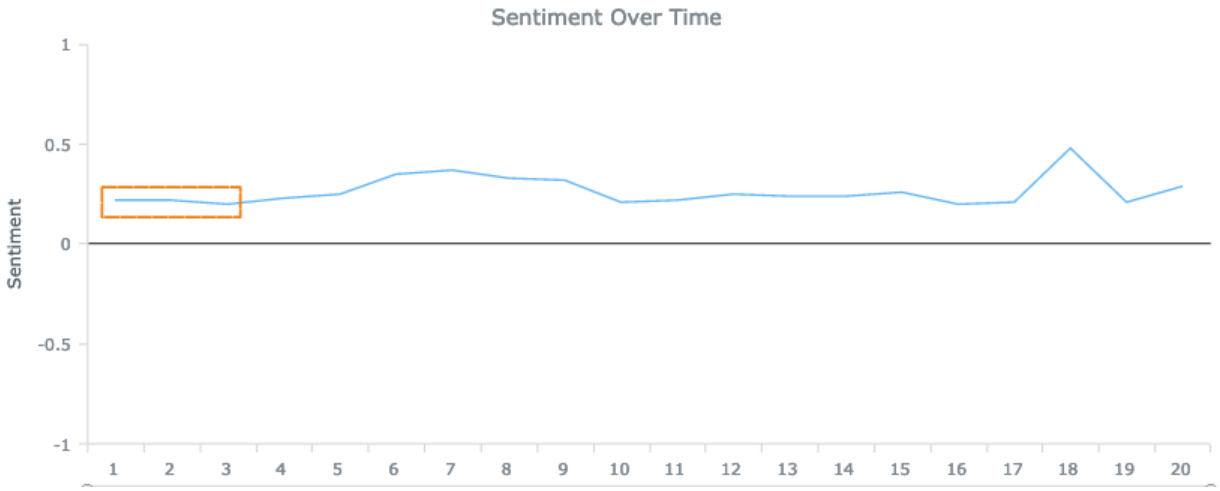

Figure 4.2: Sentiment Signal 2

Signal 1 is a positive signal containing 1 positive sentiment shift on data point 6 with a duration of 4 points, and a positive shift on data point 18 with a duration of 1 point. Signal 2 is a positive signal containing 1 negative sentiment shift on data point 6 with a duration of 4 points, and a negative shift on data point 18 with a duration of 1 point. Signal 3 is a negative signal containing 1 negative sentiment shift on data point 6 with a duration of 4 points, and a negative shift on data point 18 with a duration of 1 point. Signal 4 is a negative signal containing 1 positive sentiment shift on data point 6 with a duration of 4 points, and

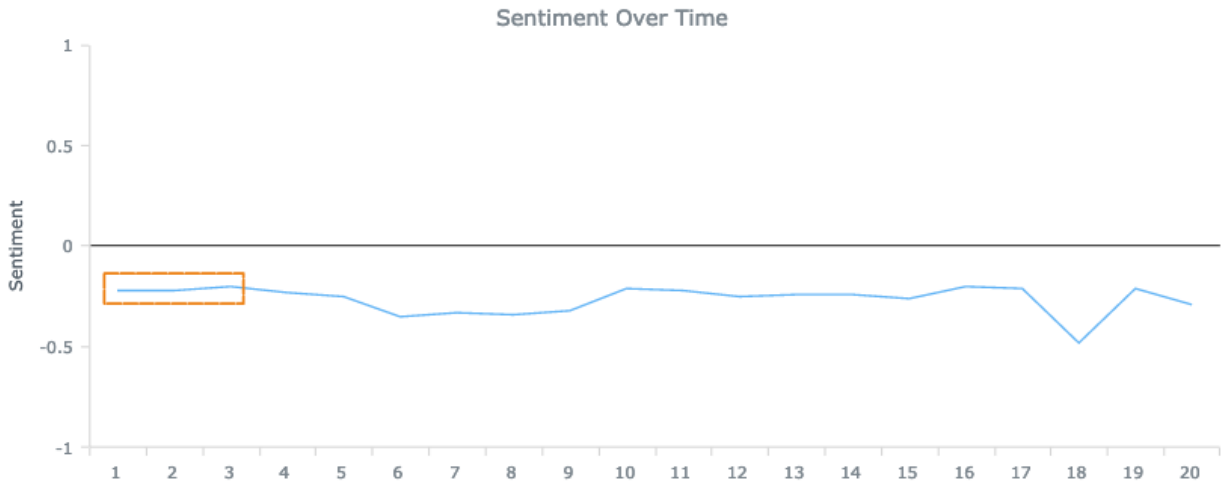

Figure 4.3: Sentiment Signal 3

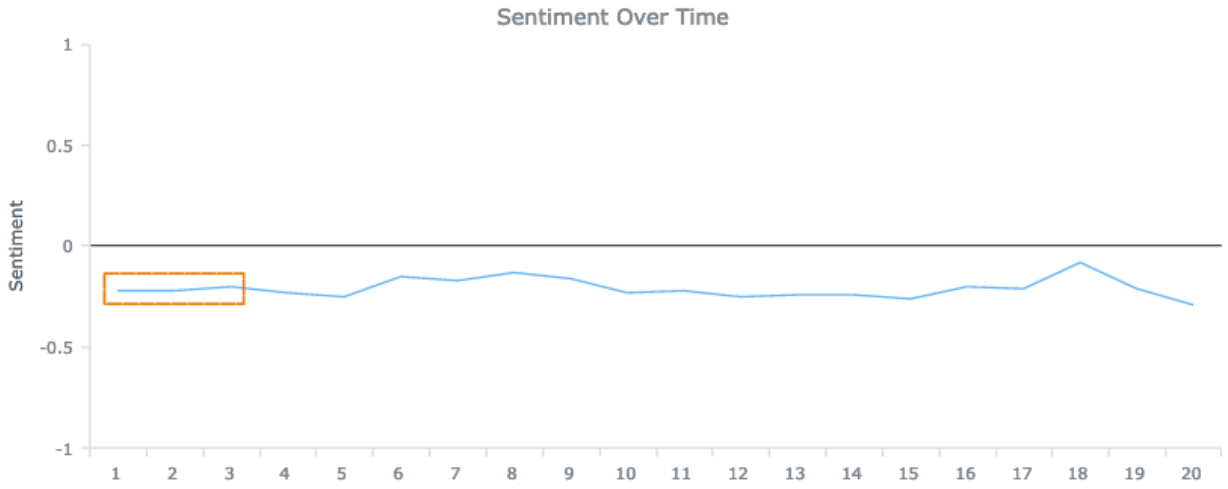

Figure 4.4: Sentiment Signal 4

a positive shift on data point 18 with a duration of 1 point. Signal 5 is a monotonously decreasing signal containing 0 sentiment shift. Signal 6 is a monotonously increasing signal containing 0 sentiment shift. Signal 7 is a decreasing signal with a gradual decline containing 0 sentiment shift. Indeed, although a change of sentiment can be observed, the brevity is too large for it to be considered a sentiment shift. Signal 8 is an increasing signal with a gradual incline containing 0 sentiment shift. Indeed, although a change of sentiment can be observed, the brevity is too large for it to be considered a sentiment shift. Signal 9 is a fluctuant signal

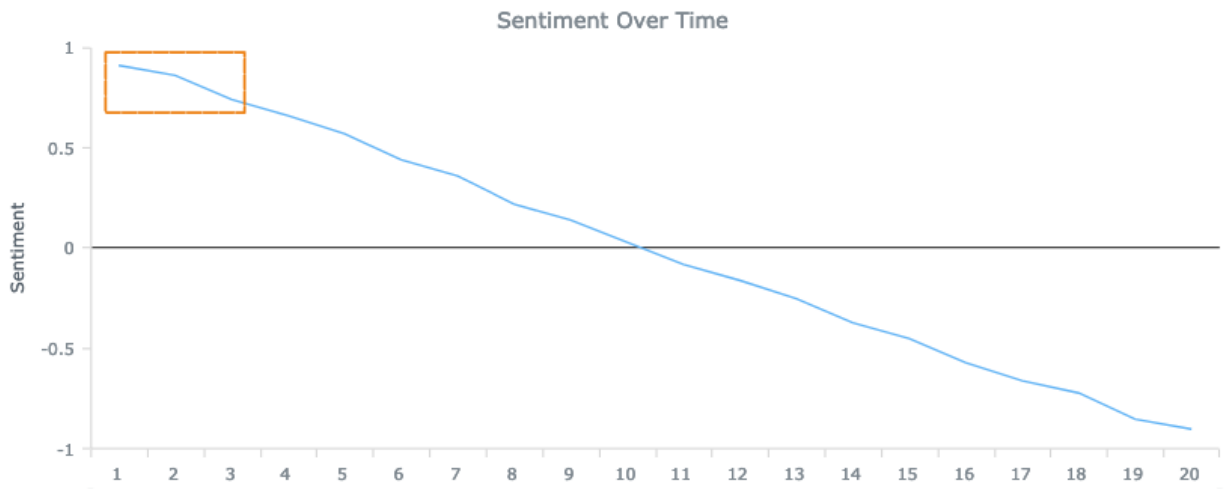

Figure 4.5: Sentiment Signal 5

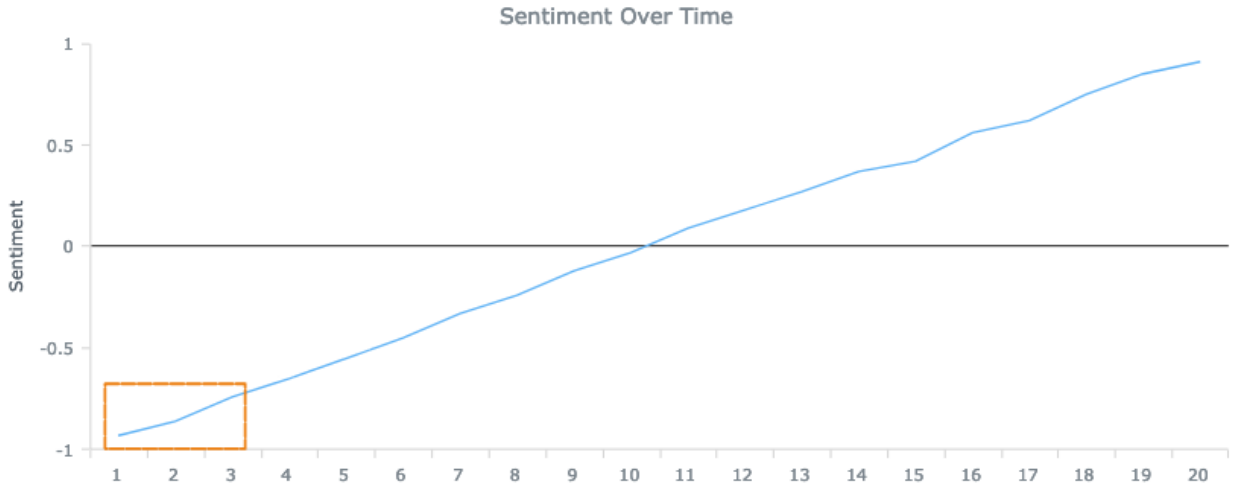

Figure 4.6: Sentiment Signal 6

containing 0 sentiment shift. Indeed, although sentiment shifts are thought to be observed, the ubiquitous fluctuations indicate that the signal is constantly changing, and shifts are therefore considered "normal". Signal 10 is a random signal containing 1 positive sentiment shift on data point 7 with a duration of 1 point, and a negative sentiment shift on data point 15 with a duration of 1 data point.

Figure 4.7: Sentiment Signal 7

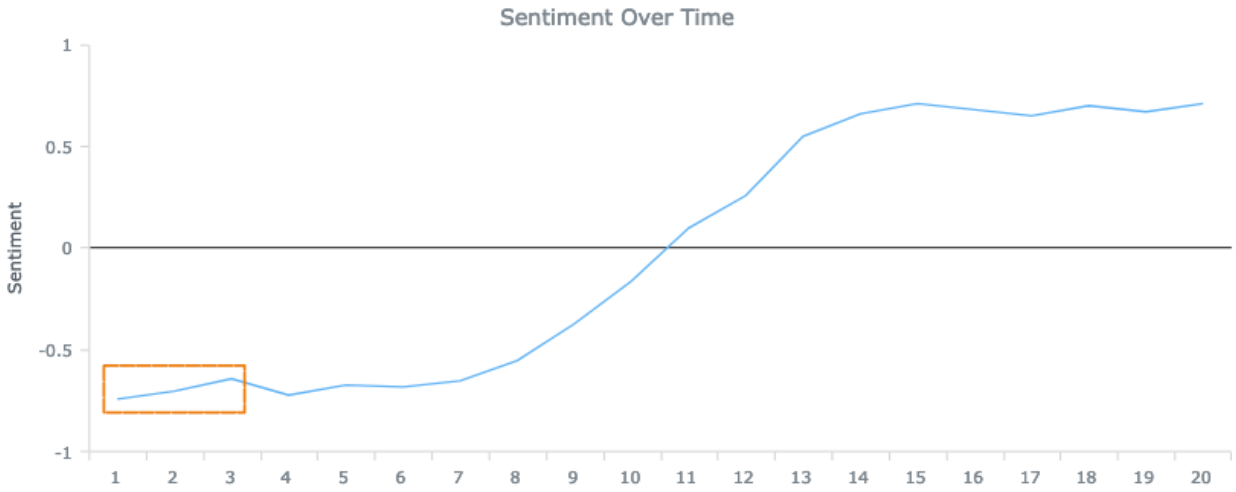

Figure 4.8: Sentiment Signal 8

4.3.2 Method

We will evaluate the sentiment shift detection algorithms described in Section 3.3.2 using. As a consequence, a data point is deemed *true positive* if it carries a sentiment shift and the algorithm detected a shift on that point whereas a data point is deemed *true negative* if it does not carry a sentiment shift and the algorithm did not detect a shift on that point. Similarly, a data point is said to be *false positive* if it does not carry a sentiment shift and

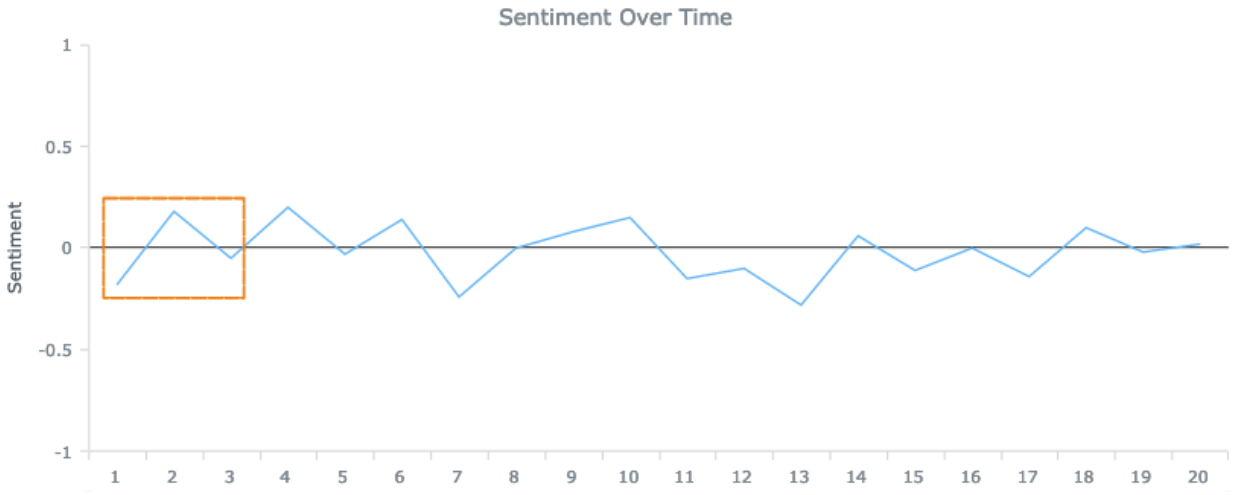

Figure 4.9: Sentiment Signal 9

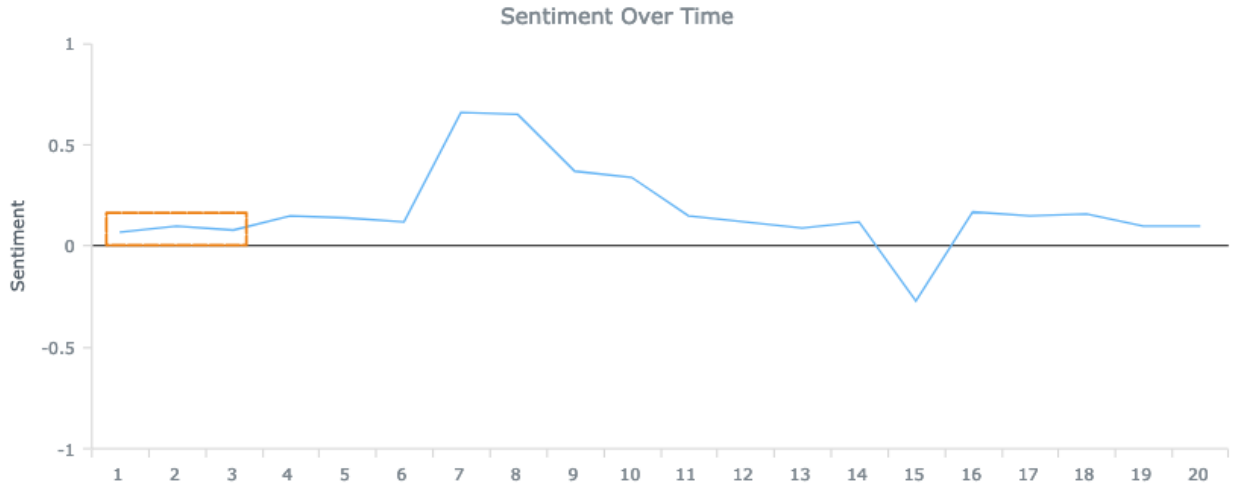

Figure 4.10: Sentiment Signal 10

the algorithm detected a shift on that point whereas a data point is said to be *false negative* if it carries a sentiment shift but the algorithm did not detect a shift on that point.

The sentiment shift detection algorithms use a window of various sizes to capture the signal's trend. The window size will range from 3 to 10 with an increment of 1, yielding a total of 8 different windows. The points contained within the trend window are not being evaluated. Consequently, the bigger the trend window, the lesser the number of points to evaluate within the signal. For example, for a window size of 6, there will be 14 points left

Table 4.4: Sentiment shift detection dataset - signal combined

	w3	w4	w5	w6	w7	w8	w9	w10	total
Total points	170	160	150	140	130	120	110	100	1080
# True label	10	10	10	6	5	5	5	5	56
# False label	160	150	140	134	125	115	105	95	1024

to be considered, since all signals are made of 20 points.

For all 8 windows and for all 10 signals within that window, we will compute all 4 metrics described below. For each algorithm, we then compute and report the micro-average. We will also compute and report the macro-average, that reports the average with no knowledge about the shape of the signals nor the size of the window (c.f. Table 4.4), as opposed to computing each metric per window.

