Short Text Classification with Tolerance Near Sets

by

Vrushang Patel

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Science

in the Department of Applied Computer Science

Short Text Classification with Tolerance Near Sets

by

Vrushang Patel

Supervisory Committee

Dr. Sheela Ramanna, Supervisor
(Department of Applied Computer Science)

Dr. Talal Halabi, Member
(Department of Applied Computer Science)

Dr. Ketan V. Kotecha, External Member
(Symbiosis Centre for Applied Artificial Intelligence (SCAAI), Symbiosis International, Pune, India)

ABSTRACT

Text classification is a classical machine learning application in Natural Language Processing, which aims to assign labels to textual units such as documents, sentences, paragraphs, and queries. Applications of text classification include sentiment classification and news categorization. Sentiment classification identifies the polarity of text such as positive, negative or neutral based on textual features. In this thesis, we implemented a modified form of a tolerance-based algorithm (TSC) to classify sentiment polarities of tweets as well as news categories from text. The TSC algorithm is a supervised algorithm that was designed to perform short text classification with tolerance near sets (TNS). The proposed TSC algorithm uses pre-trained SBERT algorithm vectors for creating tolerance classes. The effectiveness of the TSC algorithm has been demonstrated by testing it on ten well-researched data sets. One of the datasets (Covid-Sentiment) was hand-crafted with tweets from Twitter of opinions related to COVID. Experiments demonstrate that TSC outperforms five classical ML algorithms with one dataset, and is comparable with all other datasets using a weighted F1-score measure.

Keywords: Sentiment Classification, machine learning, tolerance near sets, Transformer, News Classification and Natural Language Processing.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

Learning is a continuous process and every journey you traverse brings you something new. I take this great opportunity to thank everyone, who has rendered his or her valuable support in my academic life at and out of the university. Many many thanks to Dr. Sheela Ramanna for giving me this opportunity to do my studies under her guidance. Support and encouragement from my professor Dr. Ramanna helped me accomplish my goals throughout my research. I am grateful for the knowledge, experience and guidance I received from her during this study. I would like to take this opportunity to acknowledge and express my gratitude to the professors in the Department of Applied Computer Science at the University of Winnipeg, for their invaluable instructions. The courses I attended under my professors were educative and showcased the opportunities and future possibilities. The knowledge I obtained from them was critical and essential in the success of my research. I would like to thank Dr. Talal Halabi who is not only my former professor but also my committee member and provided his guidance resulting in my first conference paper. I would also like to thank the Dean of Graduate Studies Dr. Mavis Reimer and the staff of Graduate Studies, Ms. Dagmawit Habtemariam, Dylan Jones and Mr. Eric Benson. I would like to show appreciation to Ms. Connie Arnhold from the ACS department. Lastly, I would say special thanks to Resty Jimenez and Ferdinand Borillo for providing a GPU machine for this research. Thank you to my family who consistently encouraged me and gave me strength in times of adversity and most importantly for having faith in me. Thank you to all my friends who have assisted me and helped me in ways that I cannot measure. Finally, I would like to thank everyone who has assisted me and supported me which has allowed me to achieve my goal. This research has been supported by the funds from Dr.Ramanna's NSERC Discovery Grant, and also by the Faculty of Graduate Studies, UW President's Distinguished Graduate Student Scholarship (PDGSS), UW Discretionary Grant, and UW Graduate Studies Travel Grant. We are very grateful to Professor James F. Peters, University of Manitoba who founded near set theory and gave me a chance to extend his research for text analysis. Last but not least, I would like to thank my parents and family for their support during my tough time

Vrushang Patel

DEDICATION

To my family for having faith in me,
to my professor for encouraging me,
to my friends for helping me,

Chapter 1

Introduction

The popularity of World Wide Web (WWW) and the internet users are increasing day by day[1]. Several websites allow users to share their views on internet via social media platforms. It creates an opportunity for researchers to analyze such massive data with the help of machine learning and deep learning techniques. The data can be a text, video, image or audio. The analysis of text data falls into the domain of Natural Language Processing (NLP). Text classification, also known as text categorization, is a classical problem in NLP, which aims to assign labels to textual units such as documents, sentences, paragraphs, and queries. It has a wide range of applications including sentiment analysis, news categorization, question answering, user intent classification, spam detection, content moderation, and so on [80]. Text data can come from many sources like email, social media, movie reviews, news headlines, chats, and questions and answers from customer services. There are several algorithms in machine learning that perform text classification.

Opinion mining and sentiment analysis are two popular NLP research areas that where the opinions of users are analyzed to detect sentiment polarity [36]. Polarity determination has been performed for product reviews, forums, blogs, news articles, and micro-blogs. The advent of microblogging platforms such as Twitter, having a large user-base, has led to a vast amount of information for sentiment analysis. The tasks of Twitter sentiment analysis include sentiment polarity detection, Twitter opinion retrieval, tracking sentiments over time [1], irony detection, emotion detection [12, 27, 58, 57, 69]. Due to the word limit of 280 characters, micro-blogs do not contain complete sentences. Moreover, micro-blogs often contain abbreviations and

[1]https://www.internetworldstats.com/emarketing.htm

noisy texts. Therefore, it needs standard pre-processing techniques such Parts-of-Speech (POS) tagging, removing of URLs, Hashtags, @usernames, stopwords, stemming, and spelling correction are applied to tweets due to the nature of messages posted by users. Twitter sentiment classification, which identifies polarity such as positive, negative, or neutral, is based on textual features: i) syntactic (e.g., n-grams, term frequencies, dependency trees), ii) semantic (e.g., opinion and sentiment words), and usually with the aid of lexicons, iii) stylistic (e.g., emoticons) and iv) twitter-specific features (e.g., hashtags and retweets). Two main challenges encountered with tweets are the length of the text (max. of 280 characters) and incorrect or improper use of language. Different machine learning approaches: supervised [37], semi-supervised [71] and unsupervised [35, 21] were applied to text classification which included well-known learning algorithms such as Naive Bayes (NB), Maximum Entropy (ME), and Support Vector Machines (SVM). Well-known sentiment lexicons such VADER (Valence Aware Dictionary for Sentiment Reasoning) [18] was developed which was an improvement over NLTK[2] and Textblob[3] tools. These methods used either sentiment lexicons and/or a variety of hand-crafted textual features for short text sentiment classification. Deep learning (DL) approaches that were initially used to learn word embeddings from large amounts of text corpus are now used extensively in sentiment analysis [11, 68, 79, 20].

1.1 Problem Definition

In this thesis, we consider the application tolerance near sets for short text classification. Near set theory was introduced by [43, 44]. Tolerance near sets [45] (TNS) provide an intuitive as well as a mathematical basis in defining what it means for pairs of objects to be *similar*. Near sets are disjoint sets that resemble each other. Resemblance is determined by considering set descriptions defined by feature vectors [42]. Descriptively near sets have been successfully applied in the following areas: content based image retrieval [15, 67], solar flare detection in images [47], audio signal and music genre classification [52, 63, 72] and community detection in social networks with the TCD algorithm [23, 19]. The TCD algorithm is now included in CDlib[4] which is Python software package that allows to extract, compare and evaluate com-

[2]https://www.nltk.org/
[3]https://textblob.readthedocs.io/en/dev/
[4]https://cdlib.readthedocs.io/en/latest/reference/algorithms.html#ensemble-methods.

munities from complex networks. In this work, set descriptions are textual features represented by vectors.

1.2 Proposed Approach

We address the short text classification problem by using tolerance classes from near set theory. The notion of tolerance is directly related to the idea of closeness between objects, such as image, audio or text segments that resemble each other with a tolerable level of difference. The term *tolerance space* was coined by Zeeman in modelling visual perception with tolerances [81]. Mathematically, a tolerance space $(X, \simeq)$ consists of a set X supplied with a binary relation $\simeq$ (*i.e.*, a subset $\simeq \subset X \times X$) that is reflexive (for all $x \in X$, $x \simeq x$) and symmetric (for all $x, y \in X$, $x \simeq y$ and $y \simeq x$) but transitivity of $\simeq$ is not required. The tolerance relation provides us with a mechanism for clustering text into groups termed as *tolerance classes*. The motivation for using tolerance classes is that the tolerance relation defines *similarity* rather than equivalence. The tolerance classes are directly induced from the feature vectors using the tolerance level ε and a distance function. The proposed sentiment/text classification pipeline is given in Figure 1.1 where feature vectors are generated using two pre-trained deep learning models BERT [10] and SBERT [54].

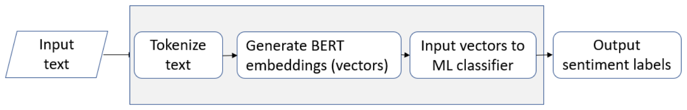

Input text
Tokenize text
Generate BERT embeddings (vectors)
Input vectors to ML classifier
Output sentiment labels

Figure 1.1: Text Classification Pipeline

1.3 Contributions

The contributions of this work are as follows [38]:

- We propose a novel algorithm (TSC) to perform short text classification using a tolerance form of near set theory. This is the first application of TNS in the NLP domain. The orginal supervised classifier was first introduced in [47].

- We provide a practical implementation of transformer-based vectors for our proposed TSC algorithm and the following supervised ML algorithms: Support Vector Machine (SVM), Maximum Entropy (ME), Random Forest (RF), Stochastic Gradient Descent (SGD), and Light Gradient Boosting Machine (LGBM) implementations from the Scikit-learn library [5]. In addition, TF-IDF-based vectors were also used for the above mentioned supervised ML algorithms.

- We present empirical evidence on the performance of our proposed TSC algorithm in terms of the size of the dataset as well as number of classes applied to ten well researched text datasets using a weighted F1-score.
- We handcrafted a dataset referred to as Covid-Sentiment which is a subset derived from [5] from the opinions of people related to COVID-19.
- Our proposed TSC algorithm outperforms all transformer vector-based supervised ML algorithms with three datasets, and outperforms TF-IDF-based supervised ML algorithms with two datasets. In addition, TSC algorithm achieves the best performance with a dataset having 20 classes.

1.4 Thesis Layout

The rest of this thesis organized as follows:

Chapter 2 provides an overview of previous work in text classification.

Chapter 3 provides a theoretical framework for the TSC algorithm with the transformer architecture.

Chapter 4 explains the ten various text datasets used as a case study in this thesis.

Chapter 5 gives experiments conducted on the datasets with various algorithms followed by a discussion of the results.

Chapter 6 concludes the thesis and provides future research directions.

[5]https://scikit-learn.org/stable/

Chapter 2

Related Work

There are three popular approaches for text classification: 1) Machine Learning-based approach, 2) Lexicon-based approach 3) Deep Learning-based approach. The text classification task is not only for Twitter, but it also considers news classification and reviews classification.