Metrics

To evaluate the performances we will use the Accuracy, Fall-out (or False Positive Rate), Miss rate (or False Positive Rate), and the balanced accuracy, defined as follows:

$$Accuracy = \frac{\text{true positive} + \text{true negative}}{\text{true positive} + \text{true negative} + \text{false positive} + \text{false negative}}$$

$$\text{Fall-out} = FPR = \frac{\text{false positive}}{\text{false positive} + \text{true negative}}$$

$$\text{Miss rate} = FNR = \frac{\text{false negative}}{\text{false negative} + \text{true positive}}$$

$$\text{Balanced Accuracy} = \frac{TPR + TNR}{2}$$

where TPR is the True Positive Rate, and TNR is the True Negative Rate, defined as follows:

$$\text{TPR} = \frac{\text{true positive}}{\text{true positive} + \text{false negative}}$$

$$\text{TNR} = \frac{\text{true negative}}{\text{true negative} + \text{false positive}}$$

Note: Signals that do not contain sentiment shift (i.e., Signal 5, Signal 6, Signal 7, Signal 8, Signal 9) will not be evaluated using the Miss rate, since they lack data points labeled as "true", and therefore have 0 true positive data points.

The evaluation of the performances of the algorithms focuses on three aspects: (1) the overall performances, (2) type I errors (i.e., False Positive), and (3) type II errors (i.e., False Negative). The Type I error is evaluated through the Fall-out (or FPR), and measures the capacity of an algorithm to detect a shift when no shift is actually occurring whilst the Type II error is evaluated through the Miss rate (or FNR), and measures the capacity of an algorithm to not detect a shift when a shift is actually occurring. Both (2) and (3) are therefore important metrics to consider for such an application. Naturally, the lower the FPR and the lower the FNR, the better the algorithm.

Although the accuracy is a very good indicator of the algorithms' overall performance, it will be biased since 94.8% of the dataset is composed of data points labeled as "false" against 5.2% of the data points labeled as "true". The balanced accuracy will circumvent this problem by taking the average of the True Positive Rate (TPR) and the True Negative Rate (TNR), thereby normalizing the true positive and true negative predictions.

5 Experimental Results and Discussions

5.1 Results of Goal 1: Representing Tweets in the Vector Space Model

5.1.1 Capturing Word Sentiment

5.1.1.1 Results

We implemented four variations of the sentiment score formula described in Section 3.1.1:

(i) unmodified formula as described in equation (2)

(ii) equation (2) where pdf, ndf, and df are normalized by the size of the tweet, that is, $pdf = \frac{pdf}{|tweet|}$, $ndf = \frac{ndf}{|tweet|}$, and $df = \frac{df}{|tweet|}$

(iii) equation (2) where we remove duplicate words from the tweet, e.g., *This restaurant was so so good* becomes *This restaurant was so good*

(iv) equation (2) where we remove duplicate word and use the relative pdf, ndf, df as explained up above

Then for each scaling formulae (c.f. equation 3 and equation 4) we compared all four variations of the sentiment score formula. Table 5.1 depicts the averaged results while Table 5.2 depicts the detailed results, i.e., the results for each dataset.

As we can see in Table 5.1 (avg) the scaling formula (4) achieves a slightly higher accuracy (74.25% against 73.76%) and a higher precision (0.85 vs 0.74) while the scaling

Table 5.1: Averaged performance of the different formulae

Scaling Formula (3)

	Variation i	Variation ii	Variation iii	Variation iv	Average
Accuracy	73.69%	73.73%	73.78%	**73.84%**	73.76%
Precision	**0.75**	0.73	0.74	0.74	0.74
Recall	0.89	**0.93**	0.90	**0.93**	0.91
F1-Score	0.80	**0.82**	0.81	**0.82**	0.81

Scaling Formula (4)

	Variation i	Variation ii	Variation iii	Variation iv	Average
Accuracy	73.98%	74.33%	74.00%	**74.71%**	74.25%
Precision	0.85	0.85	0.85	0.85	0.85
Recall	0.73	0.73	0.73	**0.74**	0.73
F1-Score	0.78	**0.79**	0.78	**0.79**	**0.79**

formula (3) achieves a higher recall (0.91 vs 0.73) and F1-score (0.81 vs 0.79). These results are not statistically significant and do not allow us to distinguish which scaling formula performs best, it is therefore necessary to look at the detailed results depicted in Table 5.2 and Table 5.3. Tables 5.2 and 5.3 reveal that the scaling formula (3) often yields a recall of 1 or very close to 1, while the scaling formula (4) does not. As per the definition of the Recall, having a recall of 1 implies having a true positive rate (TPR) of 1 and a false negative rate (FNR) of 0. Such a system is very good at predicting the positive class but is unable to predict the negative class. Since we are dealing with sentiment quantification, such a system is inappropriate. We will choose the scaling formula (4) to scale our sentiment scores.

We now focus on the performances of all four variations of the scaling formula (4). Since all four variations yields very similar results, we are unable to distinguish which would better capture the sentiment of words. However, the nature of variation (i) and variation (iii) implies that the pdf, ndf, and df of the words are not normalized with the size of the tweet. Consequently, words that share a similar document frequency will most likely have the same sentiment. Intuitively, it is preferable to use variation (ii) or variation (iv) which use the normalized pdf, ndf, and df. Both variation (ii) and variation (iv) yield equivalent

Table 5.2: Detailed performances of the different formulae for each dataset

Scaling Formula (3)	Variation i	Variation ii	Variation iii	Variation iv
SST				
Accuracy	**71.83%**	63.10%	67.47%	63.10%
Precision	**0.73**	0.61	0.72	0.61
Recall	0.80	**0.97**	0.71	**0.97**
F1-Score	**0.77**	0.75	0.72	0.75
Sanders				
Accuracy	72.15%	**72.60%**	72.15%	**72.60%**
Precision	**0.79**	0.69	0.69	0.69
Recall	0.54	**0.73**	**0.73**	**0.73**
F1-Score	0.64	**0.71**	**0.71**	**0.71**
SemEval 2013				
Accuracy 72.62%	75.86%	73.80%	**75.96%**	
Precision	0.73	**0.76**	0.74	**0.76**
Recall	**1**	0.96	0.99	0.97
F1-Score	0.84	**0.85**	**0.85**	**0.85**
SemEval 2014				
Accuracy	82.94%	**83.28%**	83.19%	**83.28%**
Precision	0.83	**0.85**	0.83	**0.85**
Recall	**1**	0.97	1	0.97
F1-Score	0.91	0.91	0.91	0.91
SemEval 2015				
Accuracy	73.98%	76.48%	74.77%	**77.05%**
Precision	0.74	**0.78**	0.75	**0.78**
Recall	**1**	0.96	0.99	0.63
F1-Score	0.85	**0.86**	0.85	**0.86**
SemEval 2016				
Accuracy	68.61%	71.05%	**71.29%**	71.05%
Precision	0.69	0.70	**0.71**	0.70
Recall	1	1	1	1
F1-Score	0.81	**0.83**	**0.83**	**0.83**

results, however, variation (ii) is slightly more computationally efficient and scales better with the size of the dataset since the removal of duplicated words is omitted.

Experimental results show that the scaling formula (4) with the sentiment score variation (ii) is the best. It is therefore the sentiment score formula that we will use throughout the rest of the document.

Table 5.3: Table 5.2 (Cont)

Scaling Formula (4)	Variation i	Variation ii	Variation iii	Variation iv
		SST		
Accuracy	70.96%	**72.49%**	71.18%	72.27%
Precision 0.74	**0.76**	0.74	**0.76**	
Recall	0.76	0.76	0.76	0.76
F1-Score	0.75	**0.76**	0.75	**0.76**
		Sanders		
Accuracy	73.06%	75.80	73.52%	**77.63%**
Precision	0.71	0.74	0.73	**0.75**
Recall	0.69	0.73	0.68	**0.76**
F1-Score	0.70	0.74	0.70	**0.76**
		SemEval 2013		
Accuracy 72.76%	71.73%	**73.21%**	72.37%	
Precision	0.89	0.89	0.89	0.89
Recall	0.71	0.70	**0.72**	0.71
F1-Score	0.79	0.78	**0.80**	0.79
		SemEval 2014		
Accuracy	**81.50%**	81.08%	80.83%	81.08%
Precision	0.94	**0.95**	0.94	**0.95**
Recall	**0.83**	0.82	0.82	0.82
F1-Score	0.88	0.88	0.88	0.88
		SemEval 2015		
Accuracy	**71.42%**	70.85%	70.99%	70.49%
Precision	**0.92**	**0.92**	**0.92**	0.91
Recall	**0.67**	**0.67**	**0.67**	0.66
F1-Score	**0.78**	0.77	0.77	0.77
		SemEval 2016		
Accuracy	74.16%	74.05%	74.28%	**74.40%**
Precision	0.88	0.88	0.88	0.88
Recall	0.72	0.72	0.72	0.72
F1-Score	0.79	0.79	0.79	0.79

5.1.1.2 Discussions

Sentiment Score Formula Variations

In variation (ii) a word will have a stronger sentiment (positive or negative) when used in a concise manner, that is, in a short tweet rather than a long one. Indeed, when a tweet is short then each word potentially contains "more" sentiment since the contribution of the word to the overall sentiment of the tweet is greater. For instance, consider two

Table 5.4: Effect of normalizing the document frequency of words

Sentiment score	lame	insane	scandal	excellent	fabulous	congrats
without normalization	-0.106	-0.106	-0.106	0.342	0.342	0.342
with normalization	-0.081	-0.038	-0.021	0.175	0.246	0.228

positive tweets related to a restaurant's service, such as *Excellent drinks and great food* and *Excellent!!*. Although the former provides more information, the positivity of the tweet is mainly due to two words, e.g., *Excellent* and *great*, thus both words each contribute about 50% to the overall positivity of the tweet. While in the latter, the overall positivity of the tweet is due to one single word, e.g., *Excellent*. Clearly, the same word expresses a stronger sentiment in the latter tweet rather than in the former. However, variation (i) would assign both *Excellent* and *great* a *pdf* of 1. Through the use of variation (ii), the *pdf* of *Excellent* would be $1/5 = 0.2$ in the long tweet and $1/1 = 1$ in the short tweet, which is what we are expecting.

In variation (iii) the sentiment of a word is not affected by its repetition within one tweet. For example, consider two positive tweets such as *This burger was good good good good!!!* and *This burger was good!*. Although the **tweet** sounds more positive in the former case than in the latter, the sentiment of the **word** *good* itself will not be stronger. In other word, although a tweet might sound more positive, the words that it contains might not always be more positive. Experimental results show that the repetition of words within a tweet does not impact the sentiment score of words. That could be explained by the fact that tweet are so short that most of the words actually only appear once within a tweet.

We now illustrate how the normalization of pdf, ndf, and df used in variation (ii) could affect the sentiment score of some words. We compared the sentiment scores of three negative words and three positive words with and without the normalization feature. Table 5.4 shows that all three words *lame*, *insane*, and *scandal* have the same negative sentiment

score (-0.106) and all three words *excellent*, *fabulous*, and *congrats* have the same positive sentiment score (0.342) when the document frequencies of the words are not normalized. Contrariwise, when the sentiment score formula accounts for the length of the Tweet all three negative words have a different sentiment score and all three positives words have a different positive score. Not only the sentiment scores are different but they sometimes differ greatly, take for instance the word *scandal* which score changed from -0.106 to -0.021, or *excellent* which score changed from 0.342 to 0.175.

Another consequence is that some words no longer bear the same sentiment strength, e.g., they are no longer "equal" in terms of sentiment. For instance, *scandal* is no longer as negative as *lame*, or *insane*, but instead *lame* is almost 4 times more negative than *scandal*. Similarly, while *excellent* was as positive as *fabulous* without normalization, it is less positive with normalization.

This confirms our intuition that normalizing the document frequency of a words in Tweets impacts the sentiment scores and therefore impact the performances of the lexicon.

Inverse-Document Frequency in Tweets vs Amazon Products Reviews

The sentiment score formula used in this work is highly inspired by our preliminary work published in [32] on Amazon customer review data. One key difference, however, is that the Twitter sentiment score formula presented in this document does not use the information-theoretic scores (which employs the Term Frequency Inverse Document Frequency (TF-IDF) of the words).

Our intuition is that Tweets are by nature different from traditional text such as customer reviews, blog posts, or news articles. Indeed, Tweets are very short pieces of text

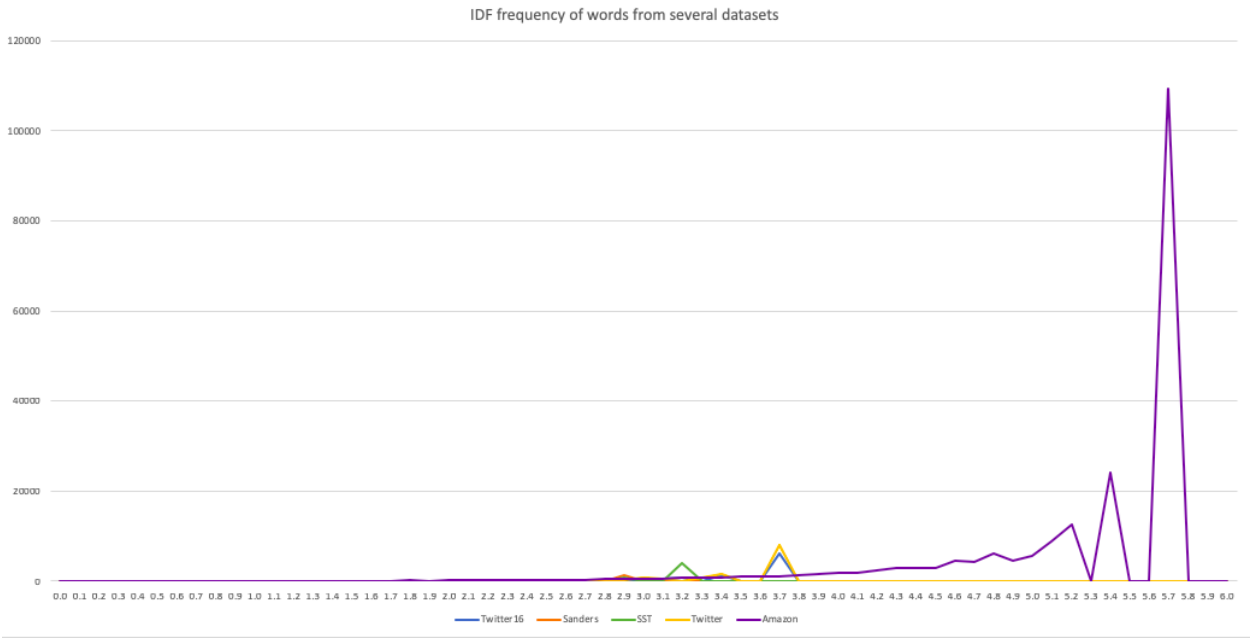

Figure 5.1: Frequency of IDF of words from Twitter vs words from Amazon

(280 characters maximum) that are often typed rapidly from a small portable device such as a smart phone or a tablet. They are prone to misspelling, slang words, and abbreviations. We therefore expect words within a tweet to have a very low term-frequency (TF). Likewise, we expect most words within a tweet to be very common words that carry little meaning, e.g., lol. That is, we expect most words in a tweet to have a low inverse-document frequency (IDF), which would add noise or very little value to our sentiment score calculation.

To validate our intuition we compute and plot the frequency of the IDF of words from several Twitter datasets and words from an Amazon products reviews dataset. The result is depicted in Figure 5.1.

We first notice from Figure 5.1 that despite having four different Twitter datasets, the IDF frequency are very similar, indeed the IDF frequencies are clustered around the same value. Furthermore, words from the Amazon dataset have a much higher IDF than words from the Twitter datasets. Indeed, most words belonging to the Amazon dataset have an

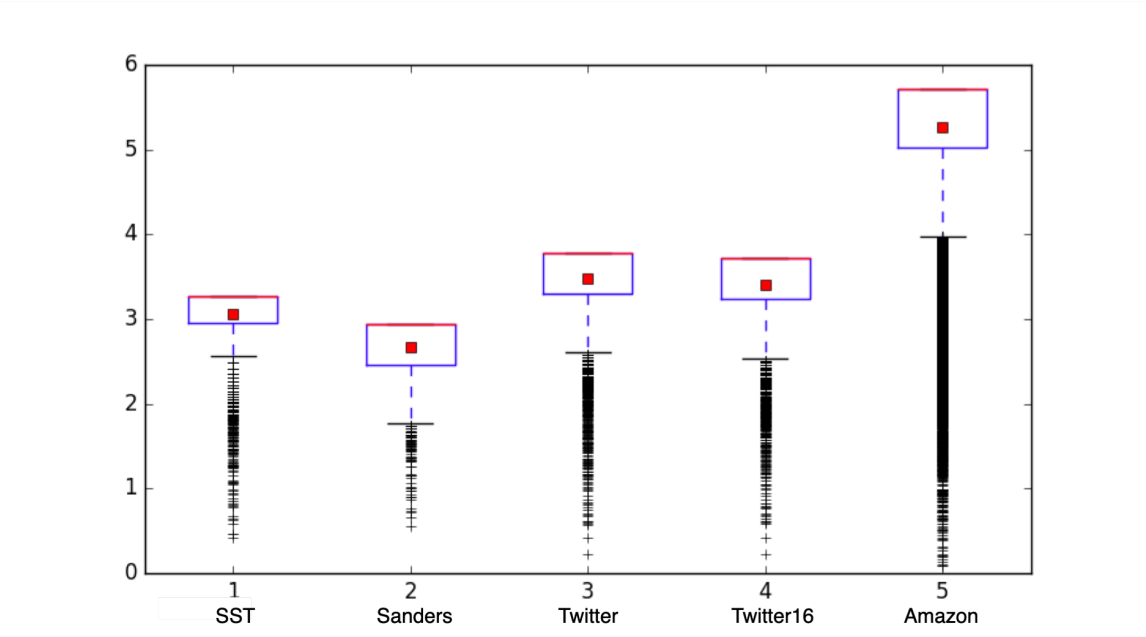

Figure 5.2: Boxplots of the IDF of words from Twitter vs words from Amazon

IDF between 5 and 5.8 whereas most words from the Twitter datasets have an IDF between 3 and 3.8. The lower the IDF of a word, the less important, and the more frequent the word is within the collection of documents. We also notice that the size of the vocabulary in Amazon is drastically bigger than that of the Twitter datasets. Thus, it appears that a common feature among Twitter datasets is having a small vocabulary that is composed of frequent and common words. These results confirm our intuition about the nature of Tweets.

We further analyze the IDF distribution of the words from the Amazon dataset and the distribution of the words from the Twitter datasets.