2.1 Machine Learning-based approach

The Machine Learning (ML) approach employs a ML method to classify sentiments of the text based on training data provided to it. These approached can be broadly classified as: supervised, unsupervised and semi-supervised learning. The supervised approach requires a training set to train the model and use a test set to evaluate performance of that model. The unsupervised approach finds hidden patterns from the data and assigns a sentiment based on the discovered pattern to the dataset. The semi-supervised approach is combination of supervised and unsupervised approaches. Supervised algorithms require a set of representative labeled data for building classification models. However, labeled data are usually difficult and expensive to obtain, which motivates the interest in semi-supervised learning. This type of learning uses both labeled and unlabeled data in the training process and is particularly useful in applications such as tweet sentiment analysis, where a large amount of unlabeled data is available.

2.1.1 Supervised Learning Approach

ML based text classification was pioneered in [37] by applying Naive Bayes (NB), Maximum Entropy (ME), and Support Vector Machine (SVM) for binary sentiment classification of movie reviews. For experiments, movie reviews were collected from IMDB.com where SVM outperformed all other algorithms with the highest accuracy of 82.9% with unigrams features. SVM was used to classify sentiments with different feature selection methods in [9]. Experiments were performed on 305 positive reviews and 307 negative reviews on digital cameras. SVM was trained on three features set based on domain free, domain dependent, and sentiment features. To reduce the number of features the Information Gain (IG) was applied. The reduced features set performed better on the multi-domain dataset than the digital camera dataset and yielded an accuracy of 84.15% for kitchen appliances.

The Restaurant reviews written in the Cantonese language were classified [83] by using NB and SVM. The authors studied the effects of feature representations and feature size on classification performance. Experiments were performed on 1500 positive and 1500 negative reviews. They experimented with different feature representations like unigram, unigram_freq, bigram, bigram_freq, trigram, trigram_freq, and a varying number of features in the range of 50 to 1,600 features. The highest accuracy reported was 95.67% using NB for 900–1100 features. The performance of three popular ensemble methods bagging, boosting, and random subspace based on five base learners, namely NB, ME, Decision Tree (DT), K Nearest Neighbour (KNN), and SVM for sentiment classification were studied in [74]. The authors experimented with ten different datasets and reported better accuracy over base learners at the cost of computational time.

A supervised learning approach was used in [13] where different n-gram features were combined with parts-of-speech (POS) tags. Three supervised algorithms were employed: NB and SVM trained with POS and unigrams, and ME trained with unigrams and bigrams. The ME algorithm outperformed the other two algorithms by almost 3%. IMDB Movie Review dataset and classification with a combination of unsupervised and supervised methods were introduced in [31] where TF-IDF vectorization was applied to the dataset. A semi-supervised learning method was used in [84] to Sina microblog data to annotate as well classify sentiments.

The ensemble classification on large-scale twitter dataset was applied in [30] with the help of linearly combined ME classifiers. Their research was only limited for one algorithm with only one dataset. The bootstrapping ensemble framework was proposed in [14] which was able to cope with class imbalance using unigram, bigram and POS for the analysis.

2.1.2 Unsupervised Learning Approach

One of the most fundamental modes of understanding and learning is organizing data into sensible grouping. The formal study of methods and algorithms for grouping, or clustering, objects according to similarities and same characteristics is *Cluster analysis*. The training data does not require labels in this type of learning. A sentiment analysis system named Document based Sentiment Orientation System was proposed in [62]. It uses an unsupervised approach which determines the sentiment orientation of the movie reviews and Word Net lexicon was used to identify synonyms and antonyms of opinion word list. Negation was also handled in the proposed system. The documents were classified as positive, negative or neutral. The approach provides a summary of the total number of positive and negative documents.

Authors in [24] applied natural language analysis on seven different languages from India using an unsupervised learning approach to perform text classficiation on an unlabelled dataset. An interesting unsupervised sentiment analysis method (ESSA) based on emotional signals was proposed in [17]. The emotional signals were divided into two categories: emotion correlation and emotion indication. The ESSA method was built on the orthogonal nonnegative matrix tri-factorization model. Two different datasets were used for the evaluation of the ESSA approach.

An unsupervised, lexicon-based classifier that estimates the level of emotional valence in text in order to make a prediction introduced in [35]. Classifier contains an extensive list of linguistically driven functionalities, such as: negation/capitalization detection, intensifier/diminisher detection and emoticon/exclamation detection, all of which contribute to the final prediction. The proposed algorithm was applicable in two different but complementary settings: opinion detection (i.e., detecting whether the text contains an expression of opinion or is objective) and polarity detection (i.e., predicting whether a subjective text is negatively or positively oriented), overall constituting a solution that can be applied without any modification or training to a wide set of environments and settings. Extensive experiments using real-world datasets from social Web sites and annotated by human assessors, demonstrate that the lexicon-based approach performs surprising well in the vast majority of cases, persistently outperforming state-of-the-art machine learning solutions, even in environments where there is a significant number of training instances for the latter.

Aspect-based sentiment analysis (ABSA) that is focused on the use of a knowledge base for the extraction of aspects and the use of grammatical relationships proposed in [21]. It was an unsupervised approach used with lexicon-based approach to classify sentiments. This research is based on comparison analysis of three sentiment lexicons to determine what is the best combination for polarity classification at aspect-level. Authors compared their algorithm with supervised approach to get performance of unsupervised appraoch.

2.1.3 Semi-supervised Learning Approach

The first work on sentiment classification that does not require labeled data was proposed in [71], in which a document is classified as either positive or negative by taking into account the average semantic orientation of its phrases that contain adjectives or adverbs. This approach was assessed on automobile reviews and movie reviews, which are data sources that are very different from the type of short texts found in tweets.

An improved semi-supervised learning framework for twitter sentiment classification was introduced in [8]. Authors combined unsupervised information, captured from a similarity matrix constructed from unlabeled data, with a classifier. The authors state that such a similarity matrix is a powerful knowledge-discovery tool that can help to classify unlabeled tweet sets. Their framework makes use of the $C^3E - SL$ Self-training algorithm to induce a better tweet sentiment classifier. Experimental results demonstrate that their algorithm outperformed the lexicon-based and stand-alone SVM algorithms on real world datasets.

A topic-based modeling for sentiment analysis with semi-supervised approach was introduced in [78]. The authors performed cluster analysis and several classification phases on the same dataset (a training set). As a result, a sentiment mixture model was obtained and then used to predict the class of unlabeled tweets, from which a subset was chosen to augment the training set. The authors considered a large unlabelled data of 2 million tweets and 9684 labelled tweets to report their results. As the topic structure formed by clustering comes exclusively from the training set, an important gap in this approach is that no topic analysis takes place in the unlabeled tweets, where this supplementary information could be useful.

2.2 Lexicon-based Sentiment analysis approach

Lexicon-based methods leverage lists of words annotated by polarity or polarity score to determine the overall opinion score of a given text. The main advantage of these methods is that they do not require training data. One of the most well-known lexicon-based algorithms developed for social media is SentiStrength introduced in [70]. SentiStrength can effectively identify the sentiment strength of informal text including tweets using a human-coded lexicon that contains words and phrases that are frequently confronted in social media. Apart from the sentiment lexicon that contains about 700 words, SentiStrength uses a list of emoticons, negations, and boosting words to assign the sentiment to a text. Initially, the algorithm was tested on MySpace comments. SentiStrength was compared with many machine-learning approaches and tested on six different datasets, including a dataset with tweets posts.

A Lexicon-based model VADER (Valence Aware Dictionary for Sentiment Reasoning) was introduced in [18] for sentiment classification. It is a simple rule-based model for general sentiment analysis, and compares its effectiveness to eleven typical state-of-practice benchmarks. Using a combination of qualitative and quantitative methods, they first constructed and empirically validated a gold standard list of lexical features along with their associated sentiment intensity measures, which are attuned explicitly to sentiment in microblog like contexts and combining these lexical features with consideration for five general rules that embody grammatical and syntactical conventions for expressing and emphasizing sentiment intensity. VADER is able to get better results only for short microblogs like tweets. In addition, a well-known sentiment lexicon for microblog-like contexts was also developed, which was an improvement over NLTK[1] and Textblob[2] tools. In [59], semantic patterns were used as features for the classification of tweets using the concept of a bag of senti-circles to capture contextual semantic similarities among words. The above methods used either sentiment lexicons and/or various hand-crafted textual features for Twitter sentiment classification.

[1]https://www.nltk.org/

[2]https://textblob.readthedocs.io/en/dev/

2.3 Deep Learning-based approach

Feed-forward networks are among the simplest deep learning models for text representation. These models view a text as a bag of words. For each word, they learn a vector representation using an embedding model such as Word2vec [32], and Glove [41], by taking vector sum or average of the embedding as a representation of text and passing it through many feed-forward layers called as Multi-Layer Perceptron(MLP).

Recurrent Neural Networks (RNN) based models view text as a sequence of words, and these models are intended to capture word dependencies and text structures for text classification. Among many variants to RNNs, Long Short-Term Memory (LSTM) is the most well-known architecture designed to capture long-term dependencies better. LSTM introduces memory cells to remember values by using three gates. It has input gate, output gate and forget gate to obtain input/output from the memory cells and addresses the gradient vanishing or exploding problems. A Tree-LSTM model was introduced in [66] which is a generalization of LSTM to tree-structured network typologies to learn rich semantic representations. The Tree-LSTM is a better model than the traditional chain-structured LSTM for NLP tasks. The authors validated effectiveness of Tree-LSTM on sentiment classification and predicting the semantic relatedness of two sentences. RNNs face difficulties remembering long-range dependencies but they are good at capturing the local structure of a word sequence.

To overcome the limitations of deep networks such as RNN and its variants Long Short Term Memory (LSTM) and Gated Recurrent Unit (GRU) both in terms of memory and sequential execution, attention-based Transformer Architecture BERT (BERT: Bidirectional Encoder Representations from Transformers) was introduced by [73] and later improved by [10]. Attention is motivated by how we pay visual attention to different regions of an image or correlate words in one sentence. Attention has become an increasingly popular concept and helpful tool in developing Deep Learning models for NLP. In a nutshell, attention in language models can be interpreted as a vector of importance weights. To predict a word in a sentence, the attention vector how strongly it is correlated with, or “attends to,” other words is estimated and the sum of their values weighted by the attention vector is taken as the approximation of the target.

These models rely primarily on transfer learning and pretrained models. BERT has several variants such as RoBERTa, ELECTRA, DistilBERT, and ALBERT [33].