Figure 5.2 depicts the boxplots of the IDF of words from four Twitter datasets and one Amazon dataset. Boxplots are a useful tool that provides information on the distribution of the data. We notice that all five datasets are left-skewed, that is, majority of the words

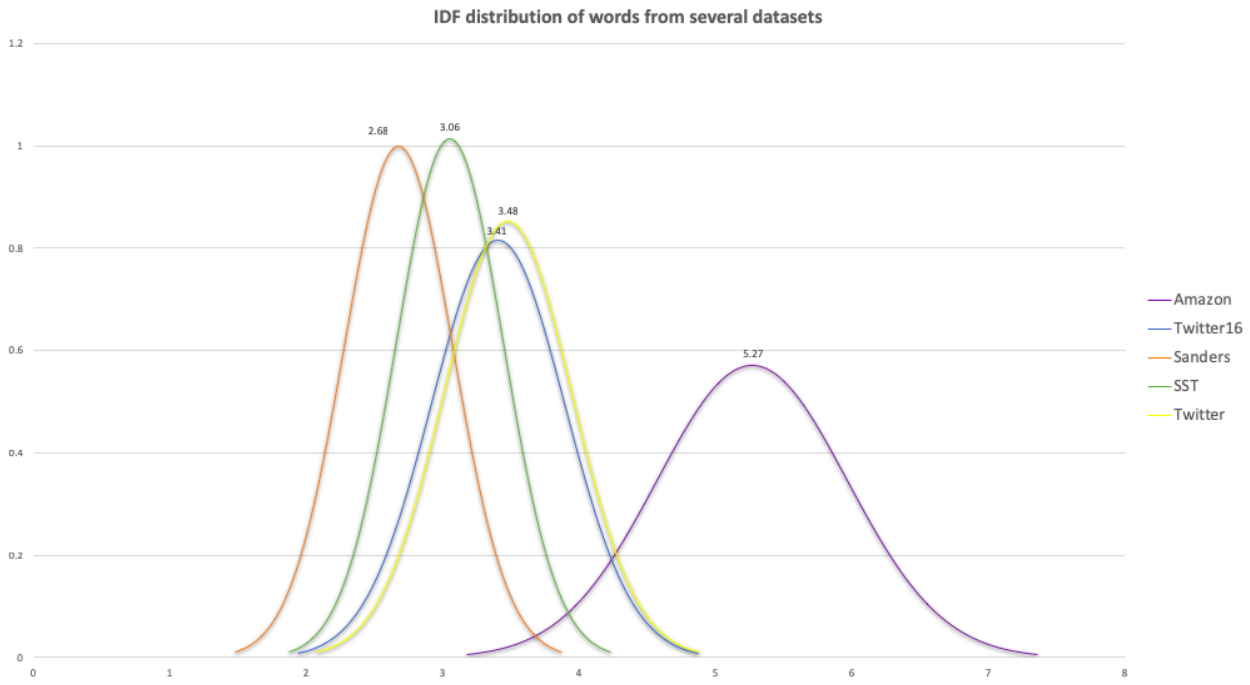

Figure 5.3: IDF distribution of words from Twitter vs words from Amazon fitted into a normal distribution

have an IDF that is close to the highest IDF of the corpus. Consequently, the median (represented by a red line on the boxplots) is higher than the average and is also very close to the maximum IDF. These observations indicate that majority of the words have a similar IDF, and that it is close to the maximum IDF of the dataset. Although the distribution of the IDF of words is similar in Twitter datasets and the Amazon dataset, the variability of the IDF of words is different. Indeed, the boxplots show that the Twitter datasets have a smaller interquartile range (IQR) than that of the Amazon dataset, which indicates a lesser variability of the IDF in the Twitter datasets.

Finally, we study the mean and variance of the IDF values for all five datasets. To better visualize how different they are from Twitter and Amazon, we fit the IDF distributions into a Gaussian distribution as showed on Figure 5.3. From the shape of the Gaussian

curves, it appears that all four Twitter IDF distributions have a very similar mean and variance, which differ greatly from the mean and variance of the IDF distribution of Amazon. The shape of the Gaussian curves also indicates that the IDF of words in Twitter have a very low variance and a low mean while the IDF of the words from Amazon have a high variance and a high mean. More specifically, the IDF of the words from the SST dataset have a mean of 3.05 and a variance of 0.15, the IDF of the words from the Sanders dataset have a mean of 2.68 and a variance of 0.16, the IDF of the words from the Twitter dataset have a mean of 3.48 and a variance of 0.22, the IDF of the words from the Twitter2016 dataset have a mean of 3.41 and a variance of 0.24, and the the IDF of the words from the Amazon dataset have a mean of 5.27 and a variance of 0.48.

The variance is a measure of how spread the numbers are around their mean value. Therefore, a low mean IDF combined with a low variance indicates that (1) most words in Twitter are not rare within the collection, and (2) most words in Twitter appear in the same number of Tweets. On the contrary, a high mean IDF combined with a high variance indicates that not only words in the Amazon dataset are not so common within the collection, but they appear in a wide range of frequency, e.g., many words appear in few documents and many words appear in many documents.

These results confirm our intuition that the inverse document frequency (IDF) of words would not contribute much to estimate and capture the sentiment of words from Twitter data.

Table 5.5: Performances of binary BOW and tf-idf BOW models on the sentiment quantification task

		Binary BOW model		tf-idf BOW model	
Feature	BOW	x	x	x	x
	i		x		x
	j		x		x
SemEval2016	KLD	0.126	0.107	0.126	**0.096**
	AE	0.144	0.140	0.145	**0.136**
	RAE	**1.321**	1.421	1.397	1.538
SemEval2017	KLD	0.286	0.215	0.274	**0.192**
	AE	0.289	0.247	0.280	**0.235**
	RAE	3.954	3.413	3.891	**3.203**
Average	KLD	0.206	0.161	0.200	**0.144**
	AE	0.216	0.194	0.213	**0.185**
	RAE	2.638	2.417	2.644	**2.371**

5.1.2 Representing Tweets as a Feature Vector

5.1.2.1 Results

As explained in Section 3, our feature vectors are based on the bag-of-word (BOW) model, therefore it is necessary to first assess which of the two BOW approaches perform best. We compare both BOW feature vectors without including any sentiment features, e.g., features derived from our lexicons. We report the results in Table 5.5. The reported KLD, AE, and RAE are the macro averaged metrics of each topics.

We study the effect of using the tf-idf instead of the binary model by comparing the average KLD of both models. Results from Table 5.5 indicate that for both feature vectors the tf-idf model outperforms the binary model. Indeed, the binary model has an average KLD of 0.183 whereas the average KLD of the tf-idf model is 0.172. The aforementioned results valid our intuition that using the tf-idf in place of a binary value for the bag-of-word feature improves the classification of the SVM machine.

We then focus on the effect of the feature i and feature j to the vector. We com-

pare the average KLD yielded by both feature vectors from the binary BOW model. We do likewise with the tf-idf BOW model. Results clearly show the effect of i and j on the performances of the SVM machine. Adding i and j to the BOW feature vector yielded a decrease of the average KLD (which reflects an improvement) of 0.045 (from 0.206 to 0.161) for the binary model and a decrease of the average KLD of 0.056 (from 0.200 to 0.144) for the tf-idf model. This improvement is considerable. Clearly, knowing the length of the tweet as well as the number of word that our dictionary covers provides helpful information in the quantifying task.

We therefore use the tf-idf BOW model as a base for our feature vector. Note that none of these features make use of the sentiment lexicon from Goal 1, so they do not provide information on the sentiment reflected by the Tweet.

Next, we study the effect of each sentiment feature derived from the single score lexicon and report our findings in Table 5.6. We separately study the effect of each sentiment feature derived from the paired score lexicon and sum up the results in Table 5.7. For both tables the reported KLD, AE, and RAE are macro the macro-averaged.

Table 5.6: Contribution of each sentiment feature extracted from the single score lexicon

univariate SVM single score		Features							
tf-idf BOW		x	x	x	x	x	x	x	x
i		x	x	x	x	x	x	x	x
j		x	x	x	x	x	x	x	x
sum			x	x	x	x	x		
max				x	x	x	x	x	
min				x	x	x	x	x	
avg					x	x			
nb pos						x	x	x	x
nb neg						x	x	x	x
SemEval2016	KLD	0.096	0.099	0.100	**0.069**	0.094	0.087	0.090	0.090
	AE	0.136	0.137	0.138	**0.118**	0.132	0.126	0.130	0.129
	RAE	1.538	1.278	1.278	**1.210**	1.269	1.247	1.369	1.353
SemEval2017	KLD	0.192	0.179	0.176	0.363	0.174	**0.173**	0.182	0.187
	AE	0.235	0.221	0.219	0.326	**0.216**	0.217	0.222	0.225
	RAE	3.203	3.016	2.978	4.355	**2.972**	2.985	3.091	3.140
Average	KLD	0.144	0.139	0.138	0.216	0.134	**0.130**	0.136	0.139
	AE	0.185	0.179	0.178	0.222	0.174	**0.172**	0.176	0.177
	RAE	2.371	2.147	2.128	2.783	2.121	**2.116**	2.230	2.247

Table 5.7: Contribution of each sentiment feature extracted from the paired score lexicon

univariate SVM paired score		Features							
tf-idf BOW		x	x	x	x	x	x	x	x
i		x	x	x	x	x	x	x	x
j		x	x	x	x	x	x	x	x
max pos			x				x	x	x
min pos			x				x	x	x
max neg				x			x	x	x
min neg				x			x	x	x
sum pos					x				
sum neg					x				
sum								x	
avg pos						x			
avg neg						x			
ratio									x
SemEval2016	KLD	0.097	0.096	0.097	0.100	0.097	0.096	0.095	**0.090**
	AE	0.139	0.136	0.139	0.134	0.139	0.136	0.131	**0.130**
	RAE	1.731	1.671	1.729	1.336	1.728	1.669	**1.298**	1.378
SemEval2017	KLD	0.185	0.181	0.183	0.186	0.184	0.181	0.207	**0.138**
	AE	0.228	0.225	0.228	0.224	0.228	0.225	0.242	**0.188**
	RAE	3.175	3.090	3.162	3.085	3.167	3.105	3.324	**2.559**
Average	KLD	0.141	0.138	0.140	0.143	0.141	0.138	0.151	**0.114**
	AE	0.184	0.181	0.183	0.179	0.183	0.180	0.187	**0.159**
	RAE	2.453	2.380	2.445	2.210	2.448	2.387	2.311	**1.969**

Table 5.6 shows that the base feature vector has an average KLD of 0.144. The best performance is achieved by a feature vector composed of the following features: tf-idf BOW, i, j, sum, max, min, nb pos, and nb neg, with an average KLD of 0.130 which is a decrease of 0.014 from the base feature vector. The worst performance is achieved by a feature vector having the following features: tf-idf BOW, i, j, sum, max, min, avg and yielded an average KLD of 0.216, which is a worse performance than the base vector. It is interesting to note that all feature vectors performed much better on the SemEval2016 dataset than the SemEval2017 dataset. Indeed, the average KLD for the SemEval2016 is 0.091 while it is 0.203 on the SemEval2017 dataset.

Table 5.7 shows that the base feature vector has an average KLD of 0.141. The best performance is achieved by a feature vector composed of the following features: tf-idf BOW, i, j, max pos, min pos, max neg, min neg, and ratio, with an average KLD of 0.114 which is a decrease of 0.027 from the base feature vector. The worst performance is achieved by a feature vector having the following features: tf-idf BOW, i, j, max pos, min pos, max neg, min neg, and sum, and yielded an average KLD of 0.151, which is a worse performance than the base vector. Here again, all feature vectors performed better on the SemEval2016 dataset than they did on the SemEval2017 dataset, with an average KLD of 0.096 against 0.162 respectively.

These results show that adding sentiment features derived from a sentiment lexicon greatly improve the performances of the quantification task. Finally, we compare the best feature vector that uses the single score lexicon with the best feature vector that uses the paired score lexicon. The feature vector that uses the paired score sentiment lexicon outperformed the feature vector that uses the single score lexicon, with an average KLD of 0.130 against 0.114 respectively and represents a decrease of 0.016. The feature vector that uses

the single score lexicon performed slightly better on the SemEval2016 dataset with a KLD of 0.087 against 0.090 for the feature vector that uses the paired score lexicon. However, the feature vector that uses the paired score lexicon outperformed the feature vector that uses the single score lexicon on the SemEval2017 dataset, with a KLD of 0.138 against 0.173 respectively.

Our experimental results show that although the sentiment features derived from the paired score lexicon perform as well as the sentiment features derived from the single score lexicon on one dataset, they clearly perform better on the second dataset. Therefore, from this point forward we will represent Tweets in the Vector Space Model with a feature vector that uses sentiment features derived from the paired score lexicon, specifically, the following features will be used: tf-idf BOW, i, j, max pos, min pos, max neg, min neg, and ratio.

From this experiment, we conclude that the sentiment of a Tweet is better represented through features derived from a lexicon that models the positivity and the negativity of words separately.

5.1.2.2 Discussions

Single Score Lexicon vs Paired Score Lexicon

Recall from Section 3.1.1, equation (1) that the sentiment score of a word is composed of its positivity score and its negativity score. The single score lexicon is the difference between these two quantities and therefore words have a dimension of 1, while the paired scored lexicon make use of both quantities and therefore words have a dimension of 2. Our intuition is that some word can have the same positivity but different negativity, while some

Table 5.8: Sample words having different single score with either an equal positivity or equal negativity

Word	single score	positivity	negativity
reason	-0.0261	0.1543	0.1804
support	0.0959	0.1544	0.0584
boo	0.0641	0.2835	0.2193
poor	-0.2196	0.0000	0.2196
knows	0.3523	0.4444	0.0921
redeemed	-0.0921	0.0000	0.0921

words can have a same negativity but different positivity. In such cases the single sentiment will have different values. However, if two words have a different positivity and a different negativity the difference could also yield two different values. The single score lexicon cannot distinguish between both cases while the paired score lexicon will.

Furthermore, the single scores capture information about the **difference** between the positivity and negativity of the words, while the paired scores in fact quantify **how much** positive and **how much** negative a word is. Consequently, every sentiment features derived from the single scores are actually related to the **difference** between the positivity and the negativity (for instance, the feature max actually reflects which word has the highest difference), whereas sentiment features derived from the paired scores are related to the **quantity** of the positivity and negativity of the words (here, max pos or max neg actually reflects what the maximum quantity of positivity or negativity is in the Tweet). This would explain why the sentiment features derived from the paired score perform much better than the features derived from the single score. Table 5.8 shows a few sample words extracted from one of our lexicon that illustrates our intuition:

From Table 5.8 we can see that the words *reason* and *support* have a different single score, reason is deemed positive while support is deemed negative. Yet, they have an equal positivity, which means that the difference of sentiment for these two words is due to the

negative aspect that they reflect. Note that the single sore could not reveal this information. Similarly, the words *knows* and *redeemed* have different sentiment scores, in fact, one is deemed positive while the other is deemed negative. However, here again the single scores will not provide information as to what cause this difference. The paired scores of these two words reveal that they have a same negativity but a different positivity, specifically, *knows* has a much higher positivity than *redeemed*.

To have a better intuition of how the positivity and negativity of words relates, we plotted the positivity versus negativity of the words extracted from one of our paired score lexicon onto a 2-dimensional plane. The scatter plot is reported in Figure 5.4.

Figure 5.4 shows that most positive words either have a moderate positivity with a very small negativity or a very high positivity with a small negativity. Likewise, most negative words either have a moderate negativity with a very low positivity or a very high negativity with a very small negativity. Neutral words seem to have a moderate positivity with a moderate negativity. The case of having neutral words with a high positivity and a high negativity seems to be nonexistent.

Paired Score Lexicon Feature Vector

As we previously said, the feature vector that yielded the best quantification is the feature vector that uses sentiment features derived from the paired score lexicon and has the following features: tf-idf BOW, i, j, max pos, min pos, max neg, min neg, and ratio. In our experiments we did not actually evaluate the role of each individual sentiment feature but rather what is the effect of **adding** this feature in combination with **other features**. In other words, we do not evaluate the absolute effect of a sentiment feature but rather its

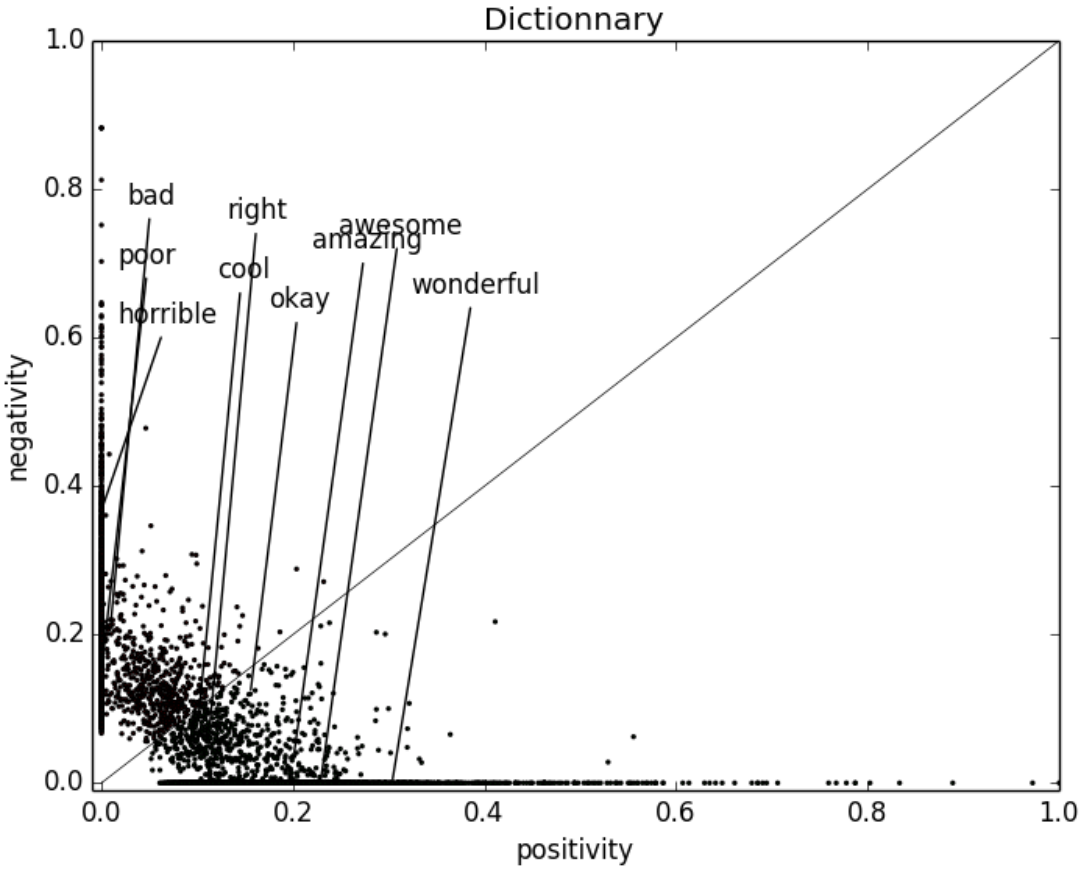

Figure 5.4: Words represented on a 2-dimension plane

relative effect when combined with other features.

From Table 5.7, adding the features max pos and min pos, or max neg and min neg very slightly improve the performances of the quantifier, since the KLD only decreases of 0.03 and 0.01 respectively, when compared to the base feature vector. Similarly, adding either the sum pos and sum neg, or the average pos and average neg almost has no effect on the quantification task. When adding all four features max pos, min pos, max neg and min neg to the base vector, we notice a very small improvement of 0.03, so we choose to continue with this feature vector. Adding the sum feature to the aforementioned feature vector negatively impacts the performances with an increase of KLD of 0.13, meaning that

the sum combined to this vector actually adds more noise. Finally, adding the ratio feature to the vector drastically improves the performance with a decrease of KLD of 0.027.

The i and j features tells us how many words from the Tweet we have sentiment knowledge about and therefore how many words actually contribute to the sentiment. The max pos, max neg, min pos and min neg features provide further information about the spread or range of the positivity and negativity contained in the Tweet, e.g., it tells us how much positivity and how much negativity is in the Tweet. Finally, the ratio feature (which from our results seem to play an important role for the quantification task and which is computed from the average positivity and average negativity) provides information on how much stronger the positivity for each word on average is as compared to their negativity. We will use this feature vector with the univariate SVM as our baseline for Goal 2.

5.2 Results of Goal 2: Sentiment Quantification

5.2.1 Results

We now compare the baseline univariate SVM from Goal 1 to several multivariate SVM approaches described in Section 3 and report our findings in Table 5.9. The reported KLD, AE, and RAE are the macro-averaged.