In this thesis, we used the vectors generated by SBERT and Distil-BERT [60] was used for classification experiments. SBERT (Sentence-BERT), which is a modification of the pre-trained BERT network [54].

Chapter 3

Theoretical Framework, Tolerance Class Learner and Vector Generation Methods

In this chapter, a brief introduction to the main definitions underlying near sets is given. In addition, we give the algorithm used to generate the tolerance classes and representative vectors. We also briefly describe the specific transformer model used in our experiments.

3.1 Rough Sets and Tolerance Rough Sets

Rough Set theory was introduced by Zdzisław Pawlak during the early 1980s [39] as a mathematical framework for reasoning about ill-defined objects and applied to many areas machine learning and knowledge discovery [40]. In *classical* rough set theory a universe of objects is partitioned into disjoint classes or granules i.e., *equivalence classes* by using an equivalence relation. Given a concept that is determined to be vague (not precise), this theory makes it possible to express the vague concept by a pair of precise concepts called the lower and the upper approximation. A vague concept is defined as a category (or decision) that cannot be properly classified. Informally, the elements of the lower approximation can be classified with certainty. However, the elements of the upper approximation may not be classified with certainty. Figure 3.1 shows the regions that emerge using the two set approximation operators namely lower approximation $\mathcal{L}_{\mathcal{A}}(X)$ and $\mathcal{U}_{\mathcal{A}}(X)$ where X represents the concept (example) that needs to be classified. The regions are depicted as squares only for the sake of illustration, but they can be of arbitrary shape. We should note that each granule can contain an arbitrary number of objects or may be empty [51]. The difference between these two regions (also known as the boundary region) contains objects that may belong to more than one category or class or concept.

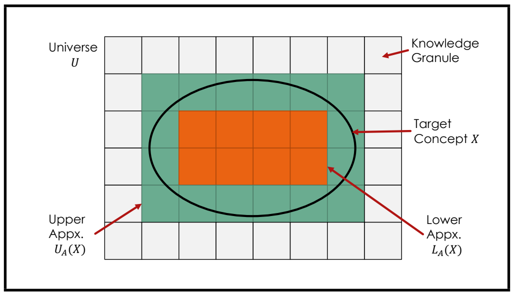


Figure 3.1: Rough sets and set approximation

However, there are certain applications such as document classification [16] and natural language processing [61, 34] where overlapping classes are necessary with a tolerance form of rough sets [64]. A *tolerance model* uses a tolerance relation instead of equivalence where the transitivity property is relaxed which enables overlapping classes (in other words, objects can belong to more than one class). Tolerance relations can be considered as generalizations of equivalence relations [48]. A history of tolerance rough sets from an axiomatic point of view can be found in [51].

3.2 Preliminaries - Tolerance Near Sets

Near set theory was influenced by rough set theory [43, 44]. Disjoint sets containing objects with similar descriptions are near sets provided the intersection of the sets is nonempty. Near sets do not require set approximation operators (lower and upper approximation). Tolerance near sets [45] provide an intuitive as well as mathematical basis in defining what it means for pairs of objects to be *similar*. The basic structure which underlies near set theory is a perceptual system which consists of perceptual objects (i.e., objects that have their origin in the physical world [44]). Near sets are characterized by i) a perceptual system ii) a nearness relation and iii) near set [77].

Definition 1. Perceptual System [44]
A perceptual system is a pair $\langle O, F \rangle$, where O is a nonempty set of perceptual objects and F is a countable set of real-valued probe functions $\phi_i : O \to \mathbb{R}$.

An object description is defined by means of a tuple of probe function values $\Phi(x)$ associated with an object $x \in X$, where $X \subseteq O$ as defined by Eq.3.1:

$$\Phi(x) = (\phi_1(x), \phi_2(x) \ldots, \phi_n(x)) \tag{3.1}$$

where $\phi_i : O \to \mathbb{R}$ is a probe function of a single feature. The probe functions give rise to a number of perceptual relations. This approach is useful when decisions on nearness are made in the context of a perceptual system i.e., a system consisting of objects and our perceptions of what constitutes features that best describe these object. The notion of tolerance is directly related to the idea of closeness between objects, that resemble each other with a tolerable level of difference. A tolerance space $(X, \simeq)$ consists of a set X endowed with a binary relation $\simeq$ (*i.e.*, a subset $\simeq \subset X \times X$) that is reflexive (for all $x \in X$, $x \simeq x$) and symmetric (for all $x, y \in X$, $x \simeq y$ and $y \simeq x$) but transitivity of $\simeq$ is not required.

Definition 2. Perceptual Tolerance Relation[45]
Let $\langle O, F\rangle$ be a perceptual system and let $\mathcal{B} \subseteq F$,

$$\cong_{\mathcal{B},\epsilon} = \{(x, y) \in O \times O : \parallel \phi(x) - \phi(y) \parallel_2 \leq \varepsilon\} \tag{3.2}$$

where $\|\cdot\|_2$ denotes the L^2 norm of a vector.

Definition 3. Text-based Tolerance Relation $\cong_{\mathcal{T},\epsilon}$
Let $\langle T, F\rangle$ be a perceptual system for a nonempty set of objects T and let $\mathcal{T} \subseteq F$ where $\mathcal{T}$ represents textual features. A tolerance space $\langle T, \cong_{\mathcal{T},\epsilon}\rangle$ is defined as:

$$\cong_{\mathcal{T},\epsilon} = \{(t_i, t_j) \in T \times T : \, dist(t_i, t_j) \leq \varepsilon\} \tag{3.3}$$

where $dist(t_i, t_j)$ denotes the Cosine distance of a vector given in Eq. 3.4. The tolerance relation $\cong_{\mathcal{T},\epsilon}$ induces a tolerence class TC where ε is a user-defined tolerance level.

$$dist(t_i, t_j) = \frac{\phi(t_i).\phi(t_j)}{\|\phi(t_i)\| \, \|\phi(t_j)\|} \tag{3.4}$$

In other words, given a set of text (objects) T, where $t_i \in T$, $i \in N$, each tweet or text t_i can be represented as a k-dimensional word vector $\phi(t_i)$ where text similarity is measured using the cosine distance measure.

Remark 3.2.1. *Our universe of text T described by set of vectors ϕ, is spread amongst tolerance classes with a tolerance level ε for semantic textual similarity, where a tolerance class TC is maximal with respect to inclusion [75].*

3.2.1 Examples of TNS

In this section, we illustrate our formal model with examples (sentences) drawn from the UCI[1] sentence dataset.

Example 1. $T = \{t_0, t_1, t_2, t_3, t_4\}$ *is set of tweets elaborated as:*

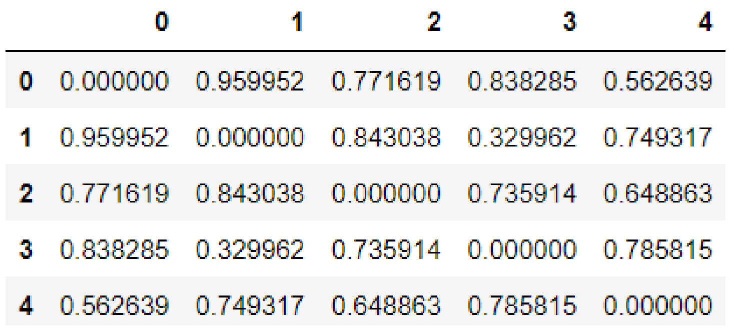
T = {"So there is no way for me to plug it in here in the US unless I go by a converter.", " Good case, Excellent value.", " Tied to charger for conversations lasting more than 45 minutes.MAJOR PROBLEMS!!", "He was very impressed when going from the original battery to the extended battery.", "The design is very odd, as the ear clip is not very comfortable at all."}

Example 2. *The Φ is set of 768 dimensional SBERT vectors $\phi(t_i)$ representation for the set T.*

Φ = { [7.22507477e-01, 1.02917385e+00,....., -6.84888303e-01, 2.86179781e-01], [-2.96921253e-01, 7.52622932e-02,...., -4.93599415e-01, -1.82238102e-01], [1.55841690e-02, 7.04942644e-02,...,3.76997262e-01, -2.59568900e-01], [7.95753375e-02, -2.57343829e-01,..., 6.54604957e-02, -2.85186619e-01], [4.41945344e-01, 5.39983749e-01,....,-4.36222434e-01, 1.82313472e-01] }

Example 3. *Figure 3.2 shows the cosine distance matrix created by using Eq.3.4.*

[1]https://archive.ics.uci.edu/ml/datasets/Sentiment+Labelled+Sentences



	0	1	2	3	4
0	0.000000	0.959952	0.771619	0.838285	0.562639
1	0.959952	0.000000	0.843038	0.329962	0.749317
2	0.771619	0.843038	0.000000	0.735914	0.648863
3	0.838285	0.329962	0.735914	0.000000	0.785815
4	0.562639	0.749317	0.648863	0.785815	0.000000

Figure 3.2: Distance matrix for T

Example 4. *Based on the distance matrix shown in Figure 3.2, and $\varepsilon \leq 0.79$, we can derive the following following tolerance classes:*
$dist(t_0, t_4) = 0.56$
$dist(t_4, t_2) = 0.65$
$dist(t_0, t_2) = 0.77$
since all of the pairs satisfy $\varepsilon \leq 0.79$ which creates the first tolerance class TC_0.
$TC_0 = \{t_0, t_2, t_4\}$.
$dist(t_1, t_3) = 0.32$
$TC_1 = \{t_1, t_3\}$ *which creates the second tolerance class.*
$dist(t_2, t_4) = 0.65$
$dist(t_2, t_3) = 0.74$
$dist(t_4, t_3) = 0.786$
$TC_2 = \{t_2, t_3, t_4\}$ *which creates the third tolerance class.*
The final tolerance class is union of all above classes TC_0, TC_1, TC_2.
$TC = \{\{t_0, t_2, t_4\}, \{t_1, t_3\}, \{t_2, t_3, t_4\}\}$

Remark 3.2.2. *The set of all text objects with similar value vectors creates similarity classes TC_0, TC_1, TC_2. The tolerance space T is covered by overlapping classes where tweets t2, t3 and t4 appear in more than one class.*

Remark 3.2.3. *The higher ε values increases the number of tolerance classes and lower tolerance value decreases the number of tolerance classes. We can get best ε value by only experiments. Algorithm 1 gives the method for generating tolerance classes from the training set and Algorithm 2 gives the method for generating tolerance classes from the test set. In classification, we consider labelled text t_i where s_i is a sentiment label for t_i and $s_i \in \{positive, negative\}$ for a two-class classification.*

3.3 Transformer

One of the computational disadvantages suffered by RNNs is the sequential processing of text. RNN cannot parallelize text processing. Although CNNs are less sequential than RNNs, the computational cost to capture relationships between words in a sentence also increases with the sentence's increasing length, similar to RNNs. Transformers [73] overcome this limitation by applying self-attention to compute in parallel for every word in a sentence or document an "attention score" to model the influence each word has on another.[2] Due to this feature, Transformers allow for much more parallelization than CNNs and RNNs, which makes it possible to efficiently train huge models on large amounts of data on GPUs.

Since 2018, there is a rise of a set of large-scale transformer-based pre-trained language models (PLMs). Compared to earlier contextualized embedding models based on CNNs [6] or LSTMs [46], Transformer-based PLMs use much deeper network architectures (e.g., 48-layer transformers [50]), and are pre-trained on much more significant amounts of text corpora to learn contextual text representations by predicting words conditioned on their context. These PLMs are fine-tuned using task-specific labels, and have created a new state of the art in many downstream NLP tasks, including text classification. Although pre-training is unsupervised (or self-supervised), fine-tuning is supervised learning. A recent survey [49] categories popular PLMs by their representation types, model architectures, pre-training tasks, and downstream tasks.

PLMs can be grouped into two categories, autoregressive and autoencoding PLMs. One of the earliest autoregressive PLMs is OpenGPT [50], a unidirectional model that predicts a text sequence word by word from left to right (or right to left), with each word prediction depending on previous predictions. One of the most widely used autoencoding PLMs is BERT [10]. Unlike OpenGPT, which predicts words based on previous predictions, BERT is trained using the masked language modeling (MLM) task that randomly masks some tokens in a text sequence, and then independently recovers the masked tokens by conditioning on the encoding vectors obtained by a bidirectional Transformer. There have been numerous PLMs over the BERT Base vector embeddings. Reimers et al. [54] introduced SBERT to get better semantically meaningful vector embeddings.

[2]Transformer is an instance of hybrid models, since each Transformer layer is a composite structure consisting of a feed-forward layer and a multi-head attention layer.

3.3.1 Vector Embeddings with SBERT

Sentence-BERT (SBERT), a modification of the pre-trained BERT network that uses siamese and triplet network structures to derive semantically meaningful sentence embeddings that can be compared using cosine-similarity. SBERT is fine-tuned on SNLI [3] and the Multi-Genre NLI [76] data, which creates sentence embeddings that significantly outperforms other state-of-the-art sentence embedding methods like InferSent [7] and and Universal Sentence Encoder [4]. The SBERT architecture is shown in 3.5.

The sentence is tokenized in first step and each token get its respective vector embedding, and then positional encoding [73] is added to those vectors based on the position of the token. The odd positions are added with cosine equation 3.6 and even positions are added with sin equation 3.5. In the equations 3.5 and 3.6 the d_{model} is dimension of the model which is 768 for SBERT.

$$PE_{(pos,2i)} = \sin(pos/1000^{2i/d_{model}}) \tag{3.5}$$

$$PE_{(pos,2i+1)} = \cos(pos/1000^{2i/d_{model}}) \tag{3.6}$$

The linear layer adds final positional encoding with the BERT tokenized embedding. These vectors are projected into three linear projections Query (Q), Key (K), and Value (V). The Self-Attention [73] computed by the Key, Query, and Value parameters of the transformers by using softmax function shown in equation 3.7.

$$Self - Attention(Q, K, V) = softmax(\frac{Q * K^T}{\sqrt{d_k}}) * v \tag{3.7}$$

There are 12 Self-Attention blocks in the SBERT where each block is pair of Self-Attention and Feed Forward Network (FFN). The output of Self-Attention blocks works as input for the FFN. FFN learns the representation of the word, and it improves every time with different weights of the network. There are several black-boxes in the transformer architecture. After getting output from Decoder, the SBERT uses Mean Pooling to derive semantically more meaningful vectors than BERT.

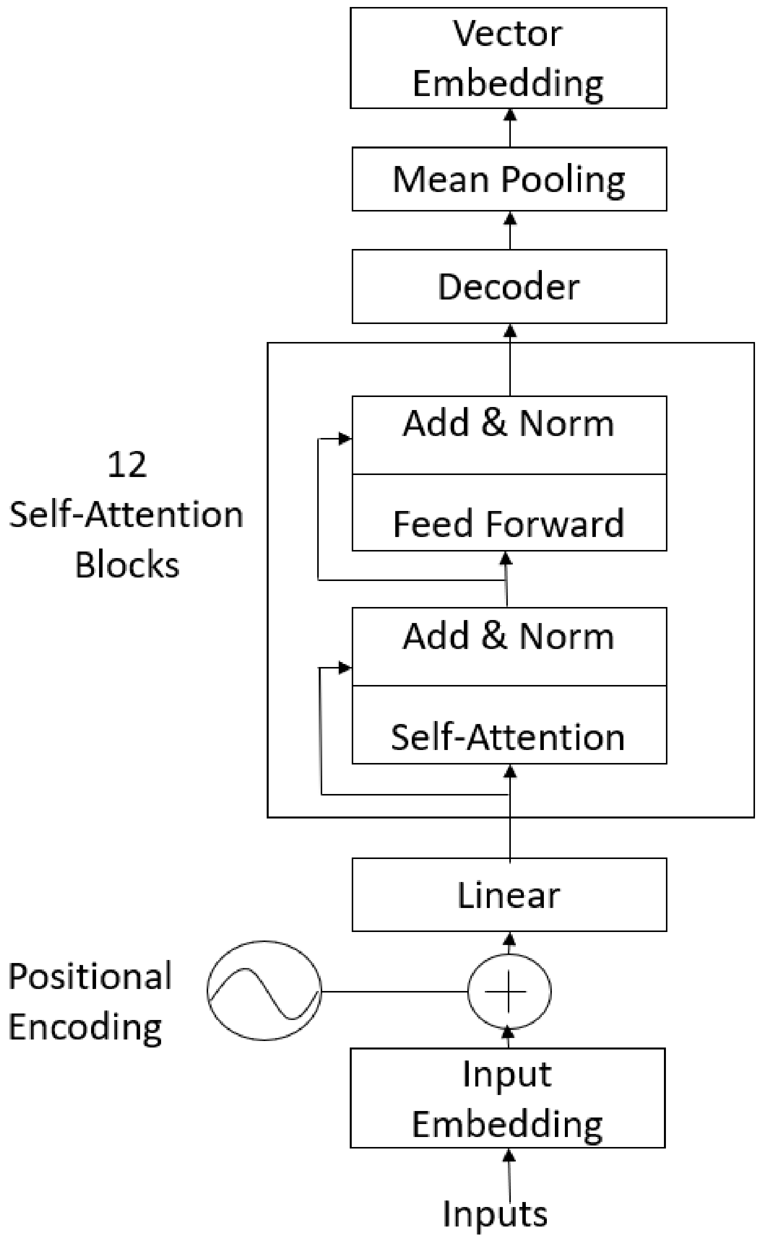


Figure 3.3: SBERT Architecture for vector embedding

3.3.2 Importance of position encoding

The position encoding shown by Equations 3.5 and 3.6 is an advantage of the transformer architecture over other embedding algorithms. It is also one important reason to choose the transformer embedding over other embedding options for the TSC vector embedding. Due to positional encoding, transformers can understand the meaning of the sentence with a position of that word. Figure 3.4 shows the comparative analysis of cosine distance with Spacy, SBERT, and RNN-based Glove model. The comparison is shown with two sentences where they have different meanings but similar words. If the distance between two words is high, then it will have lower similarity, and only SBERT provides a higher distance compare to other two approaches.

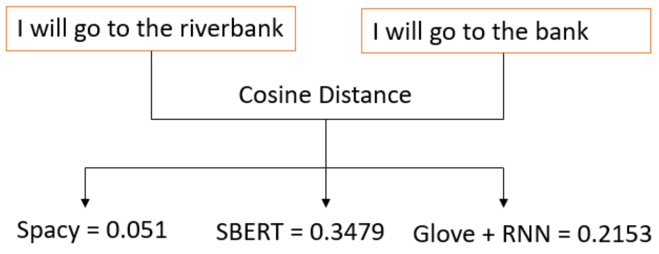


Figure 3.4: Cosine Distance comparison with other embeddings

3.4 Tolerance Sentiment Classifier - TSC

In this section, we present our proposed Tolerance Sentiment Classifier (TSC) in terms of a supervised learning algorithm in two parts. We first give a high-level flow chart of the TSC classifier shown in Fig. 3.5 followed by the discussion of the algorithms 1 and 2.

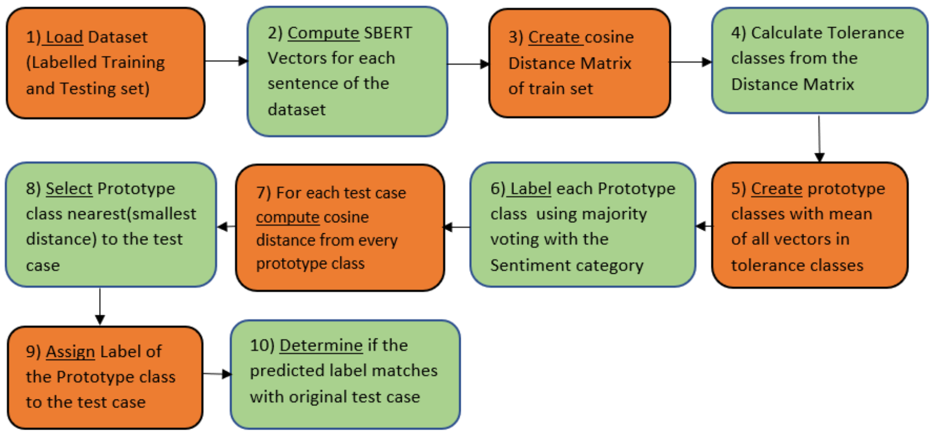

1) Load Dataset (Labelled Training and Testing set)
2) Compute SBERT Vectors for each sentence of the dataset
3) Create cosine Distance Matrix of train set
4) Calculate Tolerance classes from the Distance Matrix
5) Create prototype classes with mean of all vectors in tolerance classes
6) Label each Prototype class using majority voting with the Sentiment category
7) For each test case compute cosine distance from every prototype class
8) Select Prototype class nearest(smallest distance) to the test case
9) Assign Label of the Prototype class to the test case
10) Determine if the predicted label matches with original test case

Figure 3.5: Highlevel Flow chart showing overview of TSC

3.4.1 Phase I: Creating Prototype Classes

In the *training phase*, given a tolerance level ε, tolerance classes are induced from the training set, and the representative of each tolerance class is computed. The representative class is determined as the mean value of the feature vector. The text category is determined based on majority voting. In the *testing phase*, the cosine distance is computed for each element in the test set with all of tolerance class representatives obtained in the training phase and assigned the category of the tolerance class representative based on the lowest distance value.