Table 5.9 shows that all but one multivariate SVM outperform the univariate SVM. Precisely, the multivariate SVM(perf) (which is the original multivariate SVM) did not perform better than our baseline univariate SVM. However, this is not surprising since SVM(perf) optimizes the error rate which is not a statistical distance and therefore must not be suitable to perform sentiment quantification. Our experimental results show that the

best performances are achieved with our SVM(HD) which outperforms all other multivariate SVM with an average KLD of 0.007, against 0.041 for our baseline. The worst performances are achieved by the SVM(perf) with an average KLD of 0.063. SVM(HD) is closely followed by SVM(TVD) which has an average KLD of 0.008. SVM(KLD) and SVM(RAD) both performed well with an average KLD of 0.022, followed by SVM(BD) with an average KLD of 0.024 and SVM(JSD) with an average KLD of 0.028. Clearly, a multivariate SVM that optimizes a statistical distance is best suited for sentiment quantification.

We further compare our best approach (SVM(HD)) to results from various works that uses the same datasets that we used. We therefore compare our SVM(HD) with the results from [1] who are the authors of SVM(KLD) on the following datasets: SST, Sanders, SemEval2013 Task A, SemEval2014 Task A, and SemEval 2015 task A. Furthermore, to assess the effectiveness of our feature vector rather than the effectiveness of the SVM, we compare the results obtained from our feature vectors using their SVM(KLD). We report our results in Table 5.10.

Table 5.9: Results of the quantification using univariate SVM vs multivariate SVM

	Metrics	univariate SVM	SVM(perf)	SVM(KLD)	SVM(HD)	SVM(BD)	SVM(JSD)	SVM(TVD)	SVM(RAD)
SST	KLD	0.031	0.011	0.036	0.005	0.044	0.046	**0.000**	0.030
	AE	0.124	0.148	0.266	0.100	0.295	0.301	0.028	0.245
	RAE	0.254	0.149	0.268	0.101	0.296	0.303	0.029	0.246
Sanders	KLD	0.000	0.004	0.010	0.001	0.007	0.028	**0.000**	0.007
	AE	0.005	0.088	0.138	0.037	0.115	0.230	0.005	0.115
	RAE	0.010	0.088	0.138	0.037	0.115	0.231	0.005	0.115
SemEval 2013 task A	KLD	0.003	0.046	0.019	**0.000**	0.018	0.024	0.003	0.019
	AE	0.032	0.275	0.194	0.006	0.191	0.219	0.080	0.194
	RAE	0.081	0.290	0.204	0.006	0.201	0.230	0.084	0.204
SemEval 2014 task A	KLD	0.011	0.018	0.022	**0.001**	0.020	0.026	0.001	0.022
	AE	0.059	0.171	0.204	0.040	0.197	0.222	0.045	0.204
	RAE	0.208	0.191	0.228	0.045	0.221	0.249	0.050	0.228
SemEval 2015 Task A	KLD	0.017	0.041	0.032	**0.001**	0.030	0.036	0.002	0.031
	AE	0.085	0.260	0.251	0.047	0.244	0.267	0.058	0.248
	RAE	0.220	0.276	0.266	0.050	0.259	0.283	0.061	0.263
SemEval 2016 Task D	KLD	0.090	0.069	**0.010**	0.013	0.014	0.011	0.018	0.014
	AE	0.130	0.242	0.098	0.111	0.108	0.098	0.136	0.106
	RAE	1.378	0.266	0.111	0.125	0.121	0.111	0.156	0.119
SemEval 2017 Task D	KLD	0.138	0.254	**0.024**	0.028	0.034	0.025	0.031	0.033
	AE	0.188	0.577	0.171	0.182	0.207	0.176	0.192	0.203
	RAE	2.559	0.676	0.200	0.213	0.241	0.205	0.225	0.237
Average	KLD	0.041	0.063	0.022	**0.007**	0.024	0.028	0.008	0.022
	AE	0.089	0.252	0.189	**0.075**	0.194	0.216	0.078	0.188
	RAE	0.673	0.276	0.202	**0.082**	0.208	0.230	0.087	0.202

Table 5.10: Comparison of our SVM(HD) and SVM(KLD) to the SVM(KLD) of [1]

	Metrics	our SVM(KLD)	SVM(HD)	SVM(KLD) from Gao et al.
SST	KLD	0.036	**0.005**	0.011
	AE	0.266	0.100	0.041
	RAE	0.268	0.101	0.146
Sanders	KLD	0.010	**0.001**	**0.001**
	AE	0.138	0.037	0.013
	RAE	0.138	0.037	0.063
SemEval 2013 Task A	KLD	0.019	**0.000**	0.029
	AE	0.194	0.006	0.072
	RAE	0.204	0.006	0.172
SemEval 2014 Task A	KLD	0.022	**0.001**	0.033
	AE	0.204	0.040	0.084
	RAE	0.228	0.045	0.227
SemEval 2015 Task A	KLD	0.032	**0.001**	0.076
	AE	0.251	0.047	0.119
	RAE	0.266	0.050	0.379

We notice from Table 5.10 that our SVM(HD) outperform the SVM(KLD) from Go et al. on all datasets but the Sanders dataset, where both approaches perform equivalently. In addition, the feature vector that we use outperforms the feature vector used by Go et al. when used with the SVM(KLD). Indeed, our approach achieved an average KLD of 0.024 against 0.030 for their approach. These results show that not only SVM(HD) approach might be a better choice, but also that the feature vector that we use to represent a Tweet is more suitable in the sentiment quantification task.

We then compare the results of our SVM(HD) with that of Stojanovski et al. [38] and Mathieu Cliche [39] using the SemEval2016 Task D dataset and the SemEval2017 Task D dataset. The approach from Stojanovski et al. ranked first and achieved a KLD of 0.034 on the SemEval2016 Task D dataset, likewise, the approach from Mathieu Cliche ranked first and achieved 0.036 on the SemEval2017 task D dataset. We report our results in Table 5.11.

As shown in Table 5.11, our approach outperform both best approaches from SemEval

Table 5.11: Comparison of our SVM(HD) with the best approaches from SemEval 2016 and SemEval 2017

	Metrics	SVM(HD)	Stojanovski et al
SemEval	KLD	**0.013**	0.034
2016	AE	0.111	0.074
Task D	RAE	0.125	0.707
	Metrics	SVM(HD)	Mathieu Cliche
SemEval	KLD	**0.028**	0.036
2017	AE	0.182	0.080
Task D	RAE	0.213	0.598

2016 and SemEval 2017.

Our experimental results show that (1) our feature vector that uses sentiment features derived from our paired score lexicon is a strong model to represent Tweet sentiment in the Vector Space Model, and (2) the multivariate SVM machine that minimizes the Hellinger Distance is a very effective approach to the sentiment quantification problem.

5.2.2 Discussions

The following discussions analyze different perspectives of the performances achieved by the multivariate SVM so as to better understand how the statistical distances compare to each other.

5.2.2.1 Performances vs Size of Dataset

We first take a look at the relationship between the size of the dataset and the performances of the multivariate SVMs. Table 5.12 reports the average KLD achieved by each multivariate SVM as well as the size of each dataset.

From Table 5.12 we notice that TVD seems to outperform all other statistical distances on the two smallest datasets (namely Sanders and SST which have a size of 872 and

1831 respectively) and achieved a KLD of 0.0000 and 0.0004 respectively. This is closely followed by HD which yielded a KLD of 0.0007 and 0.0050.

On the dataset of size 5589 the best performance was achieved by KLD with a KLD of 0.0103 closely followed by JSD with 0.0111 and HD with a KLD of 0.0126.

The next three dataset (namely SemEval2013 Task A, SemEval2014 Task A, and SemEval2015 Task A) are actually the same but they were used with three different testing dataset. HD outperform all other statistical distances on all three with a respective KLD of 0.0007, 0.0000, and 0.0012. These results were closely followed by TVD which achieved a KLD of 0.0035, 0.0012, and 0.0018 respectively.

Finally, on the very large dataset which has a size of 10,551 KLD outperformed all other statistical distances.

Results from Table 5.12 does not allow us to conclude any clear relationship between the size of the training dataset and the performances of the multivariate SVM. However, regardless of the size, it seems that HD always performs almost as good as the best statistical distance, and is therefore much better on average. **Disclaimer:** These results should in any case be treated as a proof or an evidence, but rather as what they truly are: observations on our data.

The previous observations illuminate the effect of the size of the dataset on the performances of the multivariate SVM. We now focus on the effect of the size of the dataset on each statistical distance. To better visualize this, we plotted on Figure 5.5 the resulting KLD of each method versus the size of the training datasets.

From Figure 5.5 it seems that the performances of each individual statistical distance has no clear relationship with the size of the datasets, as shown by the very fluctuating curves. However, two distinct patterns seem to be apparent: The first pattern shows that

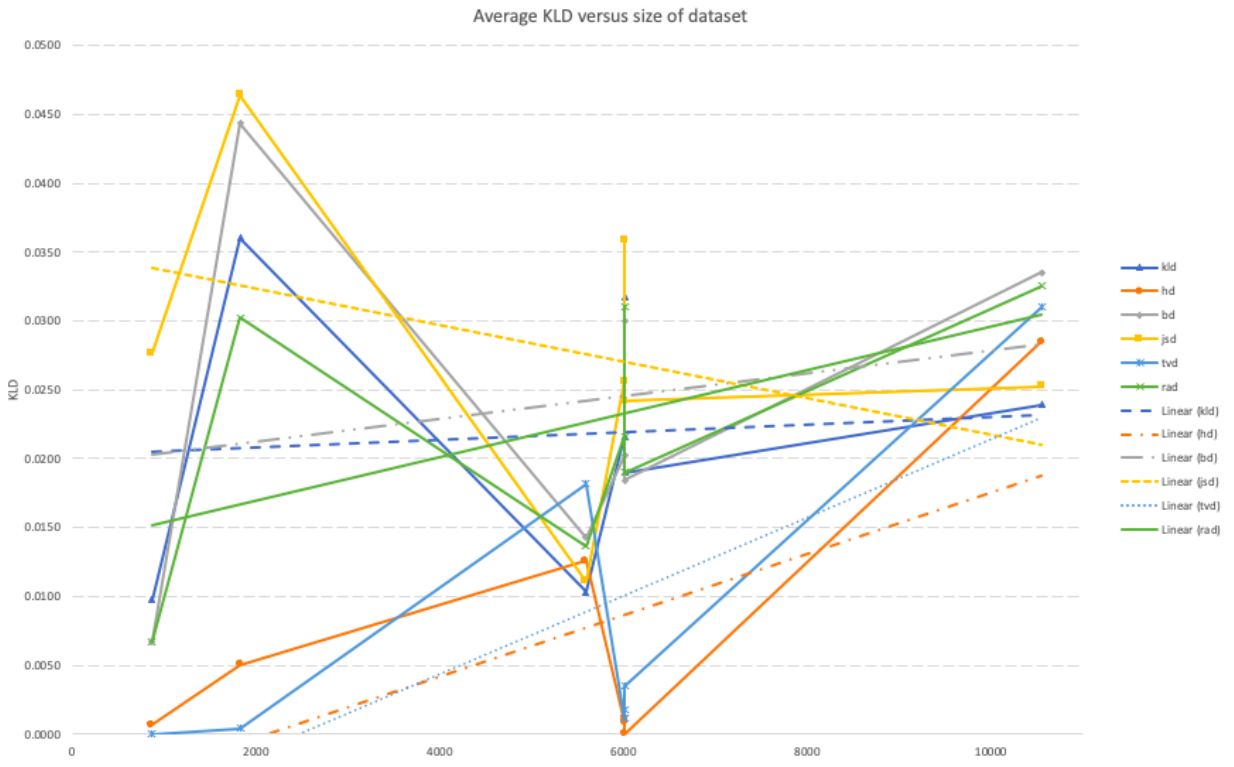

Figure 5.5: KLD of multivariate SVMs vs size of datasets

KLD, JSD, RAD, and BD follow a very similar behavior. Additionally, the linear trends (represented by the straight dashed lines on the graph) of KLD, BD and RAD seem to indicate that the performance of these 3 statistical distances slightly decreases as the dataset gets bigger whereas that of RAD suggests that it performs better. The second pattern shows that HD and TVD both have a strongly increasing linear trend meaning that their performances worsen as the dataset gets bigger.

Overall, our experimental results suggest that no clear relationship exists between the size of the training dataset and the performances of the multivariate SVM for any of the statistical distances that we explored. However, the Hellinger Distance on average outperforms all other statistical distances and seems to be robust regardless of the size of

the training dataset. We therefore suggest to minimize the Hellinger Distance with the multivariate SVM for the sentiment quantification task.

5.2.2.2 Performances vs Skewness of Data

We now investigate the effect of the skewness of the data on the performances of each statistical distance. Table 5.13 shows the skewness of each dataset along with the KLD achieved by all our approaches. The skewness will be expressed in percentage difference (positive percentage - negative percentage) where we arbitrarily consider a dataset to be skewed if the percentage difference is greater than 10%, and will consider the dataset balanced otherwise.

We will first analyze the effect of the skewness in the dataset on the different approaches, then we will study how the skewness impacts each statistical distance individually. From Table 5.13 we notice that KLD achieves the best performance for the balanced dataset Sanders while TVD achieves the best performance on the balanced dataset SST. However, the Hellinger Distance achieves a KLD of 0.0146 on average on the balanced datasets against 0.0168 and 0.0201 for KLD and TVD respectively.

Table 5.12: Effect of the size of the data on the quantification task

	size	SVM(KLD)	SVM(HD)	SVM(BD)	SVM(JSD)	SVM(TVD)	SVM(RAD)
Sanders	872	0.0097	0.0007	0.0067	0.0277	**0.0000**	0.0067
SST	1831	0.0360	0.0050	0.0443	0.0464	**0.0004**	0.0302
SemEval2016 Task D	5589	**0.0103**	0.0126	0.0143	0.0111	0.0182	0.0137
SemEval2013 Task A	6013	0.0241	**0.0007**	0.0229	0.0285	0.0021	0.0239
SemEval2014 Task A	6013	0.0190	**0.0000**	0.0184	0.0242	0.0035	0.0190
SemEval2015 Task A	6013	0.0318	**0.0012**	0.0300	0.0358	0.0018	0.0310
SemEval2017 Task D	10551	**0.0239**	0.0285	0.0335	0.0252	0.0310	0.0325
Average		0.0218	**0.0070**	0.0239	0.0280	0.0080	0.0221

Table 5.13: Effect of skewness on the quantification task

	pos %	neg %	% difference	skewness	KLD	HD	BD	JSD	TVD	RAD
Sander	47.93%	52.07%	4.14%	balanced	**0.0239**	0.0285	0.0335	0.0252	0.0310	0.0325
SST	54.01%	45.99%	8.02%	balanced	0.0097	0.0007	0.0067	0.0277	**0.0000**	0.0067
SemEval2013 Task A	70.10%	29.90%	40.20%	skewed	0.0190	**0.0000**	0.0184	0.0242	0.0035	0.0190
SemEval2014 Task A	70.10%	29.90%	40.20%	skewed	0.0318	**0.0012**	0.0300	0.0358	0.0018	0.0310
SemEval2015 Task A	70.10%	29.90%	40.20%	skewed	**0.0103**	0.0126	0.0143	0.0111	0.0182	0.0137
SemEval2017 Task D	77.83%	22.17%	55.66%	skewed	0.0216	**0.0009**	0.0202	0.0256	0.0012	0.0216
SemEval2016 Task D	80.73%	19.27%	61.46%	skewed	0.0360	0.0050	0.0443	0.0464	**0.0004**	0.0302
macro avg				balanced	0.0168	**0.0146**	0.0201	0.0264	0.0155	0.0196
macro avg				skewed	0.0237	**0.0039**	0.0254	0.0286	0.0050	0.0231

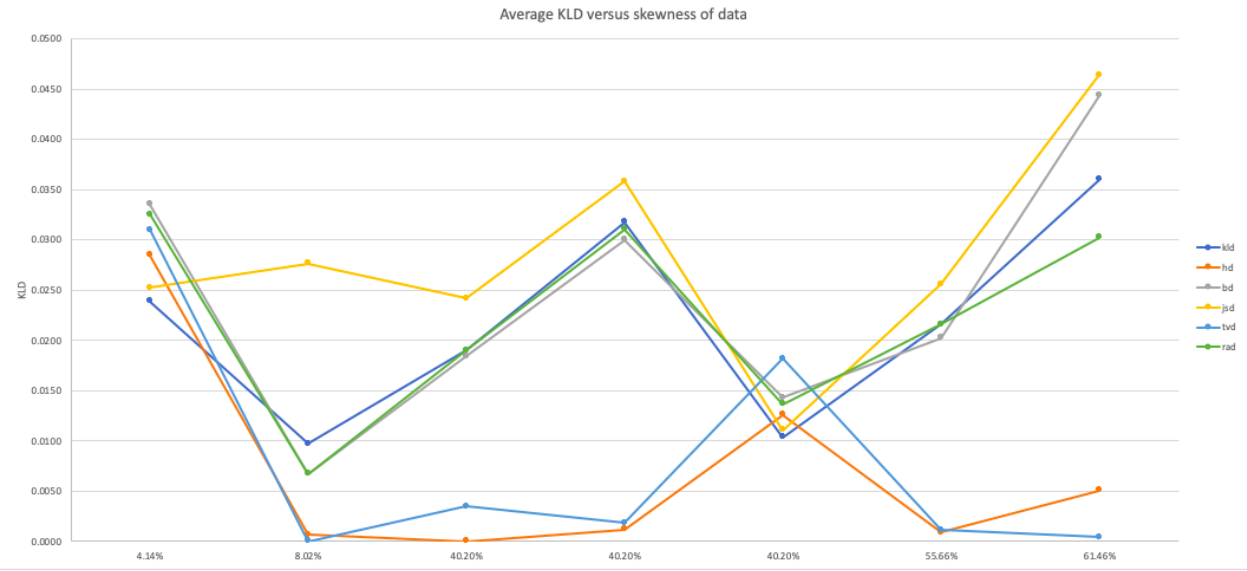

Figure 5.6: KLD of multivariate SVMs vs skewness of datasets

On the 5 skewed datasets, the Hellinger Distance achieved the best results on 3 while KLD and TVD both performed best on 1.

On average, the Hellinger Distance outperforms all other metrics on both the balanced datasets and the skewed datasets, with an average KLD of 0.0146 and 0.0039 respectively.

We will now study how the skewness of the data affect each statistical distance individually. Figure 5.6 shows the plot of the KLD of each approach versus the skewness of the training dataset.

Figure 5.6 does not reveal any clear relationships between the skewness of the data and the performance of each statistical distance. However, based on the high fluctuations of the KLD, RAD, JSD, and BD curves these four metrics seem to be sensitive to the skewness. In contrast, the HD and TVD seem to yield very good performance regardless of the skewness of the data.

Overall, our experimental results indicate that the Hellinger Distance and the Total Variation Distance are both very robust to the skewness of the data as compared to the other metrics, which is in accordance with what Cieslak and Chawla observed as well[66]. Furthermore, the Hellinger Distance outperforms the Total Variation Distance on both balanced and skewed datasets.

5.2.2.3 Observing the Behavior of Statistical Distances

Finally, we aim at trying to understand why the Hellinger Distance and Total Variation Distance perform much better than the Kullback-Leibler Divergence, the Resistor-Average Distance, the Jensen-Shannon Divergence, and the Bhattacharyya Distance. The underlying mathematical definitions and properties of each of these metrics must directly impact how they perform. Specifically, it must impact how they behave when used to measure the distance between two probability distributions in the Vector Space Model.