Algorithm 1: Training Phase: Generating class representative vectors

```
Input  : $\varepsilon > 0$, // Tolerance level
         $TR = \{TR_1, \ldots, TR_M\}$ , // Training Data Set
Output: $(NC, \{R_1, \ldots, R_{NC}\}, \{TextCat_1, \ldots, TextCat_{NC}\})$
1  for $i \leftarrow 1$ to $M$ do
2      for $j \leftarrow i + 1$ to $M$ do
3          computeCosine($TR_i, TR_j, Cosine_{ij}$);
4  for $i \leftarrow 1$ to $M$ do
5      for $j \leftarrow i + 1$ to $M$ do
6          generateToleranceclass(Cosine$_{ij}$, $\varepsilon$; SetOfPairs);
             computeNeighbour(SetOfPairs, $i, TR; N_i$); // Compute the
             neighbourhood $N_i$ of $i^{th}$ training data $TR_i$
7          for all $x, y \in N_i$ do
8              if $x, y \notin SetOfPairs$ then
9                  $C_i \leftarrow N_i$; // Include $y$ from class $N_i$ into $C_i$
10     $H \leftarrow H \cup \{C_i\}$;
11     // $C_i$ is one tolerance class induced by the tolerance relation
12     computeMajorityCat($C_i; TextCat_i$); // Determine Category by majority
         voting for each $C_i$
13 $NC \leftarrow |H|$; // Number of classes
14 // End of defineClass
15 defineClassRepresentative($NC, H; \{R_1, \ldots, R_{NC}\}, \{TextCat_1, \ldots, TextCat_{NC}\}$);
     // Based on mean value
```

The algorithms use the following notation:
$procedure\_name(input_1, ..., input_n; output_1, ..., output_m)$.

The cosine distance between each pair of training set elements is computed using *computeCosine*. Next, by using the *generateToleranceclass* function, a set of training set pairs within the tolerance level ε is obtained.

For a training set element TR_i, its neighbourhood N_i is composed of all training set elements within the tolerance level ε of TR_i, including TR_i computed by the *computeNeighbour* function. If $N_i = \{TR_i\}$ then TR_i is a tolerance class with only one element. If a pair of neighbors in N_i does not satisfy the tolerance relation, the corresponding element is excluded from the tolerance class C_i. In addition, for each *representative* tolerance class C_i, the category information is obtained by a majority vote. Lastly, the $defineClassRepresentative$ function computes the representatives (prototype) of each of the NC tolerance classes and assigns the majority class. The tolerance class prototype is a vector whose values are computed as the *mean* of the probe function values of those belonging to that class.

3.4.2 Phase II: Text Classification

In the classification phase, TSC uses class representatives of the training set and their associated text category. The $computeCosine$ function operates on the testing set data and the representative class data. The *DetermineTextCat* function chooses the representative class that is closest to the test set element. The chosen representative class category will determine the text category of the test set element.

The complexity of the algorithms is fairly straight forward to determine. In phase 1, the complexity of $computeCosine$ functions is $\mathcal{O}(n^2)$. The complexity of *generateToleranceclass* function is $\mathcal{O}(n)$. In phase 2, the complexity of *DetermineTextCat* function is $\mathcal{O}(n)$. The complexity of TSC is reduced by using some advanced python programming packages like ”pandas”[3], ”numpy”[4] and ”scipy”[5]. These packages are highly optimized and reduce the requirements of for loops.

3.5 TF-IDF Vectorization

Term Frequency-Inverse Document Frequency (TF-IDF) is evolved from IDF, which is proposed by Sparck Jones [22, 55] with heuristic intuition that a query term that occurs in many documents is not a good discriminator and should be given less weight than one which occurs in few documents. Equation 3.8 is the classical formula of TF-IDF used for term weighting. The values in the feature vectors are weighted to reflect

[3]https://pandas.pydata.org/
[4]https://numpy.org/
[5]https://www.scipy.org/

Algorithm 2: Assigning Phase: Assigning Sentiment Classes

```
Input  : $\varepsilon > 0$, // Tolerance level
         $TS = \{TS_1, \ldots, TS_M\}$ , // Testing Data Set
         $\{R_1, \ldots, R_{NC}\}, \{TextCat_1, \ldots, TextCat_{NC}\}$ // Representative
Class and their associated categories
Output: $(TS' = \{TS'_1, \ldots, TS'_M\})$ // Testing Data Set with assigned
         categories
1 for $i \leftarrow 1$ to $M$ do
2     for $j \leftarrow i + 1$ to $NC$ do
3         computeCosine($TS_i, RC_j$, Cosine$_{ij}$);
4 DetermineTextCat($Cosine_{ij}; TS'$) // Computes min. distance and assigns
   category
```

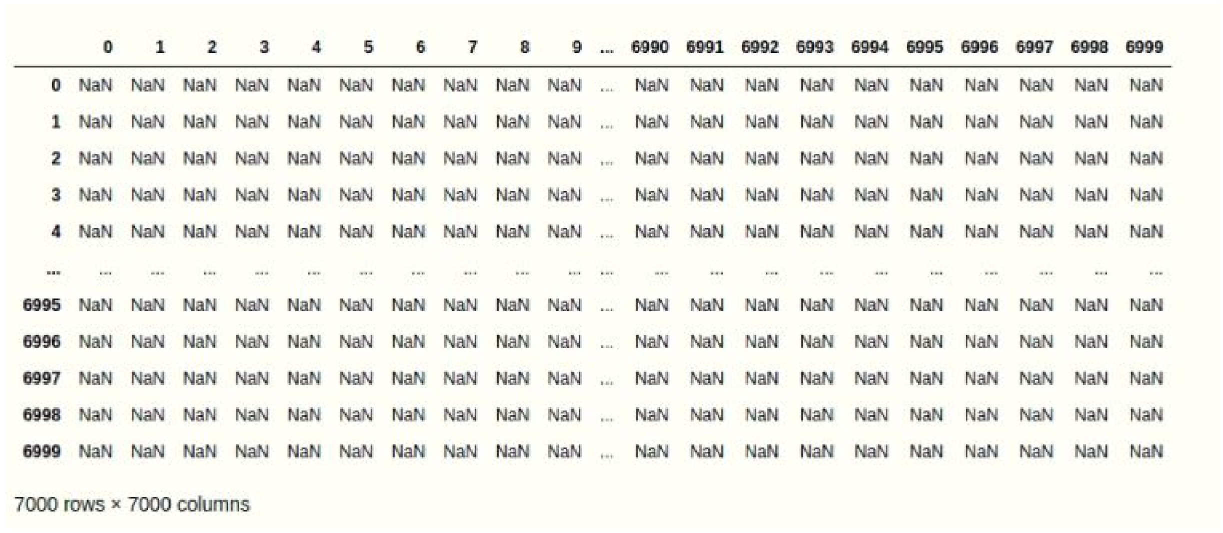
the frequency of words in the documents and the distribution of words across the collection. The more times a word/term occurs in a sentence, the more it is relevant to the text. The more times the word occurs throughout the document in the collection, the more poorly it discriminates between sentences. The ”TfidfVectorizer”[6] module from ”sklearn” package was used for generating these vectors from the text data.

$$W_{i,j} = tf_{i,j} * \log(\frac{N}{df_i}) \tag{3.8}$$

where $W_{i,j}$ is the weight for term i in document j, N is the number of documents in the collection, $tf_{i,j}$ is the term frequency of term i in document j, and df_i is the document frequency of term i in the collection.

For comparison analysis, TF-IDF Vectors are used with other machine learning algorithms like Support Vector Machine (SVM), Random Forest (RF), Light Gradient Boosting Machine (LGBM), and Stochastic Gradient Descent (SGD). The TF-IDF Vectors are not applicable for the TSC because it generates null values for cosine distance Matrix. The cosine distance matrix of TF-IDF vectors is shown in Figure 3.6.

[6]https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html



	0	1	2	3	4	5	6	7	8	9	...	6990	6991	6992	6993	6994	6995	6996	6997	6998	6999
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6995	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6996	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6997	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6998	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6999	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

7000 rows × 7000 columns

Figure 3.6: cosine distance matrix of TF-IDF vectors for Covid-Sentiment dataset

Chapter 4

Dataset

We have used ten datasets to analyze the performance of TSC with other ML algorithms. The datasets are a mix of long and short words as well as several sentiment classes. The majority of the datasets are benchmark datasets that are commonly used to perform NLP tasks.

4.1 Covid-Sentiment Dataset

We handcrafted a dataset referred to as Covid-Sentiment which is a subset derived from [5] using Tweets ID for 1^{st} April 2020 and 1^{st} May 2020. Twitter is a rich source of global information. It has approximately 192 million daily active users and users do an average of 500 million tweets per day[1]. To get a specific source of information from such a huge database, Twitter provides a hashtag feature to get specific information-related tweets. The hashtag involves adding # before an unbroken word or phrase. We used Twitter's Developer API to scrap tweets related to the COVID-19 hashtags such as #Covid19, #WHO, #Wuhan, #Corona. Algorithm 3 shows steps for the tweet scraping method in Python 3[2].

We extracted 47,386 tweets with the help of Twitter API. The tweets in languages other than English (ex: French, Hindi, Mandarin, Portuguese) were removed. Extensive pre-processing of 29,981 English language tweets from the original dataset such as removal of HTML tags, @Username, Hashtags, URLs, and incorrect spellings were also performed. A significant amount of time was spent on hand labeling of 8003

[1]https://www.dsayce.com/social-media/tweets-day/
[2]https://www.python.org/downloads/

Algorithm 3: Tweet scraping

```
Input  : Consumer-Key = 'xxxxxxxxxx'
         Consumer-Secret = 'xxxxxxxxx'
         Access-token = 'xxxxxxxxx'
         Access-token-Secret = 'xxxxxxxxx'
         Tweet_ID
Output: Return tweet information
1 for $i \leftarrow 1$ to $length(Tweet\_ID)$ do
2     $tweets = tweepy.Cursor(api.search, q = Tweet\_ID[i]).items(count)$;// get the tweets from api
3     for $tweet \leftarrow 1$ to $length(tweets)$ do
4         $tweets\_list = [tweet.text]$
5     // extract text of the tweets
```

tweets into 3 categories: positive, negative, and neutral. These sentiments are shown in Table 4.2. The average length is 12 words per tweet for this dataset.

The scope of this dataset is at the global level. The tweets are from various geographic regions and represent different issues for different countries during the pandemic time. Some countries were having problems with people dying and others were having trouble with the lockdown. Majority of tweets in this dataset were related to global politics, health, and economy from countries like U.S.A., India, Canada, Australia, etc. Since the focus of different countries was on different issues related to COVID, the content of these tweets showed less semantic similarity in the dataset. Some examples of tweets are shown in the Table 4.1.

The original tweets required a lot of pre-processing. A tweet contains @Username, hashtags, URLs, and stopwords which does not contribute to the semantic content of a tweet for the sentiment classification task. It creates a new word in the dictionary that does not exist for example #WeAreWithYou. Tweets also have a lot of grammar and spelling mistakes that should be corrected, otherwise each spelling mistake is considered as a new word for the vocabulary and will generate a different vector representation for its classification. Python regex module, and NLTK stemming and lemmatization were used in pre-processing before generating vector embedding for this dataset.

Table 4.1: Covid-Sentiment Tweet Examples

Tweet Sentiment	**Tweet**
Negative	RT @ANI: #WATCH Delhi Police release a video of its warning to senior members of Markaz Nizamuddin to vacate Markaz &amp follow lockdown guide
Negative	RT @shay_aiko Ismail Mohamed Abdulwahab from Brixton... no underlying health conditions. Everyone is at risk! in USA thaks to @Trump
Positive	RT @ANI On the appeal of Prime Minister Narendra Modi to combat #COVID19, all paramilitary forces personnel have contributed their one day
Neutral	RT @MrsGandhi As per Tamil Nadu's Health Secretary, a total 50 people from their state who attended the #TablighiJamaat gathering

4.2 Other Datasets

U.S. Airline sentiment: The U.S. Airline sentiment dataset [53] consists of tweets of reviews by passengers for five different US Airlines that are: United, US Airways, Southwest, Delta, and Virgin America. The average length of tweets is 10 words per tweet. The original dataset included 14,621 tweets and the pre-processed dataset of 13,000 tweets were used after the removal of duplicate and short tweets. The tweets were a mix of positive, negative, and neutral sentiments. The tweets are pre-labeled with the type of sentiment. This dataset has more negative reviews compared to the other two sentiments. Table 4.2 gives the sizes of the training and testing sets.

The IMDB Movie Review dataset: IMDB dataset [31] consists of an average of 230 words per review. It has the longest length among all other datasets. We used a subset of 22,000 reviews of the original dataset consisting of 50,000 reviews. The sentiment label distribution is balanced with each subset of the data.

The Stanford Sentiment Treebank (SST-2) dataset: SST-2 dataset consists [65] of phrases with fine-grained sentiment labels in the parse trees of 11,855 sentences from movie reviews. The original dataset included 69,723 phrases and only 16,500 examples were used. The average length of the text in this dataset is 10 words per sentence.

The Sentiment140 dataset: Sentiment 140[3] was created by using emoticons to label tweets where tweets with :), :-), :D, :), and =) are mapped as positive and tweets with :(, : (and :-(symbols mapped as negative sentiments. The average number of words per tweet is 8. The original dataset size 1,600,000 and we used a subset of 16,000 tweets. The average length of the text in this dataset is 8 words per tweet.

The SemEval 2017 dataset: SemEval 2017 introduced in [56] and includes 62,671 tweets in original dataset. We were able to use only 20,547 tweets in our experiments due to memory limitation. The average sentence length for this dataset is 12 words. The sentiment label distribution is imbalanced (low number of negative tweets).

The Sanders Corpus: The Sanders Corpus [4] dataset consists of 5,074 manually annotated tweets with respect to four different categories: Apple, Google, Microsoft, Twitter. Each tweet was annotated as positive, negative, neutral, or irrelevant. The average length of the text in this dataset is 10 words per tweet.

UCI Sentence dataset: UCI Sentence dataset was introduced in [26]. This dataset is a combination of review datasets from Amazon, IMDB, and Yelp. The authors considered only strong positive and negative reviews for this dataset. It is the smallest dataset among all other datasets. It contains a total of 3000 sentences. This dataset contains cell phones and accessories category reviews from Amazon, Movie reviews from IMDB, and restaurant reviews from Yelp. Each dataset has 1000 samples and we considered 2700 samples for training and 300 samples for testing. The Scikit-learn's Train_Test_Split method is used for generating train and test sets. There are 12 words per review in this dataset.

AG News: The AG News dataset introduced by Zhang et al [82] and includes 1,20,000 training samples and 7,600 testing news articles. This dataset is obtained from the AG's corpus of news articles on the web[5]. It contains 496,835 categorized news articles from more than 2000 news sources. Only the 4 largest classes from this corpus were selected to construct this dataset. The title and description fields are included in this dataset. These two columns are used as features for classification. To generate vector embeddings, these columns are combined into a single column. It contains four categories of news such as ”World”, ”Sports”, ”Business” and ”Science”.

[3]https://www-cs.stanford.edu/people/alecmgo/papers/TwitterDistantSupervision09.pdf
[4]http://www.sananalytics.com/lab/twitter-sentiment/
[5]http://groups.di.unipi.it/g̃ulli/AG_corpus_of_news_articles.html

We used 3,000 samples from each category as our training set for our experiments. The average length of the dataset is 40 words per sentence. AG-News dataset does not require any further pre-processing because it contains news headlines and descriptions of news which is published after peer-reviewing by editors. It does not contain any grammar or spelling mistakes. Table 4.3 shows the data distribution for the AG-news dataset.

Table 4.2: Text classification Dataset Information

Dataset	**Type**	**Size**	**Positive**	**Negative**	**Neutral**	**Irrelevant**
Covid-Sentiment	Train	7000	1542	2124	3334	-
	Test	1003	236	374	393	-
U.S. Airline Sentiment	Train	12000	2015	7322	2663	-
	Test	1000	130	675	195	-
IMDB Movie Review	Train	20000	10055	9945	-	-
	Test	2000	1007	993	-	-
SST-2	Train	15000	8306	6694	-	-
	Test	1500	833	667	-	-
Sentiment140	Train	15000	7500	7500	-	-
	Test	1000	500	500	-	-
SemEval 2017	Train	17001	6915	2551	7534	-
	Test	3546	1473	559	1513	-
Sanders corpus	Train	4059	416	462	1837	1344
	Test	1015	100	107	484	324
UCI Sentence	Train	2700	1326	1374	-	-
	Test	300	174	126	-	-

Table 4.3: AG News Dataset Information

Dataset	**Type**	**Size**	**World**	**Sports**	**Business**	**Science**
AG-News	Train	12000	3000	3000	3000	3000
	Test	1150	300	250	300	300

20-Newsgroups: The 20 Newsgroups dataset is a common benchmark used for evaluating the performance of classification algorithms. The dataset, introduced in [29], contains approximately 20,000 newsgroup posts and this dataset is partitioned (nearly) evenly across 20 different newsgroups. Some of the newsgroups are very closely related to each other (e.g. comp.sys.ibm.pc.hardware and comp.sys.mac.hardware), while others are unrelated (for example, misc.forsale and soc.religion.christian). The 20 topics are organized into broader categories: computers, recreation, religion, science, sale, and politics. Scikit-learn module (in Python) was used to prepare the training and testing datasets to remove noisy data. This dataset has 10314 training samples and 782 testing samples. In 20 newsgroups dataset, each document is stored in a row as a pandas data frame. There are 209 words per document in this dataset.

Table 4.4: 20-Newsgroups dataset

News Category	**# Train documents**	**# Test documents**
alt.atheism	442	30
comp.graphics	534	38
comp.os.ms-windows.misc	528	48
comp.sys.ibm.pc.hardware	540	44
comp.sys.mac.hardware	527	38
comp.windows.x	561	31
misc.forsale	535	46
rec.autos	526	43
rec.motorcycles	548	39
rec.sport.baseball	554	27
rec.sport.hockey	551	39
sci.crypt	537	46
sci.electronics	547	30
sci.med	551	32
sci.space	525	55
soc.religion.christian	542	52
talk.politics.guns	499	38
talk.politics.mideast	511	41
talk.politics.misc	421	37
talk.religion.misc	335	28

Chapter 5

Experiments, Results and Discussion

In this chapter, we discuss operational ML algorithms used in this thesis, vector generation pipeline and their importance for text classification with extensive experiments and analysis of the results.

5.1 Operational Machine Learning Algorithms

There are five other machine learning algorithms considered for the analysis and comparison with TSC. These algorithms are supervised, and some are ensemble learning algorithms. TF-IDF and BERT Vectors were used for Support Vector Machine (SVM), Maximum Entropy (ME), Random Forest (RF), Stochastic Gradient Decent (SGD), and Light Gradient Boosting Machine (LGBM) for analysis.

SVM: This algorithm works on a simple strategy of separating hyperplanes. SVM categorizes the test data into an optimal hyperplane based on training data. The data points are plotted in an n-dimension vector space (n depends upon the features of the data points). SVM algorithm is used for binary classification and regression tasks but in our case, we have a multi-class text classification. We adopt the pairwise classification technique where each pair of classes will have one SVM classifier trained to separate the classes.

ME: Maximum Entropy is also known as Logistic Regression algorithm. This algorithm was named after the core function used in it that is the logistic function. The logistic function is also known as the sigmoid function. It is a S-shaped curve that takes real values as input and converts it into a range between 0 and 1. The sigmoid function is defined in equation 5.1.

$$S(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1} \tag{5.1}$$

RF: Random Forest classifier is an ensemble learning classification algorithm. It is very similar to decision tree but contains a multitude of decision trees and the class label is the mode value of the classes predicted by individual decision trees. This algorithm is efficient in handling large datasets and thousands of input variables. This model can deal with overfitting of data points. For a dataset, D, with N instances and A attributes, the general procedure to build a Random Forest ensemble classifier is as follows. For each time of building a candidate decision tree, a subset of the dataset D, d, is sampled with replacement as the training dataset. In each decision tree, for each node, a random subset of the attributes A, is selected as the candidate attributes to split the node. By building K decision trees in this way, a RF classifier is built which uses majority voting and returns the class label with maximum votes by the individual decision trees.

SGD: Stochastic Gradient Descent is a fundamental and efficient method for differentiating or differentiable learning in linear classifications under the convex loss functions such as the Logistic regression and SVM. It was introduced in [2] to reduce complexity of ML algorithms on large-scale data. It uses Hinge loss or no-differentiable loss with a support vector machine, and the model adopts the changes over time. Its learning rate is fixed in a simple implementation of the vanilla model. In addition to this, this model loss optimized through L_2 penalty regularization approach, which is specially used SVM ML techniques. SGD classifier supports first-order learning. However, this method iterates the training samples, and each cycle the model updates the parameters. Operation of learning parameter also controls the step size in space and similarly intercept is updated but without regularization. Learning rate can be either constant or gradually decaying.