Our first intuition is that the behavior of these metrics in function of the inputs p and $\hat{p}$ (the true probability distribution and the estimated probability distribution respectively) greatly differs as p and $\hat{p}$ become further apart. In other words, we suspect that the rate of growth of these statistical distances is different and could play a role in their performances.

We study the rate of growth of each statistical function with the following experiment: we measure div(p, $\hat{p}$) in function of MAE(p, $\hat{p}$), where (div) is either HD, TVD, KLD, JSD, RAD, or BD and MAE is the Mean Absolute Error between p and $\hat{p}$ and is defined as follows:

$$MAE(p, \hat{p}) = \frac{\sum x = 1^2 |\hat{p}_i - p_i|}{2}$$

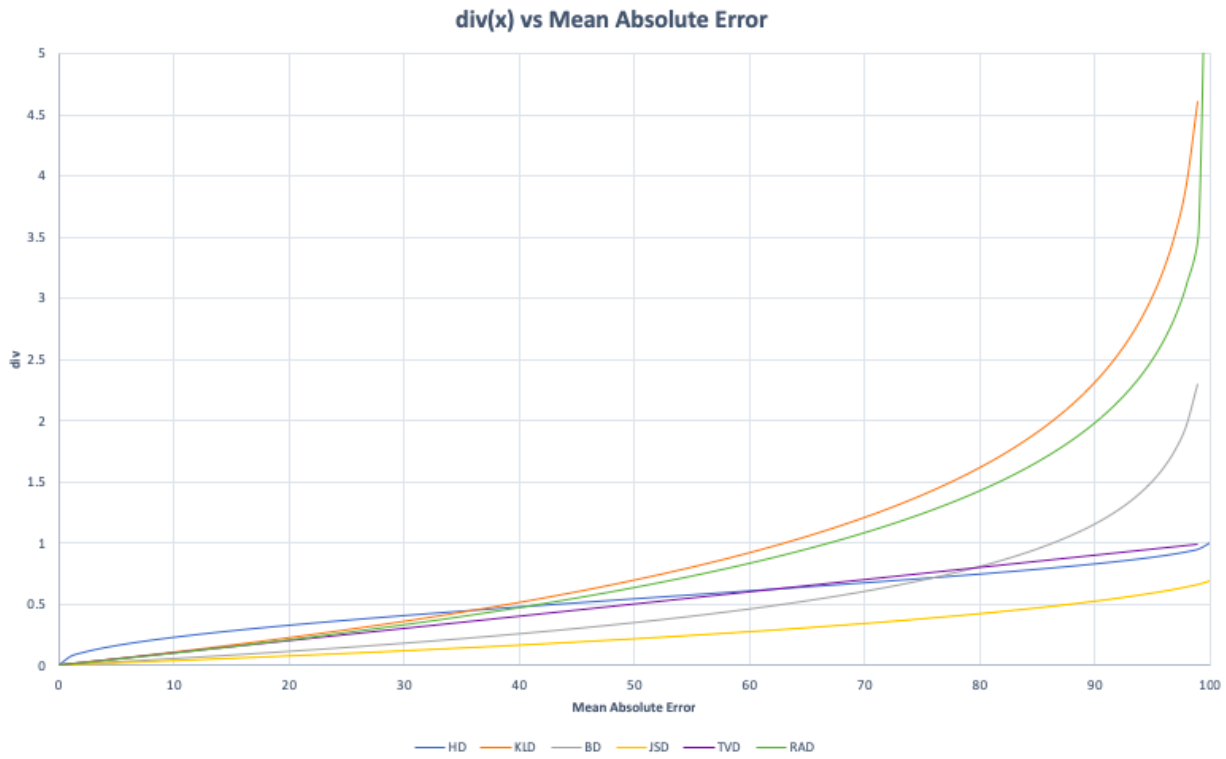

Figure 5.7: fdiv(x) versus MAE

We will compare statistical distances for each value of MAE within the range $[0, 100]$ with an increase of 1 in between each run. That is, when $MAE(p, \hat{p}) = 0$ there is a perfect prediction, e.g., $p = \hat{p}$, whereas when $MAE(p, \hat{p}) = 100$ p and $\hat{p}$ are opposite. i.e., $p = [1.0, 0.0]$ and $\hat{p} = [0.0, 1.0]$. We set the true probability to $p = [1.0, 0.0]$ throughout the entire experiment and we only update the predicted probability $\hat{p}$ at each run. Figure 5.7 shows the resulting plot of the experiment.

From Figure 5.7 we can clearly observe the difference in the behavior of the statistical distances. KLD, BD, and RAD have an exponential growth meaning that as the Mean Absolute Error gets larger ($\hat{p}$ gets further away from the truth p), the measure of the distance between the true probability distribution and the estimated probability distribution becomes exponentially larger. While JSD seems to have a polynomial growth, HD seem to have a

logarithmic growth and TVD has a linear growth. As a consequence, as the $MAE(p, \hat{p})$ gets bigger, JSD, HD, and TVD will still evaluate the distance to be larger, but not drastically.

From an optimization perspective, these observations mean that the feedback returned by KLD, BD, and RAD when $\hat{p}$ is far from the truth is much bigger than the feedback returned when $\hat{p}$ is close to p. We suspect this behavior to negatively impact the SVM machine as it could make the classification more difficult.

Our second intuition is that unlike KLD, BD, JSD, and RAD, both the Hellinger Distance and the Total Variation Distance are true distance metrics, that means that they perfectly capture the notion of distance within a space, since they satisfy all three axioms. While property (i) is satisfied by all six metrics that we compared, property (ii) and (iii) are not always satisfied.

The symmetrical property (ii) states that the distance between p and q must be equal to the distance between q and p, e.i., $d(p, q) = d(q, p)$. We notice that both the HD and the TVD are symmetrical distances. However, the JSD and RAD are both symmetrized version of KLD and yet provide no improvement upon KLD. Hence, we cannot positively assert that the symmetrical property plays a key role in evaluating statistical distances.

The triangular inequality property (iii) defined as follows: $d(x, y) + d(y, z) \geq d(x, z)$ states that for any $x, y, z \in \chi$, the distance $d(y, z)$ is the shortest distance from y to z in the space χ. If this property is not met then the distance measured by a function d between two points is not guaranteed to be the shortest in that space.

The triangular inequality is one key difference between a true distance metric such as the Hellinger Distance or the Total Variation Distance and a pseudo-distance measure (also named divergence measure) such as KLD, RAD, JSD, or BD. When optimizing a divergence measure through SVM we are minimizing a distance that is not guaranteed to be the minimal

distance between both points.

Therefore, we believe that optimizing a true distance metric through a multivariate SVM is more effective for the sentiment quantification task.

5.3 Results of Goal 3: Sentiment Shift Detection

5.3.1 Results

We will compare the effectiveness of all of our sentiment shift detection algorithms to each other as well as against the baseline in order to see which one achieves the best performances. Detecting a sentiment shift is the primary objective of the algorithms, it is not the only criteria to evaluate nonetheless. Indeed, an algorithm that detects a sentiment shift on every new data point would then be 100% accurate whilst it would obviously provide poor results. The approaches must therefore answer two criteria to be considered effective:

1. They should accurately detect a sentiment shift when one has occurred
2. They should accurately not detect a sentiment shift when none has occurred

We will first analyze the results on the micro level, that is, when for each approach the metrics are computed per window size and per signal. Then we will present the results of the macro analysis which computes the metrics for each approach with no knowledge about the shape of the signals nor the size of the window, in other words, all data points are combined and analyzed together as if they were one unique signal. As explained in Section 4, we will evaluate the effectiveness through the accuracy, fall-out (or FPR), miss rate (or FNR), and balanced accuracy.

5.3.1.1 Micro results

Table 5.14 shows the average of the accuracy while Table 5.15 shows the average balanced accuracy achieved across all 10 signals. The last column represents the average across all window sizes. Furthermore, Table 5.16 shows the average fall-out and Table 5.17 shows the average miss rate achieved across all 10 signals for each window size ranging from 3 to 10. The last column represents the average across all window sizes.

Accuracy and Balanced accuracy

We notice from Table 5.14 that the highest accuracy, on average, is achieved by the Dist p2av* approach with 94.44% accuracy while the lowest is achieved by the baseline with an average accuracy of 44.87%. Although we could conclude that the approach Dist p2av* is best, these results are not reliable since our dataset is extremely unbalanced with approximately 95% of the points belonging to one class (not carrying a sentiment shift) and 5% belonging to the other class (carrying a sentiment shift). The accuracy therefore is biased towards the dominant class. Since the balanced accuracy normalizes the accuracy obtained in both classes, it is a more reliable metric to use. The balanced accuracy from Table 5.15 indicates that the approach Dist p2p is in fact a better approach with 88.32% balanced accuracy, closely followed by Dist p2p* with 87.34%. The approach Dist p2av* achieved 70.95% balanced accuracy. These results show that Dist p2av* was very accurate in the dominant class and not so accurate in the minute class, which does not meet both necessary requirements. The lowest balanced accuracy was achieved by Sig prop with 68.35%. In addition, the approaches Dist p2p and Dist p2p* introduced in this work outperform the baseline which achieved 71.40% balanced accuracy.

The balanced accuracy gives a general feeling as to which approach performs well in both classes. Yet, a deeper analysis is necessary to ensure that both criteria mentioned above are met. An analysis of the fall-out and miss rate is thus required.

Table 5.14: Micro average Accuracy achieved across all 10 signals

	w3	w4	w5	w6	w7	w8	w9	w10	average
baseline	38.51%	43.75%	44.00%	42.14%	46.15%	50.00%	46.36%	48.00%	44.87%
Sig Prop	65.29%	83.75%	93.33%	96.43%	96.92%	97.50%	95.45%	95.00%	90.46%
Dist p2p	72.35%	75.63%	75.33%	77.14%	80.00%	80.83%	75.45%	85.00%	77.72%
Dist p2p*	78.24%	89.38%	92.00%	95.00%	96.92%	97.50%	89.09%	97.00%	91.89%
Dist p2v	81.18%	70.00%	66.00%	69.29%	73.08%	79.17%	70.00%	78.00%	73.34%
Dist p2v*	92.35%	88.13%	94.00%	95.71%	95.38%	96.67%	95.45%	95.00%	94.09%
Dist p2av	84.12%	79.38%	79.33%	79.29%	80.00%	77.50%	77.27%	79.00%	79.49%
Dist p2av*	92.94%	87.50%	96.00%	95.00%	96.92%	96.67%	95.45%	95.00%	94.44%

Table 5.15: Micro average Balanced Accuracy achieved across all 10 signals

	w3	w4	w5	w6	w7	w8	w9	w10	average
baseline	67.65%	70.40%	70.51%	70.24%	72.37%	74.43%	72.41%	73.22%	71.40%
Sig prop	72.73%	82.14%	78.00%	74.29%	69.62%	70.00%	50.00%	50.00%	68.35%
Dist p2p	85.31%	87.23%	86.90%	88.21%	89.71%	90.08%	86.95%	92.17%	88.32%
Dist p2p*	88.41%	94.33%	95.59%	97.39%	88.85%	89.17%	75.00%	70.00%	87.34%
Dist p2v	89.84%	84.29%	81.77%	84.03%	86.03%	89.13%	84.00%	78.89%	84.75%
Dist p2v*	77.27%	84.33%	78.23%	73.93%	68.85%	69.58%	50.00%	50.00%	69.02%
Dist p2av	82.18%	89.06%	89.00%	89.28%	89.65%	88.30%	78.59%	79.61%	85.71%
Dist p2av*	86.98%	74.64%	88.67%	78.13%	69.62%	69.58%	50.00%	50.00%	70.95%

Fall-out and Miss rate

The fall-out and miss rate both are metrics used to analyze errors, therefore the lower the values the fewer errors and the better the approach. The fall-out is also called False Positive Rate and therefore relates to criteria 2 mentioned earlier while the miss rate is equivalent to the False Negative Rate and therefore relates to criteria 2.

Results from Table 5.16 show that the lowest fall-out achieved is Dist p2av* with 3.09% closely followed by Dist p2v* with 3.20%, meaning that these two approaches are good at not detecting a shift when none has occurred. The highest fall-out was achieved by the baseline with 57.19%, meaning that this approach often detects a sentiment shift when none has happened. The second highest fall-out is our approach Dist p2v with 28.01%.

Results from Table 5.17 indicate that the lowest miss rate obtained are by the baseline and Dist p2p, with a miss rate of 0.00%, meaning that these two approaches have never failed to detect a sentiment shift when one occurred. These results are closely followed by Dist p2v with a miss rate of 2.50%. The highest miss rate was obtained by Dist p2v* with 58.75%, closely followed by Sig prop with 56.25%, indicating that both of these approaches highly failed to detect a shift that actually occurred.

These results give us a first feeling as to which approaches perform well in the detection task, but are insufficient to draw a conclusion. Indeed, the balanced accuracy indicates that Dist p2p and Dist p2p* are the best in detecting sentiment shift, the fall-out indicates that Dist p2v* and Dist p2av* are best, and the miss rate tells us that the baseline and Dist p2p are more appropriate.

Table 5.16: Micro average Fall-out achieved across all 10 signals

	w3	w4	w5	w6	w7	w8	w9	w10	average
baseline	64.71%	59.20%	58.97%	59.51%	55.26%	51.14%	55.18%	53.56%	57.19%
Sig prop	34.55%	15.71%	4.00%	1.43%	0.77%	0.00%	0.00%	0.00%	7.06%
Dist p2p	29.37%	25.54%	26.21%	23.58%	20.58%	19.85%	26.09%	15.67%	23.36%
Dist p2p*	23.18%	11.34%	8.82%	5.22%	2.31%	1.67%	10.00%	0.00%	7.82%
Dist p2v	20.31%	31.43%	36.46%	31.94%	27.95%	21.74%	32.00%	22.22%	28.01%
Dist p2v*	5.45%	11.34%	3.54%	2.14%	2.31%	0.83%	0.00%	0.00%	3.20%
Dist p2av	15.65%	21.88%	22.00%	21.44%	20.71%	23.41%	22.82%	20.78%	21.08%
Dist p2av*	6.04%	10.71%	2.67%	3.74%	0.77%	0.83%	0.00%	0.00%	3.09%

Table 5.17: Micro average Miss rate achieved across all 10 signals

	w3	w4	w5	w6	w7	w8	w9	w10	average
baseline	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Sig prop	20.00%	20.00%	40.00%	50.00%	60.00%	60.00%	100.00%	100.00%	56.25%
Dist p2p	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Dist p2p*	0.00%	0.00%	0.00%	0.00%	20.00%	20.00%	40.00%	60.00%	17.50%
Dist p2v	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	20.00%	2.50%
Dist p2v*	40.00%	20.00%	40.00%	50.00%	60.00%	60.00%	100.00%	100.00%	58.75%
Dist p2av	20.00%	0.00%	0.00%	0.00%	0.00%	0.00%	20.00%	20.00%	7.50%
Dist p2av*	20.00%	40.00%	20.00%	40.00%	60.00%	60.00%	100.00%	100.00%	55.00%

Table 5.18: Macro Accuracy, Fall-out, Miss rate, and Balanced accuracy for all algorithms

	Accuracy	Miss rate	Fall-out	bAcc
baseline	44.34%	0.00%	58.71%	70.65%
Sig prop	89.17%	48.21%	8.79%	71.50%
Dist p2p	77.22%	0.00%	24.02%	87.99%
Dist p2p*	91.20%	12.50%	8.59%	89.45%
Dist p2v	73.24%	1.79%	28.13%	85.04%
Dist p2v*	93.80%	51.79%	3.71%	72.25%
Dist p2av	79.72%	7.14%	21.00%	85.93%
Dist p2av*	94.17%	46.43%	3.61%	74.98%

5.3.1.2 Macro results

Table 5.18 reports the macro average accuracy, fall-out, miss rate, and balanced accuracy achieved for each algorithm ignoring all individual signals and window sizes, that is, by considering the 1080 data points at once.

The macro results from Table 5.18 indicate that the highest balanced accuracy is achieved by Dist p2p* with 89.45% while the lowest is the baseline with 70.65%. The lowest fall-out is achieved by the Dist p2av* approach with 3.61% whilst the highest is the baseline with 58.71%. Finally, the lowest miss rate are equally the baseline and Dist p2p 0.00% and the highest is Dist p2v* with 51.79%.

The macro results are strongly similar to the micro results with the exception that the highest balanced accuracy is the Dist p2p* as opposed to Dist p2p. In the following section we will discuss these results in more detail in order to select which approach is best suited for sentiment shift detection. Moreover, we will take a closer look at the effect of the window size on the performances.

Our experimental results show that both our algorithms Dist p2p* and Dist p2av outperform the baseline approach, and both yield high detection accuracy, and are therefore well suited for sentiment shift detection.

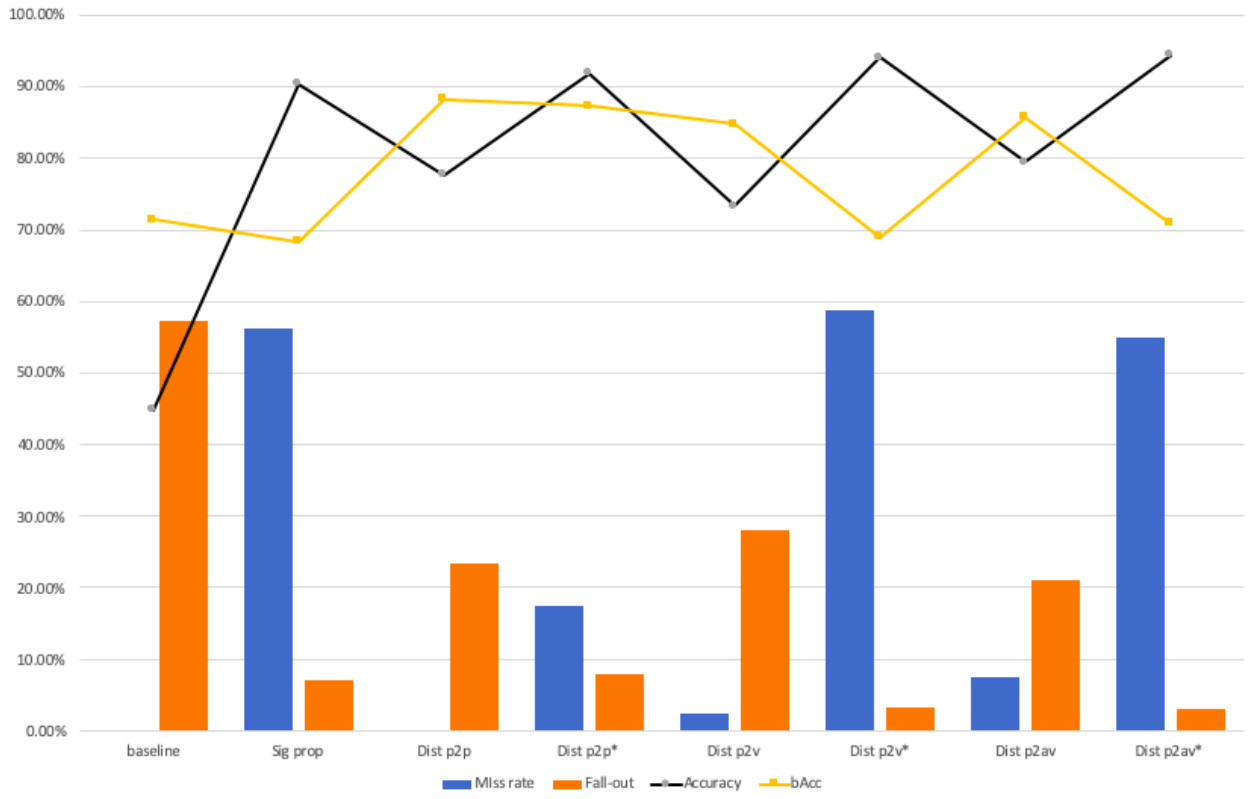

Figure 5.8: Micro averaged accuracy, fall-out, miss rate, and balanced accuracy for all signals across all window size

5.3.2 Discussions

Ideally, a good sentiment shift detection algorithm should achieve a very low fall-out and a very low miss rate while having a high enough balanced accuracy at the same time in order to satisfy both criteria explained in the previous section.

5.3.2.1 Algorithms Performances

Micro results

Figure 5.8 shows a bar plot of the fall-out and miss rate side by side as well as the accuracy and balanced accuracy performed by all algorithms on the micro level. We notice that the balanced accuracy by itself is insufficient to determine whether an algorithm is well-

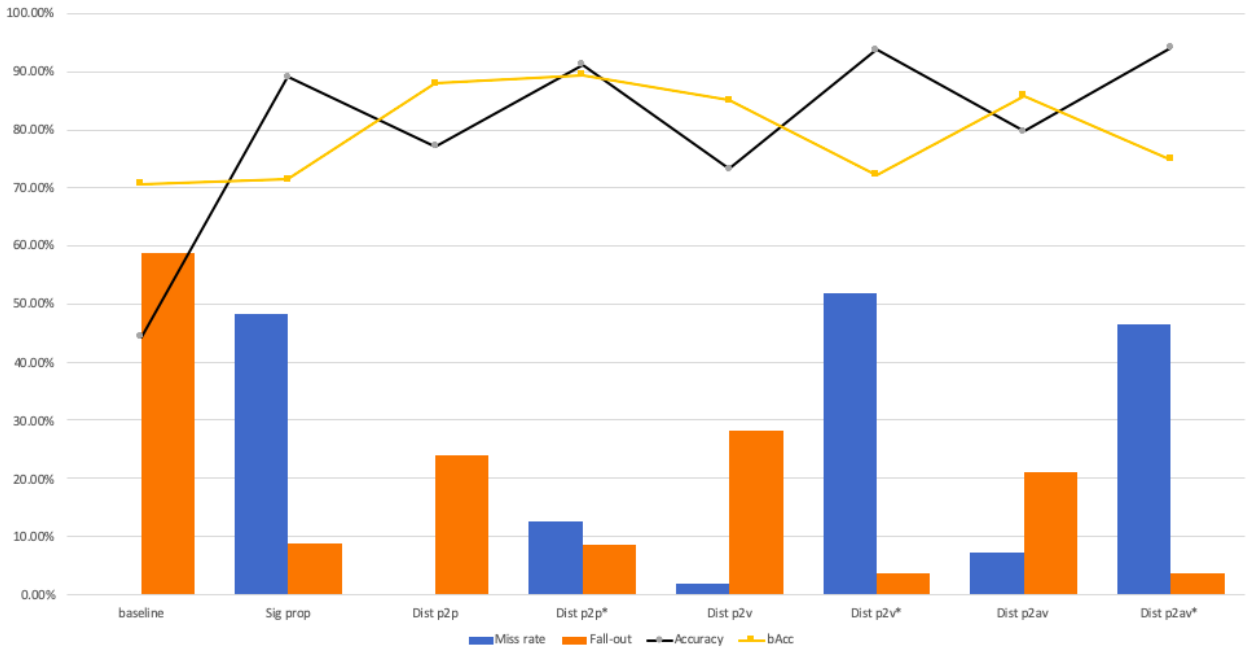

Figure 5.9: Macro averaged accuracy, fall-out, miss rate, and balanced accuracy for all signals across all window size

suited for sentiment shift detection. Similarly, the miss rate or fall out by themselves are not good indicators of the performances of an algorithm. Therefore, we need to consider all three evaluation metrics all together in order to evaluate the performances of each algorithm. We can see from Figure 5.8 that most of the time, a very low fall-out is coupled to a very high miss rate and vice-versa. Indeed, although Dist p2v and Dist p2p achieve a high balanced accuracy and very low miss rate, they both suffer from a high fall out, indicating that they perform well in our first requirement but very poorly in our second requirement. The baseline despite having a miss rate of 0.0% has an extremely high fall out, meaning that it tend to detect sentiment shift when they do not occur. Likewise, Sig prop, Dist p2v* and Dist p2av* have very low fall-out but very high miss rate, meaning that they perform well in our second requirement but very poorly in our first requirement.

On the other hand, both Dist p2p* and Dist p2av are much better suited for detecting

sentiment shift since they both achieve a low fall-out and a low miss rate while achieving a high balanced accuracy. They are therefore the two best candidates for our sentiment shift detection algorithm. We will analyze the macro result plot in order to confirm these findings.

Macro results

Figure 5.9 shows a bar plot of the fall-out and miss rate side by side as well as the accuracy and balanced accuracy performed by all algorithms on the macro level. Figure 5.9 confirms the findings discussed on the micro level, that is, the baseline, Sig prop, Dist p2p, Dist p2v, Dist2v*, and Dist p2av* are not suited for sentiment shift detection because they either have a high fall-out associated with a low miss rate or a high miss rate associated with a low fall-out which does not meet both of our requirements.

Now, comparing Dist p2p* versus Dist p2av, they show similar performances which both meet our requirements, namely, a low miss rate and a low fall-out. With a smaller fall-out, and slightly higher miss rate, we believe that Dist p2p* is better suited for automatically detecting a sentiment shift. However, if a particular application has a higher focus on not missing many sentiment shift, then the Dist p2av approach could be better suited.

5.3.2.2 Impact of the window size on the results

Figure 5.10 plots the balanced accuracy versus the size of the trend window. There is no clear pattern or correlation that is revealed on the graph, however, the algorithms seem to perform better with a window size slightly bigger than 3, i.e., 4 or 5, and the performances, overall, clearly decrease as the window gets bigger. That is, the lowest balanced accuracy are achieved with a window size of 10. In order to understand and explain this observation we can take a look at the fall-out and miss rate separately. Figure 5.11 thus is a 3D plot of the

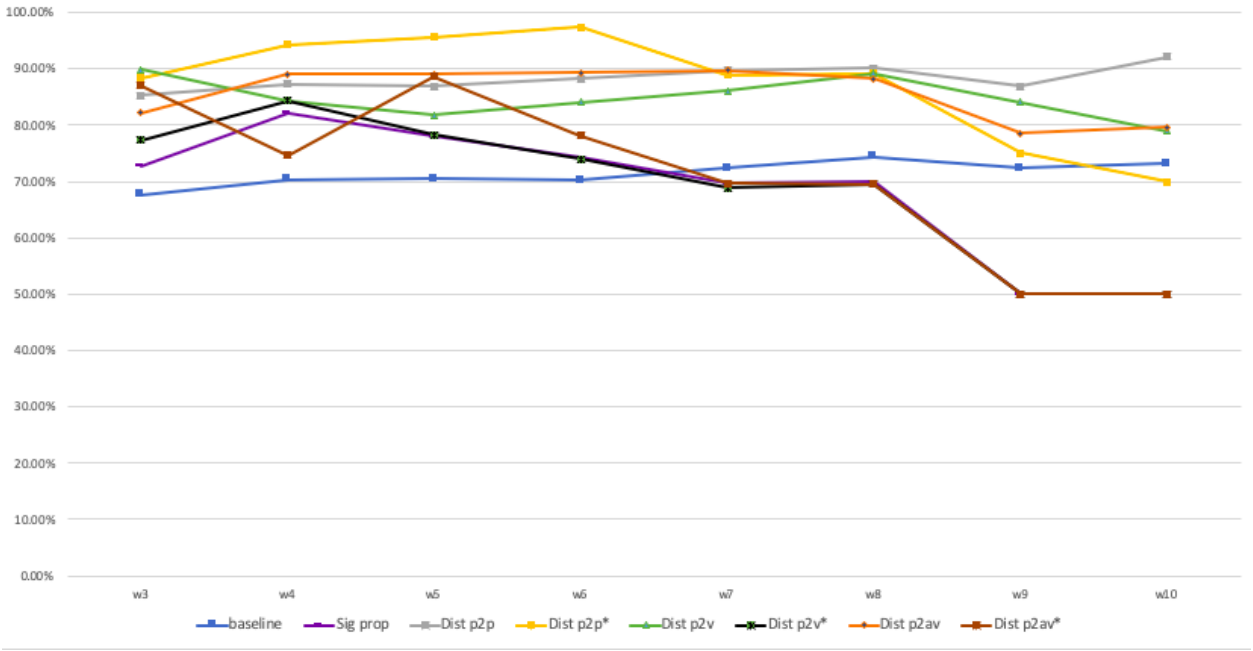

Figure 5.10: Accuracy vs window size for all algorithms

fall-out versus the window size, for all algorithms wile Figure 5.12 plots the miss rate. While Figure 5.11 shows that the fall-out tend to decrease as the window gets bigger , Figure 5.12 shows the opposite trend and shows that the miss rate greatly increase as the window gets bigger. These two observations would thus explain why the relation between the window size and performance was not observed with balanced accuracy. Indeed, as the window size gets bigger, the miss rate gets bigger but the fall-out gets smaller, the fall-out balances out the loss of performances reflected in the miss rate.

Table 5.19: Averaged bAcc, fall-out, and miss rate for all algorithm for each window size

	w3	w4	w5	w6	w7	w8	w9	w10
bAcc	81.30%	83.30%	83.58%	81.94%	79.33%	80.03%	68.37%	67.99%
Fall-out	24.91%	23.39%	20.33%	18.63%	16.33%	14.93%	18.26%	14.03%
Miss rate	12.50%	10.00%	12.50%	17.50%	25.00%	25.00%	45.00%	50.00%

We can conclude from these observations that performance is linked to the size of the window. Specifically, a narrow trend window induces more constraints and the algorithm becomes more specific and restricted regarding the trend it catches. Consequently a very

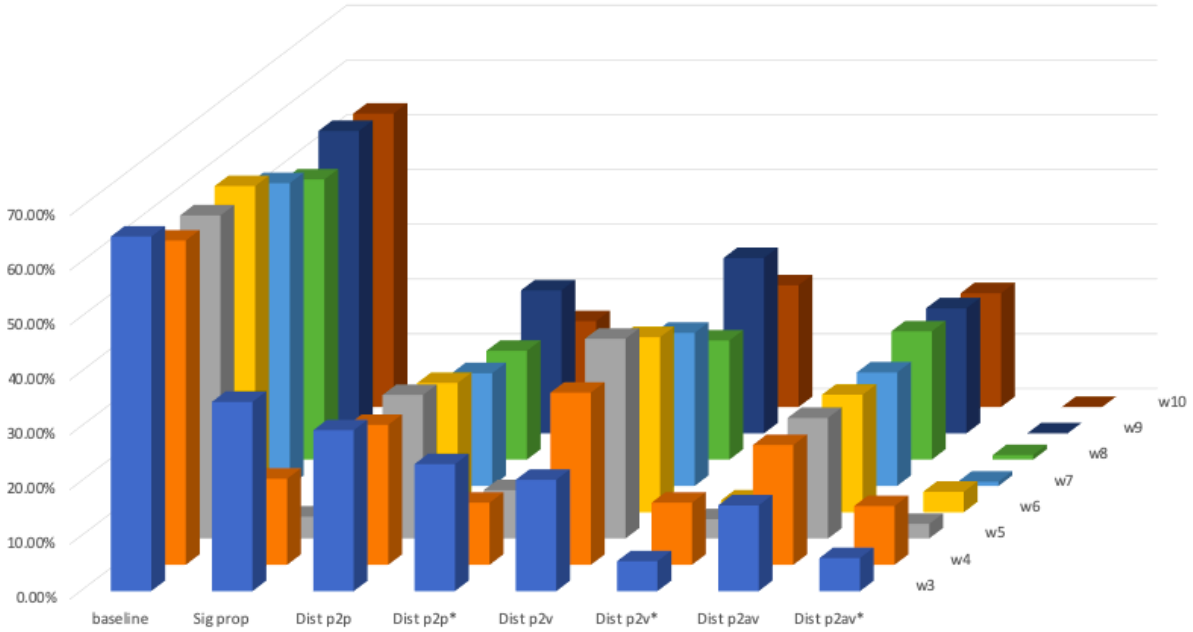

Figure 5.11: Fall-out vs window size for all algorithms

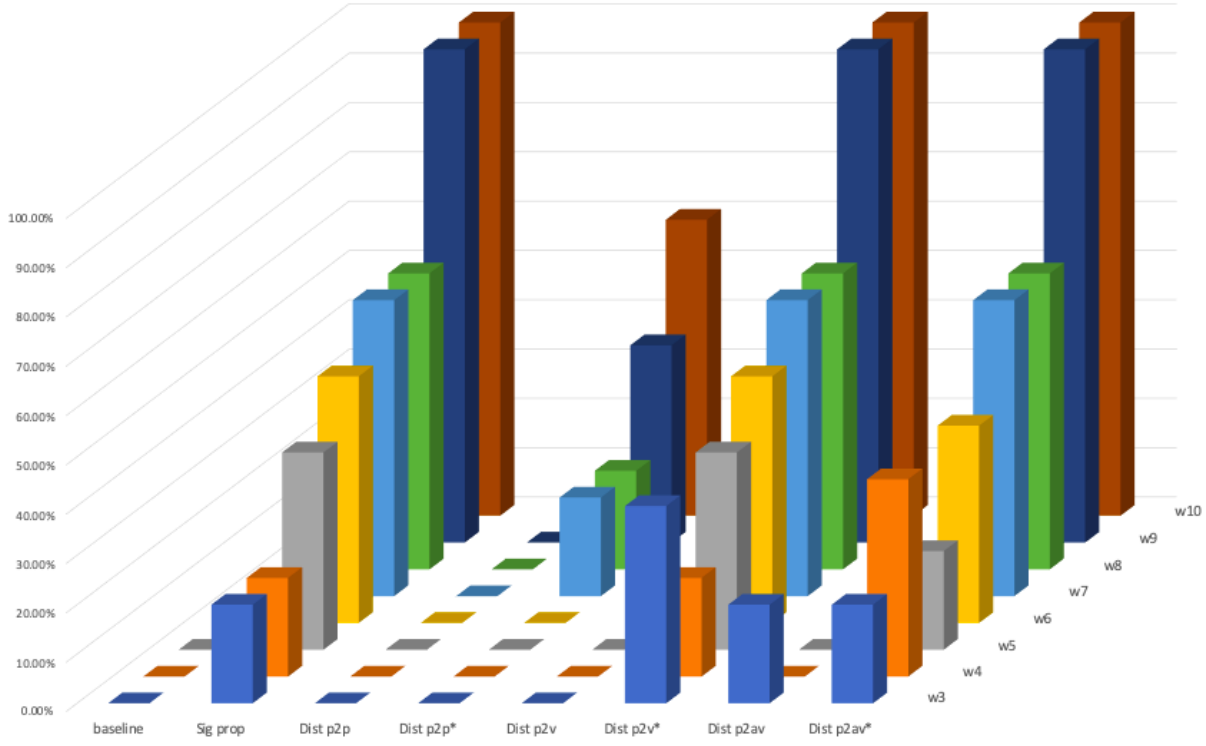

Figure 5.12: Miss rate vs window size for all algorithms

Table 5.20: bAcc, fall-out, and miss rate for the Dist p2p* algorithm for each window size

	w3	w4	w5	w6	w7	w8	w9	w10
bAcc	88.41%	94.33%	95.59%	97.39%	88.85%	89.17%	75.00%	70.00%
Fall-out	23.18%	11.34%	8.82%	5.22%	2.31%	1.67%	10.00%	0.00%
Miss rate	0.00%	0.00%	0.00%	0.00%	20.00%	20.00%	40.00%	60.00%

low miss rate but a high fall-out will be achieved, meaning that the algorithm does not do a great job of catching a trend, and will detect many sentiment shifts when they actually do not occur.

On the other hand, a wide trend window induces less constraints and the algorithm then becomes less strict and less restricted regarding the trend in catches. As a consequence, a high miss rate but a low fall-out will be achieved, meaning that the algorithm does not do a great job of catching the trend, and while it will not detect a shift when none has occurred, it will miss a high number of sentiment shift that actually have occurred.

Table 5.19 summarizes the average balanced accuracy, fall-out, and miss rate across all algorithms for each window size. It shows that on average, the highest balanced accuracy is achieved at a size of 5, the lowest fall-out is achieved with a window of size 10, and the lowest miss rate is achieved with a trend window of size 4. Since the approach Dist p2p* was chosen to be the best approach for sentiment shift detection, we take a look at its performances against the various window sizes separately and report the results in Table 5.20.

Table 5.20 indicates that the Dist p2p* approach achieves a miss rate of 0.00% for a window size ranging from 3 to 6. Amongst these window sizes, the lowest fall-out is achieved with a window of size 6 with a fall-out of 5.22%. Similarly, the highest balanced accuracy is achieved with a window size of 6 with 97.39%.

Therefore, for an accurate automatic sentiment shift detection system, we recommend using the Dist p2p* approach with a trend window of size 6.

6 Conclusion and Future Work

In this chapter, we summarize our research contributions to the field of sentiment analysis, quantification, and opinion shift detection. We also discuss possible future research directions suggested by our results.

6.1 Conclusion

Sentiment analysis can be applied to numerous domains provided that there is textual data to analyze. It is widely applied in the business intelligence area by tracking customers' opinions over time. In this document, we extended the aforementioned applications by applying sentiment quantification techniques combined with change-point detection models to automatically detect a shift of sentiment with respect to a topic or entity. To the best of our knowledge there are currently no research that focuses on automatically detecting a shift of sentiment. In this document we provide a solution to this problem by answering three questions that serve as guidelines to our work: (1) how to represent a Tweet in a way such that its representation reflects the sentiment it reflects; (2) how could we accurately quantify the proportion of positive Tweet and the proportion of negative Tweet from a collection; and (3) how to build a sentiment signal that captures the evolution of the sentiment over time, and accurately identify a sentiment shift. The answers to these questions are the core of our work. By answering these questions, we have made contributions in three areas, namely, sentiment analysis, sentiment quantification, and sentiment shift detection.

Focusing on sentiment analysis, we introduce a probabilistic approach to create a

paired sentiment lexicon that models the positivity and the negativity of words separately. We show that such a lexicon can be used to more accurately derive sentiment features for a Tweet. In addition, we show that employing these feature vectors with a multivariate Support Vector Machine (SVM) that optimizes statistical distances metrics can improve sentiment quantification accuracy. We further demonstrate that such a SVM machines achieve the best performance when they optimize the Hellinger Distance. Furthermore, we introduce a mean of representing sentiment over time through sentiment signals built from the aforementioned sentiment quantifier. Finally, we show how an automatic sentiment shift detection system can be implemented by analyzing the sentiment signals through geometric-based change-point detection algorithms. We find that measuring the change of sentiment through 2 dimensional Euclidean distance and capturing a shift using first and second order statistical moments was the most accurate of the methods we tested.