LGBM: LGBM is Gradient Boosting machine learning algorithm developed by ke et al [25]. It is a Gradient Boosting Decision tree (GBDT) framework that uses leaf-wise tree growth for learning, which results in faster training with better accuracy of the result even by utilizing less memory. LightGBM combines two techniques called Gradient-based One Side Sampling (GOSS) which reduces the size of data, and Exclusive Feature Bundling (EFB) which reduces the number of features using histogram-based algorithms instead of finding the best split point to solve the problem of GBDT.

5.2 Vector Generation

The TSC algorithm was implemented using Python on a 16 GB RAM, Nvidia RTX 2060 GPU, 512 GB SSD machine. The process flow for the TSC algorithm is given in Figure 3.5 (see Chapter 3). SBERT base vectors (1x768 dimensional vectors) were used for the TSC algorithm. We considered mean and median values for determining the prototype class vectors for the TSC algorithm. In addition, we experimented using TF-IDF vectors with the TSC algorithm for all the datasets. However, the classification results with TF-IDF vectors were unsatisfactory primarily because the cosine similarity values started to converge to zero and hence resulted in a number of null values in the distance matrix as shown in Figure 3.6 in Chapter 3.

Since TF-IDF vectors are not suitable for the TSC algorithm, we used an additional method for generating vectors using two pre-trained deep learning models BERT and SBERT. To compare the performance of the proposed TSC algorithm, we used the following supervised algorithms: RF, ME, SVM, SGD and LGBM implementations from the Scikit-learn library [1]. The vectors generated by the BERT model were extracted using a pipeline parallelism from PyTorch and training the transformer model of Python. SBERT vectors are generated by sentence transformers[2] module of Python which contains only pre-trained models and does not provide flexibility to integrate with the Scikit-learn module. We used the ”BERT-base-NLI-Mean-Token” model generated by SBERT for the TSC algorithm and ”BERT-based-uncased” model vectors for the classical ML algorithms.

5.3 Sample Vectors

The vectors play an essential role in text classification. Table 5.1 shows the vectors generated by TF-IDF, SBERT, and BERT for one tweet from the Covid-Sentiment dataset: ”state_sca us india collaborated develop rotavirus vaccine expected save 80000 children india alone”. This tweet is already pre-processed before generating vectors from it.

[1]https://scikit-learn.org/stable/
[2]https://www.sbert.net/

Table 5.1: Example of Vectors in Covid-Sentiment dataset

Type of embedding	**Dimension of vector**	**Vector**
TF-IDF	2743	[0.22987371, 0.0, 0.0,...., 0.0, 0.0]
SBERT	768	[-5.73375404e-01, 4.02642965e-01, ... , 6.87391400e-01, 2.53910303e-01]
BERT	768	[4.24835116e-01, 1.49365336e-01,.... ,-5.49187362e-01, -3.73176634e-01]

The size of the TF-IDF vector is 2743 for the Covid-Sentiment dataset. This dataset has only 8003 tweets and an average of 12 words per tweet. The length of TF-IDF vectors is different for every dataset. Longer sentences result in longer TF-IDF vectors, whereas BERT and SBERT vectors have a 768-dimensional vector which remains constant for all datasets. The dimension of the TF-IDF vector depends on unique words in a dataset.

5.4 Performance Measures

In our experiments, some datasets have an imbalanced class distribution and to evaluate them properly weighted F1-score is preferred because it considers each class, and gives a weight based on the support[3] for that class. The weighted F1-score is derived from weighted Precision and weighted Recall. The calculation of Precision and Recall for each class c is given in Equations (1) and (2) respectively. The Weighted_Precision and Weighted_Recall given in Equations(3) and (4) is an average measure over all samples by considering the support or true number of examples for each class. This support is its weight denoted by $Weight_{ci}$ where ci is the ith class and n is the number of classes. Weighted F1-Score combines Weighted Precision and Weighted Recall as shown in equation (5).

[3]the number of true instances for each class

$$\text{Precision}_\text{c} = \frac{\text{True\_Positive}}{\text{True\_Positive} + \text{False\_Positive}} \tag{1}$$

$$\text{Recall}_\text{c} = \frac{\text{True\_Positive}}{\text{True\_Positive} + \text{False\_Negative}} \tag{2}$$

$$\text{Weighted\_Precision} = \frac{\sum_{i=1}^{n} Precision_{ci} * Weight_{ci}}{\sum_{i=1}^{n} Weight_{ci}} \tag{3}$$

$$\text{Weighted\_Recall} = \frac{\sum_{i=1}^{n} Recall_{ci} * Weight_{ci}}{\sum_{i=1}^{n} Weight_{ci}} \tag{4}$$

$$\text{Weighted\_F1} - \text{Score} = \frac{2 * Weighted\_Precision * Weighted\_Recall}{Weighted\_Precision + Weighted\_Recall} \tag{5}$$

5.5 Analysis of Results

We analyzed performance of the TSC algorithm with ten datasets with different dataset sizes, unequal distribution of sentiment classes as well as more than one class.

5.5.1 Experiments with the TSC algorithm

In this section, we discuss the performance of the TSC algorithm on all datasets.

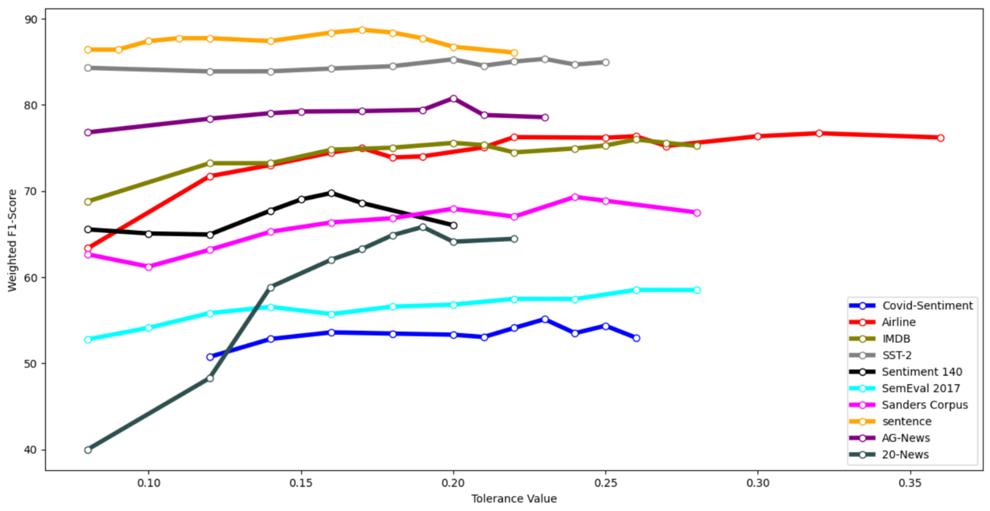


Figure 5.1: TSC-Mean for all datasets

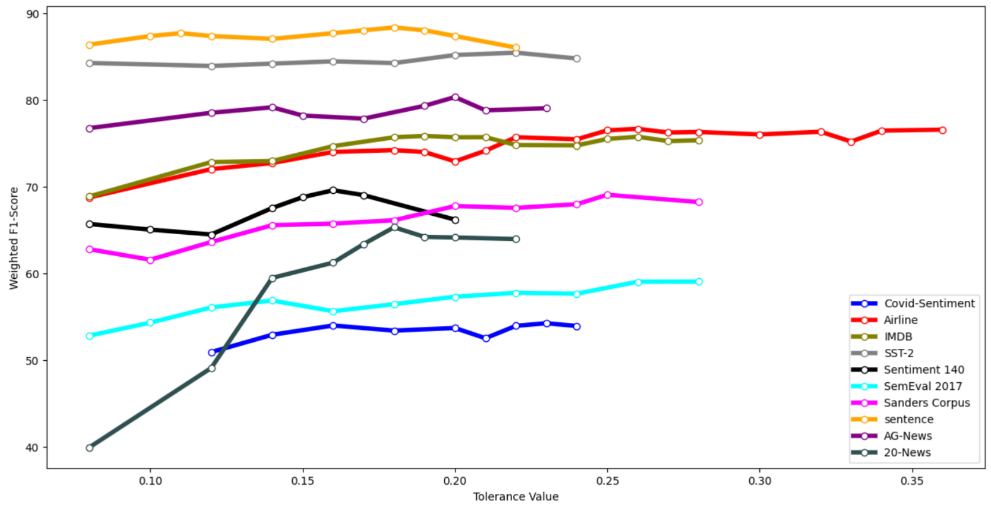


Figure 5.2: TSC-Median Approach for all datasets

Figure 5.1 represents the weighted F1-Score for all datasets for various tolerance values using the mean value for the prototype class vector and Figure 5.2 represents the median value for the prototype class vector. The results with the mean value are slightly better in terms of the weighted F1-score for all the datasets. Hence, we use the mean value for the TSC algorithm in all our subsequent experiments.

The range of tolerance values is from 0.08 to 0.38. The UCI sentence dataset performs best and Covid-Sentiment has the lowest score among all other datasets. The 20-Newsgroups dataset shows a dramatic increment after a tolerance value of 0.12.

Table 5.2 shows the number of *tolerance* classes for the best tolerance value for each dataset. The TSC algorithm generates these classes as described in Algorithm 1 (described in Chapter 3). The number of class representatives for each class plays an important role during the testing phase of the TSC algorithm. It can be observed that balanced tolerance classes (in terms of size) is an indication of good semantic similarity between vectors generated by the transformer model.

Based on the results in Table 5.2, SST-2 and UCI sentence datasets have approximately similar number of tolerance classes. It should also be noted that the experiments with these two datasets yield a weighted F1-score of over 85% using ε of 0.23 and 0.17 respectively shown in Figures 5.1 and 5.2. In addition, all algorithms give the best result with these two datasets. Another observation is that these two datasets contain binary classes. Other datasets in our experiments include three and four sentiment classes as shown in Table 5.2. However, these datasets do not generate balanced tolerance classes. Table 5.2 shows AG-News dataset with a well-balanced tolerance class distribution. All algorithms perform (third best) on this dataset (see Fig. 5.1 and 5.2).