6.2 Future Work

We believe that there is potential for future contributions in all three areas mentioned in this work. Our techniques show that text mining techniques can automatically create sentiment lexicons for Twitter data that outperform generic sentiment lexicons. However, further improvements may be possible by incorporating hashtags, usernames, or the interrelationship of tweets within threads in the lexicon construction process. In addition, unlike online reviews, Tweets do not have explicit positive or negative feedback, therefore, efforts should also be spent in improving current Twitter datasets for sentiment analysis.

Sentiment quantification is a relatively new research area and many paths still need to be explored. Although the use of neural networks has already been investigated for sen-

timent quantification, they make little or no use of sentiment lexicons. Sentiment lexicons are valuable resources that provide additional insight on the sentiment carried by words. We believe that they could make a positive contribution and enhance current neural network approaches for this particular problem and also improve the explainability of neural network results.

Sentiment signal analysis is an emerging research area. Although we introduced one way of building them, other approaches should be investigated to build that take into consideration the quantity, i.e., the actual number of documents, as well as the proportion of documents with respect to each sentiment. Finally, opinion shift detection has the potential to enhance current business intelligence systems and research should focus on improving such detection algorithms. Change-point detection (CPD) is heavily dependent on the application it is used for, and CPD that works well in one particular area will most likely not perform well in another. Therefore we suggest researchers to investigate CPD applied specifically to sentiment signals which are a particular instance of online textual streams. Graph-based CPD methods as well as clustering CPD methods are promising in this application.

Bibliography

[1] W. Gao and F. Sebastiani, "Tweet sentiment: From classification to quantification," in *Advances in Social Networks Analysis and Mining (ASONAM), 2015 IEEE/ACM International Conference on*. IEEE, 2015, pp. 97–104.

[2] S.-M. Kim and E. Hovy, "Extracting opinions, opinion holders, and topics expressed in online news media text," in *Proceedings of the Workshop on Sentiment and Subjectivity in Text*. Association for Computational Linguistics, 2006, pp. 1–8.

[3] ——, "Determining the sentiment of opinions," in *Proceedings of the 20th international conference on Computational Linguistics*. Association for Computational Linguistics, 2004, p. 1367.

[4] B. Pang and L. Lee, "Opinion mining and sentiment analysis," *Foundations and trends in information retrieval*, vol. 2, no. 1-2, pp. 1–135, 2008.

[5] H. Wang, D. Can, A. Kazemzadeh, F. Bar, and S. Narayanan, "A system for real-time twitter sentiment analysis of 2012 us presidential election cycle," in *Proceedings of the ACL 2012 System Demonstrations*. Association for Computational Linguistics, 2012, pp. 115–120.

[6] M. Hu and B. Liu, "Mining and summarizing customer reviews," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2004, pp. 168–177.

[7] S.-M. Kim and E. Hovy, "Identifying and analyzing judgment opinions," in *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*. Association for Computational Linguistics, 2006, pp. 200–207.

[8] D. Gao, F. Wei, W. Li, X. Liu, and M. Zhou, "Cross-lingual sentiment lexicon learning with bilingual word graph label propagation," *Computational Linguistics*, 2015.

[9] T. Li, Y. Zhang, and V. Sindhwani, "A non-negative matrix tri-factorization approach to sentiment classification with lexical prior knowledge," in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*. Association for Computational Linguistics, 2009, pp. 244–252.

[10] N. A. Abdulla, N. A. Ahmed, M. A. Shehab, M. Al-Ayyoub, M. N. Al-Kabi, and S. Al-rifai, "Towards improving the lexicon-based approach for arabic sentiment analysis," *International Journal of Information Technology and Web Engineering (IJITWE)*, vol. 9, no. 3, pp. 55–71, 2014.

[11] V. Ng, S. Dasgupta, and S. Arifin, "Examining the role of linguistic knowledge sources in the automatic identification and classification of reviews," in *Proceedings of the COLING/ACL on Main conference poster sessions*. Association for Computational Linguistics, 2006, pp. 611–618.

[12] B. Pang and L. Lee, "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts," in *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2004, p. 271.

[13] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up?: sentiment classification using machine learning techniques," in *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. Association for Computational Linguistics, 2002, pp. 79–86.

[14] S. Zhou, Q. Chen, and X. Wang, "Active deep networks for semi-supervised sentiment classification," in *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*. Association for Computational Linguistics, 2010, pp. 1515–1523.

[15] W. Wei and J. A. Gulla, "Sentiment learning on product reviews via sentiment ontology tree," in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2010, pp. 404–413.

[16] X. Ding, B. Liu, and P. S. Yu, "A holistic lexicon-based approach to opinion mining," in *Proceedings of the 2008 international conference on web search and data mining*. ACM, 2008, pp. 231–240.

[17] M. Hu and B. Liu, "Mining opinion features in customer reviews," in *AAAI*, vol. 4, no. 4, 2004, pp. 755–760.

[18] A. Z. Khan, M. Atique, and V. Thakare, "Combining lexicon-based and learning-based methods for twitter sentiment analysis," *International Journal of Electronics, Communication and Soft Computing Science & Engineering (IJECSCSE)*, p. 89, 2015.

[19] 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, 2011.

[20] P. D. Turney, "Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews," in *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2002, pp. 417–424.

[21] B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," in *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2005, pp. 115–124.

[22] A. B. Goldberg and X. Zhu, "Seeing stars when there aren't many stars: graph-based semi-supervised learning for sentiment categorization," in *Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing*. Association for Computational Linguistics, 2006, pp. 45–52.

[23] A. Pak and P. Paroubek, "Twitter as a corpus for sentiment analysis and opinion mining." in *LREc*, vol. 10, no. 2010, 2010.

[24] D. Davidov, O. Tsur, and A. Rappoport, "Enhanced sentiment learning using twitter hashtags and smileys," in *Proceedings of the 23rd international conference on computational linguistics: posters*. Association for Computational Linguistics, 2010, pp. 241–249.

[25] L. Barbosa and J. Feng, "Robust sentiment detection on twitter from biased and noisy data," in *Proceedings of the 23rd international conference on computational linguistics: posters*. Association for Computational Linguistics, 2010, pp. 36–44.

[26] E. Kouloumpis, T. Wilson, and J. D. Moore, "Twitter sentiment analysis: The good the bad and the omg!" *Icwsm*, vol. 11, no. 538-541, p. 164, 2011.

[27] J. Zhao, L. Dong, J. Wu, and K. Xu, "Moodlens: an emoticon-based sentiment analysis system for chinese tweets," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 1528–1531.

[28] A. Go, L. Huang, and R. Bhayani, "Twitter sentiment analysis," *Entropy*, vol. 17, p. 252, 2009.

[29] S. M. Mohammad, S. Kiritchenko, and X. Zhu, "Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets," *arXiv preprint arXiv:1308.6242*, 2013.

[30] S. Kiritchenko, X. Zhu, and S. M. Mohammad, "Sentiment analysis of short informal texts," *Journal of Artificial Intelligence Research*, vol. 50, pp. 723–762, 2014.

[31] D. Tang, F. Wei, B. Qin, T. Liu, and M. Zhou, "Coooolll: A deep learning system for twitter sentiment classification," in *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, 2014, pp. 208–212.

[32] K. Labille, S. Alfarhood, and S. Gauch, "Estimating sentiment via probability and information theory," in *Proceedings of the 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management - Volume 1: KDIR,*, 2016, pp. 121–129.

[33] D. J. Hopkins and G. King, "A method of automated nonparametric content analysis for social science," *American Journal of Political Science*, vol. 54, no. 1, pp. 229–247, 2010.

[34] V. GonzáLez-Castro, R. Alaiz-RodríGuez, and E. Alegre, "Class distribution estimation based on the hellinger distance," *Information Sciences*, vol. 218, pp. 146–164, 2013.

[35] J. Barranquero, J. Díez, and J. J. del Coz, "Quantification-oriented learning based on reliable classifiers," *Pattern Recognition*, vol. 48, no. 2, pp. 591–604, 2015.

[36] A. Esuli and F. Sebastiani, "Optimizing text quantifiers for multivariate loss functions," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 9, no. 4, p. 27, 2015.

[37] D. Vilares, Y. Doval, M. A. Alonso, and C. Gómez-Rodríguez, "Lys at semeval-2016 task 4: Exploiting neural activation values for twitter sentiment classification and quantification," in *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, 2016, pp. 79–84.

[38] D. Stojanovski, G. Strezoski, G. Madjarov, and I. Dimitrovski, "Finki at semeval-2016 task 4: Deep learning architecture for twitter sentiment analysis," in *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, 2016, pp. 149–154.

[39] M. Cliche, "Bb_twtr at semeval-2017 task 4: twitter sentiment analysis with cnns and lstms," *arXiv preprint arXiv:1704.06125*, 2017.

[40] Y. Yang, T. Pierce, and J. Carbonell, "A study of retrospective and on-line event detection," in *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 1998, pp. 28–36.

[41] J. Kleinberg, "Bursty and hierarchical structure in streams," *Data Mining and Knowledge Discovery*, vol. 7, no. 4, pp. 373–397, 2003.

[42] G. P. C. Fung, J. X. Yu, P. S. Yu, and H. Lu, "Parameter free bursty events detection in text streams," in *Proceedings of the 31st international conference on Very large data bases*. VLDB Endowment, 2005, pp. 181–192.

[43] Q. He, K. Chang, and E.-P. Lim, "Analyzing feature trajectories for event detection," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2007, pp. 207–214.

[44] L. Chen and A. Roy, "Event detection from flickr data through wavelet-based spatial analysis," in *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM, 2009, pp. 523–532.

[45] T. Sakaki, M. Okazaki, and Y. Matsuo, "Earthquake shakes twitter users: real-time event detection by social sensors," in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 851–860.

[46] S. Petrović, M. Osborne, and V. Lavrenko, "Streaming first story detection with application to twitter," in *Human language technologies: The 2010 annual conference of the north american chapter of the association for computational linguistics*. Association for Computational Linguistics, 2010, pp. 181–189.

[47] O. Ozdikis, P. Senkul, and H. Oguztuzun, "Semantic expansion of tweet contents for enhanced event detection in twitter," in *Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012)*. IEEE Computer Society, 2012, pp. 20–24.

[48] J. Weng and B.-S. Lee, "Event detection in twitter." *ICWSM*, vol. 11, pp. 401–408, 2011.

[49] M. Cordeiro, "Twitter event detection: combining wavelet analysis and topic inference summarization," in *Doctoral symposium on informatics engineering*, 2012, pp. 11–16.

[50] C. Zhang, L. Liu, D. Lei, Q. Yuan, H. Zhuang, T. Hanratty, and J. Han, "Triovecevent: Embedding-based online local event detection in geo-tagged tweet streams," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 595–604.

[51] J. Wiebe, T. Wilson, and C. Cardie, "Annotating expressions of opinions and emotions in language," *Language resources and evaluation*, vol. 39, no. 2-3, pp. 165–210, 2005.

[52] S. Baccianella, A. Esuli, and F. Sebastiani, "Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining." in *LREC*, vol. 10, 2010, pp. 2200–2204.

[53] A. Esuli and F. Sebastiani, "Sentiwordnet: A publicly available lexical resource for opinion mining," in *Proceedings of LREC*, vol. 6. Citeseer, 2006, pp. 417–422.

[54] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *European conference on machine learning*. Springer, 1998, pp. 137–142.

[55] ——, "A support vector method for multivariate performance measures," in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, pp. 377–384.

[56] D. Johnson and S. Sinanovic, "Symmetrizing the kullback-leibler distance," *IEEE Transactions on Information Theory*, 2001.

[57] A. Ullah, "Entropy, divergence and distance measures with econometric applications," *Journal of Statistical Planning and Inference*, vol. 49, no. 1, pp. 137–162, 1996.

[58] T. M. Osán, D. G. Bussandri, and P. W. Lamberti, "Monoparametric family of metrics derived from classical jensen–shannon divergence," *Physica A: Statistical Mechanics and its Applications*, vol. 495, pp. 336–344, 2018.

[59] S. Aminikhanghahi and D. J. Cook, "A survey of methods for time series change point detection," *Knowledge and information systems*, vol. 51, no. 2, pp. 339–367, 2017.

[60] D.-H. Tran, "Automated change detection and reactive clustering in multivariate streaming data," in *2019 IEEE-RIVF International Conference on Computing and Communication Technologies (RIVF)*. IEEE, 2019, pp. 1–6.

[61] H. R. Mohseni, A. Maghsoudi, and M. B. Shamsollahi, "Seizure detection in eeg signals: A comparison of different approaches," in *Engineering in Medicine and Biology Society, 2006. EMBS'06. 28th Annual International Conference of the IEEE*. IEEE, 2006, pp. 6724–6727.

[62] P. E. McSharry, L. A. Smith, and L. Tarassenko, "Prediction of epileptic seizures: are nonlinear methods relevant?" *Nature medicine*, vol. 9, no. 3, p. 241, 2003.

[63] S. S. Alam and M. I. H. Bhuiyan, "Detection of seizure and epilepsy using higher order statistics in the emd domain," *IEEE journal of biomedical and health informatics*, vol. 17, no. 2, pp. 312–318, 2013.

[64] M. Thelwall, K. Buckley, and G. Paltoglou, "Sentiment strength detection for the social web," *Journal of the Association for Information Science and Technology*, vol. 63, no. 1, pp. 163–173, 2012.

[65] H. Saif, M. Fernandez, Y. He, and H. Alani, "Evaluation datasets for twitter sentiment analysis: a survey and a new dataset, the sts-gold," 2013.

[66] D. A. Cieslak, T. R. Hoens, N. V. Chawla, and W. P. Kegelmeyer, "Hellinger distance decision trees are robust and skew-insensitive," *Data Mining and Knowledge Discovery*, vol. 24, no. 1, pp. 136–158, 2012.

A Appendix

A.1 Raw sentiment signals description

Table A.1: Detailed description of the sentiment signals

	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
point 1	0.22	0.22	-0.22	-0.22	0.91	-0.93	0.76	-0.74	-0.18	0.07
point 2	0.22	0.22	-0.22	-0.22	0.86	-0.86	0.71	-0.7	0.18	0.1
point 3	0.2	0.2	-0.2	-0.2	0.74	-0.74	0.69	-0.64	-0.05	0.08
point 4	0.23	0.23	-0.23	-0.23	0.66	-0.65	0.7	-0.72	0.2	0.15
point 5	0.25	0.25	-0.25	-0.25	0.57	-0.55	0.68	-0.67	-0.03	0.14
point 6	0.15	0.35	-0.35	-0.15	0.44	-0.45	0.68	-0.68	0.14	0.12
point 7	0.12	0.37	-0.33	-0.17	0.36	-0.33	0.65	-0.65	-0.24	0.66
point 8	0.13	0.33	-0.34	-0.13	0.22	-0.24	0.55	-0.55	0	0.65
point 9	0.12	0.32	-0.32	-0.16	0.14	-0.12	0.24	-0.37	0.08	0.37
point 10	0.21	0.21	-0.21	-0.23	0.03	-0.03	0.16	-0.16	0.15	0.34
point 11	0.22	0.22	-0.22	-0.22	-0.08	0.09	-0.1	0.1	-0.15	0.15
point 12	0.25	0.25	-0.25	-0.25	-0.16	0.18	-0.26	0.26	-0.1	0.12
point 13	0.24	0.24	-0.24	-0.24	-0.25	0.27	-0.55	0.55	-0.28	0.09
point 14	0.24	0.24	-0.24	-0.24	-0.37	0.37	-0.67	0.66	0.06	0.12
point 15	0.26	0.26	-0.26	-0.26	-0.45	0.42	-0.65	0.71	-0.11	-0.27
point 16	0.2	0.2	-0.2	-0.2	-0.57	0.56	-0.68	0.68	0 0.17	
point 17	0.21	0.21	-0.21	-0.21	-0.66	0.62	-0.66	0.65	-0.14	0.15
point 18	0.08	0.48	-0.48	-0.08	-0.72	0.75	-0.72	0.7	0.1	0.16
point 19	0.21	0.21	-0.21	-0.21	-0.85	0.85	-0.65	0.67	-0.02	0.1
point 20	0.29	0.29	-0.29	-0.29	-0.9	0.91	-0.7	0.71	0.02	0.1

A.2 Raw sentiment shift detection accuracy

Table A.2: Raw accuracy results for each window size

	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
W3										
baseline	29.41%	64.71%	64.71%	35.29%	0.00%	0.00%	17.65%	35.29%	73.33%	64.71%
Sig prop	76.47%	88.24%	82.35%	76.47%	23.53%	5.88%	58.82%	64.71%	88.24%	88.24%
dist p2p	76.47%	70.59%	64.71%	76.47%	76.47%	82.35%	58.82%	58.82%	76.47%	82.35%
Dist p2p*	76.47%	76.47%	76.47%	76.47%	82.35%	82.35%	76.47%	70.59%	76.47%	88.24%
Dist p2v	76.47%	70.59%	70.59%	82.35%	76.47%	82.35%	88.24%	88.24%	88.24%	88.24%
Dist p2v*	88.24%	94.12%	94.12%	94.12%	94.12%	82.35%	100.00%	88.24%	94.12%	94.12%
Dist p2av	76.47%	70.59%	70.59%	82.35%	76.47%	82.35%	88.24%	88.24%	88.24%	88.24%
Dist p2av*	88.24%	94.12%	94.12%	94.12%	94.12%	82.35%	100.00%	88.24%	94.12%	94.12%
W4										
baseline	43.75%	68.75%	62.50%	37.50%	0.00%	0.00%	25.00%	37.50%	81.25%	81.25%
Sig prop	81.25%	87.50%	87.50%	81.25%	75.00%	75.00%	68.75%	87.50%	93.75%	100.00%
dist p2p	81.25%	81.25%	81.25%	81.25%	68.75%	68.75%	62.50%	62.50%	75.00%	93.75%
Dist p2p*	87.50%	87.50%	87.50%	87.50%	100.00%	93.75%	75.00%	87.50%	87.50%	100.00%
Dist p2v	81.25%	75.00%	68.75%	81.25%	50.00%	68.75%	50.00%	62.50%	68.75%	93.75%
Dist p2v*	81.25%	87.50%	81.25%	87.50%	93.75%	93.75%	75.00%	93.75%	87.50%	100.00%
Dist p2av	81.25%	68.75%	68.75%	81.25%	43.75%	62.50%	50.00%	68.75%	56.25%	81.25%
Dist p2av*	75.00%	87.50%	81.25%	81.25%	81.25%	81.25%	68.75%	87.50%	81.25%	100.00%

Table A.3: Raw accuracy results for each window size (continued)

	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
					W5					
baseline	26.67%	73.33%	66.67%	46.67%	0.00%	0.00%	20.00%	33.33%	80.00%	93.33%
Sig prop	93.33%	100.00%	100.00%	86.67%	93.33%	100.00%	80.00%	93.33%	93.33%	93.33%
dist p2p	73.33%	80.00%	80.00%	73.33%	66.67%	73.33%	60.00%	66.67%	86.67%	93.33%
Dist p2p*	86.67%	93.33%	93.33%	73.33%	100.00%	100.00%	86.67%	93.33%	93.33%	100.00%
Dist p2v	66.67%	60.00%	66.67%	60.00%	46.67%	66.67%	66.67%	73.33%	66.67%	86.67%
Dist p2v*	80.00%	93.33%	93.33%	93.33%	93.33%	100.00%	86.67%	100.00%	100.00%	100.00%
Dist p2av	66.67%	66.67%	66.67%	60.00%	40.00%	40.00%	60.00%	60.00%	66.67%	86.67%
Dist p2av*	86.67%	100.00%	100.00%	93.33%	93.33%	93.33%	86.67%	86.67%	100.00%	100.00%
					W6					
baseline	50.00%	57.14%	50.00%	42.86%	0.00%	0.00%	21.43%	28.57%	78.57%	92.86%
Sig prop	92.86%	100.00%	100.00%	92.86%	100.00%	100.00%	85.71%	100.00%	100.00%	92.86%
dist p2p	78.57%	78.57%	78.57%	85.71%	71.43%	64.29%	71.43%	64.29%	85.71%	92.86%
Dist p2p*	92.86%	92.86%	92.86%	92.86%	100.00%	100.00%	85.71%	92.86%	100.00%	100.00%
Dist p2v	78.57%	57.14%	64.29%	71.43%	57.14%	64.29%	64.29%	78.57%	71.43%	85.71%
Dist p2v*	92.86%	100.00%	100.00%	92.86%	92.86%	100.00%	85.71%	100.00%	100.00%	92.86%
Dist p2av	71.43%	50.00%	57.14%	71.43%	42.86%	35.71%	64.29%	78.57%	71.43%	85.71%
Dist p2av*	85.71%	92.86%	92.86%	92.86%	100.00%	92.86%	92.86%	100.00%	100.00%	92.86%
					W7					
baseline	46.15%	69.23%	69.23%	53.85%	0.00%	0.00%	23.08%	23.08%	84.62%	92.31%
Sig prop	92.31%	100.00%	100.00%	92.31%	100.00%	100.00%	92.31%	100.00%	100.00%	92.31%
dist p2p	84.62%	84.62%	84.62%	84.62%	76.92%	69.23%	69.23%	61.54%	92.31%	92.31%
Dist p2p*	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	84.62%	92.31%	100.00%	92.31%
Dist p2v	69.23%	69.23%	76.92%	76.92%	61.54%	61.54%	76.92%	76.92%	76.92%	84.62%
Dist p2v*	92.31%	100.00%	100.00%	92.31%	92.31%	100.00%	84.62%	100.00%	100.00%	92.31%
Dist p2av	69.23%	61.54%	61.54%	76.92%	38.46%	38.46%	69.23%	92.31%	69.23%	84.62%
Dist p2av*	92.31%	100.00%	100.00%	92.31%	100.00%	92.31%	100.00%	100.00%	100.00%	92.31%

Table A.4: Raw accuracy results for each window size (continued)

	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
					W8					
baseline	50.00%	83.33%	83.33%	58.33%	0.00%	0.00%	16.67%	25.00%	83.33%	100.00%
Sig prop	91.67%	100.00%	100.00%	91.67%	100.00%	100.00%	100.00%	100.00%	100.00%	91.67%
dist p2p	83.33%	83.33%	83.33%	83.33%	83.33%	66.67%	75.00%	66.67%	91.67%	91.67%
Dist p2p*	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	91.67%	91.67%	100.00%	91.67%
Dist p2v	75.00%	75.00%	75.00%	83.33%	75.00%	75.00%	83.33%	75.00%	83.33%	91.67%
Dist p2v*	91.67%	100.00%	100.00%	91.67%	100.00%	100.00%	91.67%	100.00%	100.00%	91.67%
Dist p2av	75.00%	66.67%	58.33%	75.00%	50.00%	41.67%	75.00%	91.67%	66.67%	83.33%
Dist p2av*	91.67%	100.00%	100.00%	91.67%	100.00%	100.00%	100.00%	100.00%	100.00%	91.67%
					W9					
baseline	54.55%	63.64%	63.64%	63.64%	0.00%	0.00%	18.18%	18.18%	81.82%	100.00%
Sig prop	90.91%	90.91%	90.91%	90.91%	100.00%	100.00%	100.00%	100.00%	100.00%	90.91%
dist p2p	9.09%	81.82%	81.82%	81.82%	90.91%	72.73%	81.82%	72.73%	90.91%	90.91%
Dist p2p*	9.09%	100.00%	100.00%	90.91%	100.00%	100.00%	100.00%	100.00%	100.00%	90.91%
Dist p2v	9.09%	63.64%	63.64%	72.73%	81.82%	72.73%	81.82%	72.73%	90.91%	90.91%
Dist p2v*	90.91%	90.91%	90.91%	90.91%	100.00%	100.00%	100.00%	100.00%	100.00%	90.91%
Dist p2av	9.09%	54.55%	54.55%	72.73%	54.55%	45.45%	81.82%	90.91%	72.73%	81.82%
Dist p2av*	90.91%	90.91%	90.91%	90.91%	100.00%	100.00%	100.00%	100.00%	100.00%	90.91%
					W10					
baseline	50.00%	80.00%	80.00%	50.00%	0.00%	0.00%	20.00%	20.00%	80.00%	100.00%
Sig prop	90.00%	90.00%	90.00%	90.00%	100.00%	100.00%	100.00%	100.00%	100.00%	90.00%
dist p2p	90.00%	90.00%	90.00%	80.00%	90.00%	70.00%	80.00%	80.00%	90.00%	90.00%
Dist p2p*	90.00%	100.00%	100.00%	90.00%	100.00%	100.00%	100.00%	100.00%	100.00%	90.00%
Dist p2v	80.00%	70.00%	70.00%	80.00%	80.00%	60.00%	90.00%	80.00%	90.00%	80.00%
Dist p2v*	90.00%	90.00%	90.00%	90.00%	100.00%	100.00%	100.00%	100.00%	100.00%	90.00%
Dist p2av	60.00%	50.00%	60.00%	60.00%	50.00%	50.00%	90.00%	90.00%	60.00%	80.00%
Dist p2av*	90.00%	90.00%	90.00%	90.00%	100.00%	100.00%	100.00%	100.00%	100.00%	90.00%

A.3 Raw sentiment detection fall-out

Table A.5: Raw fall-out results for each window size

	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
					W3					
baseline	80.00%	40.00%	40.00%	73.33%	100.00%	100.00%	82.35%	64.71%	26.67%	40.00%
Sig prop	20.00%	13.33%	20.00%	20.00%	76.47%	94.12%	41.18%	35.29%	11.76%	13.33%
dist p2p	26.67%	33.33%	40.00%	26.67%	23.53%	17.65%	41.18%	41.18%	23.53%	20.00%
Dist p2p*	26.67%	26.67%	26.67%	26.67%	17.65%	17.65%	23.53%	29.41%	23.53%	13.33%
Dist p2v	26.67%	33.33%	33.33%	20.00%	23.53%	17.65%	11.76%	11.76%	11.76%	13.33%
Dist p2v*	6.67%	0.00%	0.00%	0.00%	5.88%	17.65%	0.00%	11.76%	5.88%	6.67%
Dist p2av	26.67%	33.33%	33.33%	20.00%	23.53%	17.65%	11.76%	11.76%	11.76%	13.33%
Dist p2av*	6.67%	0.00%	0.00%	0.00%	5.88%	17.65%	0.00%	11.76%	5.88%	6.67%
					W4					
baseline	64.29%	35.71%	42.86%	71.43%	100.00%	100.00%	75.00%	62.50%	18.75%	21.43%
Sig prop	14.29%	14.29%	14.29%	14.29%	25.00%	25.00%	31.25%	12.50%	6.25%	0.00%
dist p2p	21.43%	21.43%	21.43%	21.43%	31.25%	31.25%	37.50%	37.50%	25.00%	7.14%
Dist p2p*	14.29%	14.29%	14.29%	14.29%	0.00%	6.25%	25.00%	12.50%	12.50%	0.00%
Dist p2v	21.43%	28.57%	35.71%	21.43%	50.00%	31.25%	50.00%	37.50%	31.25%	7.14%
Dist p2v*	21.43%	7.14%	14.29%	14.29%	6.25%	6.25%	25.00%	6.25%	12.50%	0.00%
Dist p2av	21.43%	35.71%	35.71%	21.43%	56.25%	37.50%	50.00%	31.25%	43.75%	21.43%
Dist p2av*	21.43%	7.14%	14.29%	14.29%	18.75%	18.75%	31.25%	12.50%	18.75%	0.00%

Table A.6: Raw fall-out results for each window size (continued)

	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
W5										
baseline	84.62%	30.77%	38.46%	61.54%	100.00%	100.00%	80.00%	66.67%	20.00%	7.69%
Sig prop	0.00%	0.00%	0.00%	0.00%	6.67%	0.00%	20.00%	6.67%	6.67%	0.00%
dist p2p	30.77%	23.08%	23.08%	30.77%	33.33%	26.67%	40.00%	33.33%	13.33%	7.69%
Dist p2p*	15.38%	7.69%	7.69%	30.77%	0.00%	0.00%	13.33%	6.67%	6.67%	0.00%
Dist p2v	38.46%	46.15%	38.46%	46.15%	53.33%	33.33%	33.33%	26.67%	33.33%	15.38%
Dist p2v*	15.38%	0.00%	0.00%	0.00%	6.67%	0.00%	13.33%	0.00%	0.00%	0.00%
Dist p2av	38.46%	38.46%	38.46%	46.15%	60.00%	60.00%	40.00%	40.00%	33.33%	15.38%
Dist p2av*	7.69%	0.00%	0.00%	0.00%	6.67%	6.67%	13.33%	13.33%	0.00%	0.00%
W6										
baseline	53.85%	46.15%	53.85%	61.54%	100.00%	100.00%	78.57%	71.43%	21.43%	8.33%
Sig prop	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	14.29%	0.00%	0.00%	0.00%
dist p2p	23.08%	23.08%	23.08%	15.38%	28.57%	35.71%	28.57%	35.71%	14.29%	8.33%
Dist p2p*	7.69%	7.69%	7.69%	7.69%	0.00%	0.00%	14.29%	7.14%	0.00%	0.00%
Dist p2v	23.08%	46.15%	38.46%	30.77%	42.86%	35.71%	35.71%	21.43%	28.57%	16.67%
Dist p2v*	0.00%	0.00%	0.00%	0.00%	7.14%	0.00%	14.29%	0.00%	0.00%	0.00%
Dist p2av	30.77%	53.85%	46.15%	30.77%	57.14%	64.29%	35.71%	21.43%	28.57%	16.67%
Dist p2av*	7.69%	7.69%	7.69%	0.00%	0.00%	7.14%	7.14%	0.00%	0.00%	8.33%
W7										
baseline	58.33%	33.33%	33.33%	50.00%	100.00%	100.00%	76.92%	76.92%	15.38%	8.33%
Sig prop	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	7.69%	0.00%	0.00%	0.00%
dist p2p	16.67%	16.67%	16.67%	16.67%	23.08%	30.77%	30.77%	38.46%	7.69%	8.33%
Dist p2p*	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	15.38%	7.69%	0.00%	0.00%
Dist p2v	33.33%	33.33%	25.00%	25.00%	38.46%	38.46%	23.08%	23.08%	23.08%	16.67%
Dist p2v*	0.00%	0.00%	0.00%	0.00%	7.69%	0.00%	15.38%	0.00%	0.00%	0.00%
Dist p2av	33.33%	41.67%	41.67%	25.00%	61.54%	61.54%	30.77%	7.69%	30.77%	16.67%
Dist p2av*	0.00%	0.00%	0.00%	0.00%	0.00%	7.69%	0.00%	0.00%	0.00%	0.00%

Table A.7: Raw fall-out results for each window size (continued)

	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
					W8					
baseline	54.55%	18.18%	18.18%	45.45%	100.00%	100.00%	83.33%	75.00%	16.67%	0.00%
Sig prop	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
dist p2p	18.18%	18.18%	18.18%	18.18%	16.67%	33.33%	25.00%	33.33%	8.33%	9.09%
Dist p2p*	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	8.33%	8.33%	0.00%	0.00%
Dist p2v	27.27%	27.27%	27.27%	18.18%	25.00%	25.00%	16.67%	25.00%	16.67%	9.09%
Dist p2v*	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	8.33%	0.00%	0.00%	0.00%
Dist p2av	27.27%	36.36%	45.45%	27.27%	50.00%	58.33%	25.00%	8.33%	33.33%	18.18%
Dist p2av*	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
					W9					
baseline	50.00%	40.00%	40.00%	40.00%	100.00%	100.00%	81.82%	81.82%	18.18%	0.00%
Sig prop	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
dist p2p	100.00%	20.00%	20.00%	20.00%	9.09%	27.27%	18.18%	27.27%	9.09%	10.00%
Dist p2p*	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Dist p2v	100.00%	40.00%	40.00%	30.00%	18.18%	27.27%	18.18%	27.27%	9.09%	10.00%
Dist p2v*	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Dist p2av	100.00%	50.00%	50.00%	30.00%	45.45%	54.55%	18.18%	9.09%	27.27%	20.00%
Dist p2av*	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
					W10					
baseline	55.56%	22.22%	22.22%	55.56%	100.00%	100.00%	80.00%	80.00%	20.00%	0.00%
Sig prop	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
dist p2p	11.11%	11.11%	11.11%	22.22%	10.00%	30.00%	20.00%	20.00%	10.00%	11.11%
Dist p2p*	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Dist p2v	22.22%	33.33%	33.33%	22.22%	20.00%	40.00%	10.00%	20.00%	10.00%	11.11%
Dist p2v*	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Dist p2av	44.44%	55.56%	44.44%	44.44%	50.00%	50.00%	10.00%	10.00%	40.00%	22.22%
Dist p2av*	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%

A.4 Raw sentiment detection miss rate

Table A.8: Raw miss rate results for each window size

	s1	s2	s3	s4	s10
W3					
baseline	0.00%	0.00%	0.00%	0.00%	0.00%
Sig prop	50.00%	0.00%	0.00%	50.00%	0.00%
dist p2p	0.00%	0.00%	0.00%	0.00%	0.00%
Dist p2p*	0.00%	0.00%	0.00%	0.00%	0.00%
Dist p2v	0.00%	0.00%	0.00%	0.00%	0.00%
Dist p2v*	50.00%	50.00%	50.00%	50.00%	0.00%
Dist p2av	0.00%	0.00%	0.00%	0.00%	0.00%
Dist p2av*	50.00%	50.00%	50.00%	50.00%	0.00%
W4					
baseline	0.00%	0.00%	0.00%	0.00%	0.00%
Sig prop	50.00%	0.00%	0.00%	50.00%	0.00%
dist p2p	0.00%	0.00%	0.00%	0.00%	0.00%
Dist p2p*	0.00%	0.00%	0.00%	0.00%	0.00%
Dist p2v	0.00%	0.00%	0.00%	0.00%	0.00%
Dist p2v*	0.00%	50.00%	50.00%	0.00%	0.00%
Dist p2av	0.00%	0.00%	0.00%	0.00%	0.00%
Dist p2av*	50.00%	50.00%	50.00%	50.00%	0.00%
W5					
baseline	0.00%	0.00%	0.00%	0.00%	0.00%
Sig prop	50.00%	0.00%	0.00%	100.00%	50.00%
dist p2p	0.00%	0.00%	0.00%	0.00%	0.00%
Dist p2p*	0.00%	0.00%	0.00%	0.00%	0.00%
Dist p2v	0.00%	0.00%	0.00%	0.00%	0.00%
Dist p2v*	50.00%	50.00%	50.00%	50.00%	0.00%
Dist p2av	0.00%	0.00%	0.00%	0.00%	0.00%
Dist p2av*	50.00%	0.00%	0.00%	50.00%	0.00%
W6					
baseline	0.00%	0.00%	0.00%	0.00%	0.00%
Sig prop	100.00%	0.00%	0.00%	100.00%	50.00%
dist p2p	0.00%	0.00%	0.00%	0.00%	0.00%
Dist p2p*	0.00%	0.00%	0.00%	0.00%	0.00%
Dist p2v	0.00%	0.00%	0.00%	0.00%	0.00%
Dist p2v*	100.00%	0.00%	0.00%	100.00%	50.00%
Dist p2av	0.00%	0.00%	0.00%	0.00%	0.00%
Dist p2av*	100.00%	0.00%	0.00%	100.00%	0.00%

Table A.9: Raw miss rate results for each window size (continued)

	s1	s2	s3	s4	s10
W7					
baseline	0.00%	0.00%	0.00%	0.00%	0.00%
Sig prop	100.00%	0.00%	0.00%	100.00%	100.00%
dist p2p	0.00%	0.00%	0.00%	0.00%	0.00%
Dist p2p*	0.00%	0.00%	0.00%	0.00%	100.00%
Dist p2v	0.00%	0.00%	0.00%	0.00%	0.00%
Dist p2v*	100.00%	0.00%	0.00%	100.00%	100.00%
Dist p2av	0.00%	0.00%	0.00%	0.00%	0.00%
Dist p2av*	100.00%	0.00%	0.00%	100.00%	100.00%
W8					
baseline	0.00%	0.00%	0.00%	0.00%	0.00%
Sig prop	100.00%	0.00%	0.00%	100.00%	100.00%
dist p2p	0.00%	0.00%	0.00%	0.00%	0.00%
Dist p2p*	0.00%	0.00%	0.00%	0.00%	100.00%
Dist p2v	0.00%	0.00%	0.00%	0.00%	0.00%
Dist p2v*	100.00%	0.00%	0.00%	100.00%	100.00%
Dist p2av	0.00%	0.00%	0.00%	0.00%	0.00%
Dist p2av*	100.00%	0.00%	0.00%	100.00%	100.00%
W9					
baseline	0.00%	0.00%	0.00%	0.00%	0.00%
Sig prop	100.00%	100.00%	100.00%	100.00%	100.00%
dist p2p	0.00%	0.00%	0.00%	0.00%	0.00%
Dist p2p*	0.00%	0.00%	0.00%	100.00%	100.00%
Dist p2v	0.00%	0.00%	0.00%	0.00%	0.00%
Dist p2v*	100.00%	100.00%	100.00%	100.00%	100.00%
Dist p2av	0.00%	0.00%	0.00%	0.00%	0.00%
Dist p2av*	100.00%	100.00%	100.00%	100.00%	100.00%
W10					
baseline	0.00%	0.00%	0.00%	0.00%	0.00%
Sig prop	100.00%	100.00%	100.00%	100.00%	100.00%
dist p2p	0.00%	0.00%	0.00%	0.00%	0.00%
Dist p2p*	100.00%	0.00%	0.00%	100.00%	100.00%
Dist p2v	0.00%	0.00%	0.00%	0.00%	100.00%
Dist p2v*	100.00%	100.00%	100.00%	100.00%	100.00%
Dist p2av	0.00%	0.00%	0.00%	0.00%	0.00%
Dist p2av*	100.00%	100.00%	100.00%	100.00%	100.00%

B All Publications Published, Submitted, and Planned

"Automatic Sentiment Shift Detection from Sentiment Signal Analysis", Kevin Labille and Susan Gauch, 2019. **Planned**

"Optimizing Statistical Distance Measures in Multivariate SVM for Sentiment Quantification," Kevin Labille and Susan Gauch, 2019. **Submitted**

"Text-Mining for Word Sentiment Detection," Kevin Labille, Susan Gauch, and Sultan Alfarhood, In Communications in Computer and Information Science, ed. Ana Fred et al., Springer, 2019. DOI:10.1007/978-3-319-99701-8_7, 149-170. **invited Book Chapter**

"Estimating Sentiment via probabilities and Information Theory," Kevin Labille, Sultan Alfarhood, and Susan Gauch, 8th International Conference on Knowledge Discovery and Information Retrieval (KDIR 2016), Porto, Portugal, November 9-11, 2016, 121-129. **Best Paper Finalist**