Table 5.2: Best ε value for the TSC algorithm and Tolerance Class sizes

Dataset	**ε Value**	**TC-Positive**	**TC-Negative**	**TC-Neutral**	**TC-Irrelevant**
Covid-Sentiment	0.23	249	1136	2590	-
U.S. Airline	0.32	2009	8202	1234	-
IMDB	0.26	6979	12956	-	-
SST-2	0.23	6501	5531	-	-
Sentiment140	0.16	926	1969	-	-
SemEval 2017	0.26	6583	1755	5797	-
Sanders corpus	0.24	205	390	1415	1518
UCI Sentence	0.17	576	557	-	-

Table 5.3: Best ε value for the TSC algorithm and Tolerance Class size for AG News dataset

Dataset	**ε Value**	**TC-World**	**TC-Sports**	**TC-Business**	**TC-Science**
AG-News	0.2	1900	2607	2563	2398

The 20-Newsgroups dataset has the highest number of news classes among all other datasets. It has twenty sentiment classes and a fairly balanced tolerance class distribution as shown in Table 5.4. Due to better semantic similarity of its vectors, the performance on this dataset is better than Covid-sentiment and SemEval 2017 datasets.

5.5.2 Experiments with the Covid-Sentiment Dataset

Figure 5.3 shows the results of the Covid-Sentiment dataset. We considered seven algorithms for analysis. The TF-IDF vectors and transformer vectors are considered. Both mean and median value for the prototype class vector are considered for the TSC algorithm.

Table 5.4: 20-Newsgroups dataset for best ε value of 0.19

News Category	#Train documents	# Tolerance Classes
alt.atheism	442	230
comp.graphics	534	238
comp.os.ms-windows.misc	528	498
comp.sys.ibm.pc.hardware	540	445
comp.sys.mac.hardware	527	428
comp.windows.x	561	400
misc.forsale	535	312
rec.autos	526	342
rec.motorcycles	548	189
rec.sport.baseball	554	319
rec.sport.hockey	551	332
sci.crypt	537	437
sci.electronics	547	146
sci.med	551	285
sci.space	525	272
soc.religion.christian	542	494
talk.politics.guns	499	380
talk.politics.mideast	511	335
talk.politics.misc	421	248
talk.religion.misc	335	130

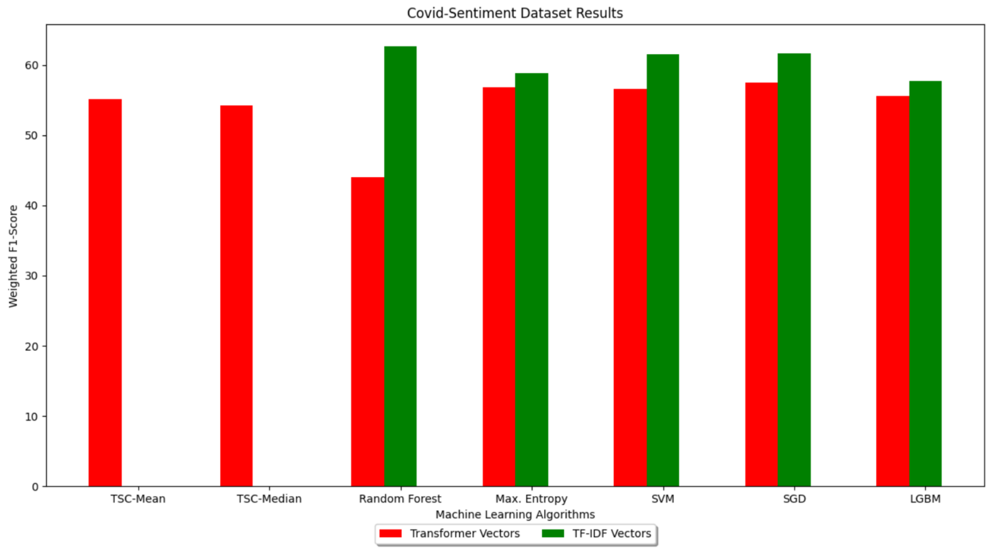


Figure 5.3: Results of Covid-Sentiment dataset

As per Figure 5.3, TF-IDF-based RF performs best among all other algorithms which is 62.62%. Overall, the TF-IDF vectors give the best results compared to transformer vectors. As outlined in Chapter 5, Covid-Sentiment has diverse range of opinions on different topics. Hence the transformer vectors are not able to establish proper semantic-similarity. TF-IDF vectors only consider the frequency of a particular word from a sentence rather than the meaning of the sentence Hence, all algorithms perform better with TF-IDF vectors than with transformer vectors. The TSC algorithm gives very comparable results with transformer vectors-based ML algorithms. The TSC-Mean performs slightly better than TSC-Median. TSC-Mean and TSC-Median achieves 55.13% and 54.28% Weighted F1-score respectively. There is a notable difference in the performance of the RF algorithm using TF-IDF vectors (highest F1-score) and transformer vectors (lowest F1-score). Interestingly, the RF algorithm also gives the *best* F1-score which shows the importance of quality of vectors in text classification.

5.5.3 Experiments with the U.S. Airline sentiment Dataset

Figure 5.4 shows the performance of seven algorithms. The highest weighted F1-score is achieved by TF-IDF-based LGBM which is 79.97%. The lowest F1-score was achieved by transformer vectors-based RF which is 71.63%. The TSC-mean and TSC-median performed equally and their weighted F1-score is 76.71%.

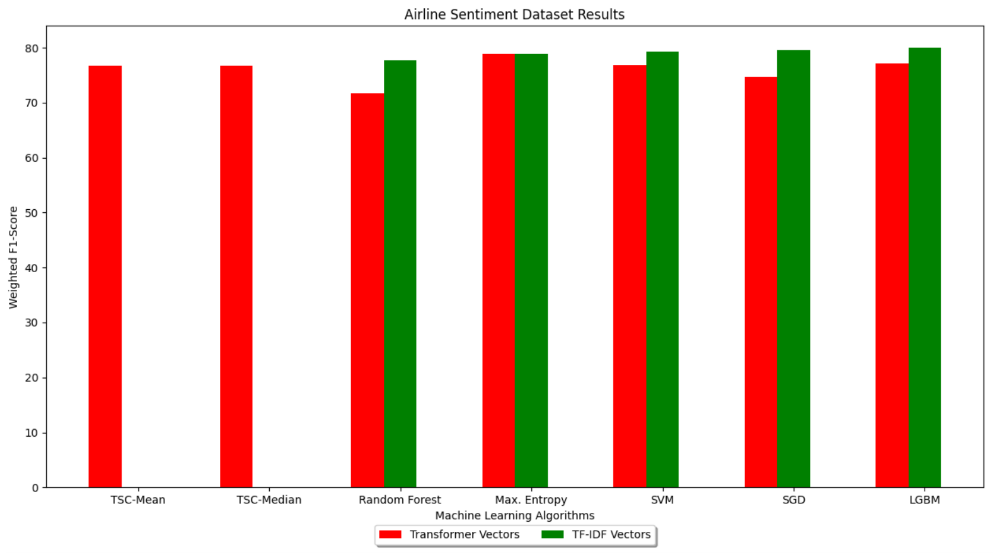


Figure 5.4: Results of U.S. Airline sentiment dataset

The ME-based algorithm shows similar results with both vectors. The results show that TF-IDF-based algorithms perform better than transformer vector-based algorithms. The Airline dataset has more negative tweets compared to positive and neutral tweets and it causes the problem of imbalance tolerance classes as shown in Table 5.2. However, it appears that the tweets have good semantic similarity with a class. The U.S. Airline dataset is focused on customer reviews only from U.S.A. and due to same geographic region there is more similarity in tweets.

5.5.4 Experiments with the IMDB Movie Review Dataset

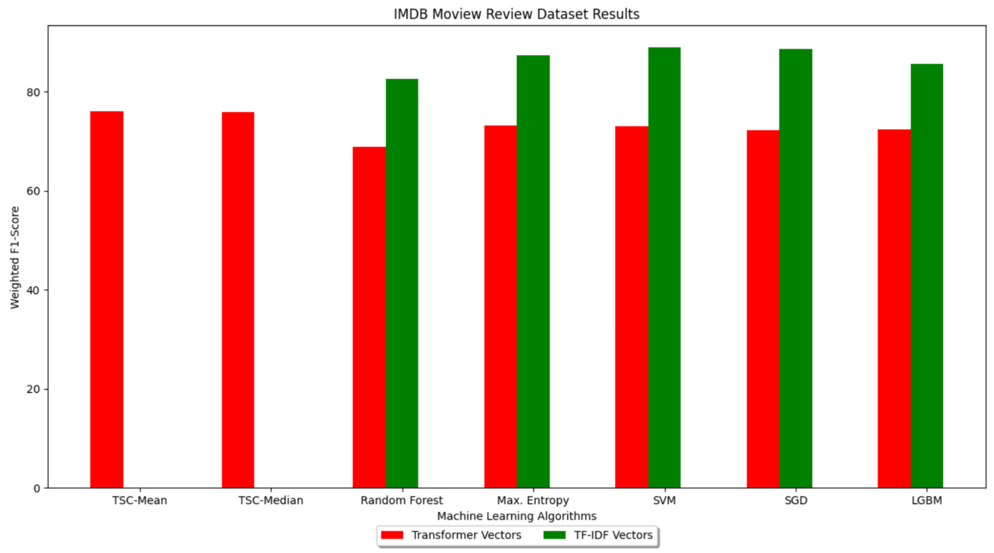


Figure 5.5: Results of IMDB Movie Review dataset

The IMDB Movie review dataset contains movie reviews that are either positive or negative. Each review is a document and has an average length of review is 230 words per review. Figure 5.5 shows that TF-IDF-based ML algorithms perform better than transformer vectors-based algorithms for this dataset. The IMDB dataset has a balanced class distribution-based on the Table 4.2 However, as can be seen in Table 5.2, the tolerance classes are highly imbalanced. Again, the transformers vectors do not induce a good performance. The TF-IDF vectors perform best with the large sentences because they can build more vocabulary for the analysis meanwhile it is opposite for the transformer vectors because a similar word creates a new vector representation. Transformer vectors have a dimension of only 1x768 which puts a limitation to represent 230 words in a single vector meanwhile for other datasets which have an average length of 10 to 16 words represent the same dimensional vector. Due to that reason transformer vectors-based algorithms have a lower weighted F1-score compared to TF-IDF-based ML algorithms.

The comparison of only transformer vectors-based algorithms shows that TSC-mean and TSC-median outperformed all other transformer vectors-based other ML algorithms. It shows the performance of TSC algorithm is better than other ML algorithms in terms of similar architecture-based vectors. The TSC-mean and TSC-median achieve 75.99% and 75.89% weighted F1-score respectively. The TF-IDF-based SVM achieves the highest weighted F1-score which is 88.90% and the lowest weighted F1-score achieved by the transformer vectors-based RF algorithm. The results demonstrate that transformer vectors-based supervised algorithms cannot perform well for document-level longer text classification compared to TF-IDF-based ML algorithms.

5.5.5 Experiments with the SST-2 Dataset

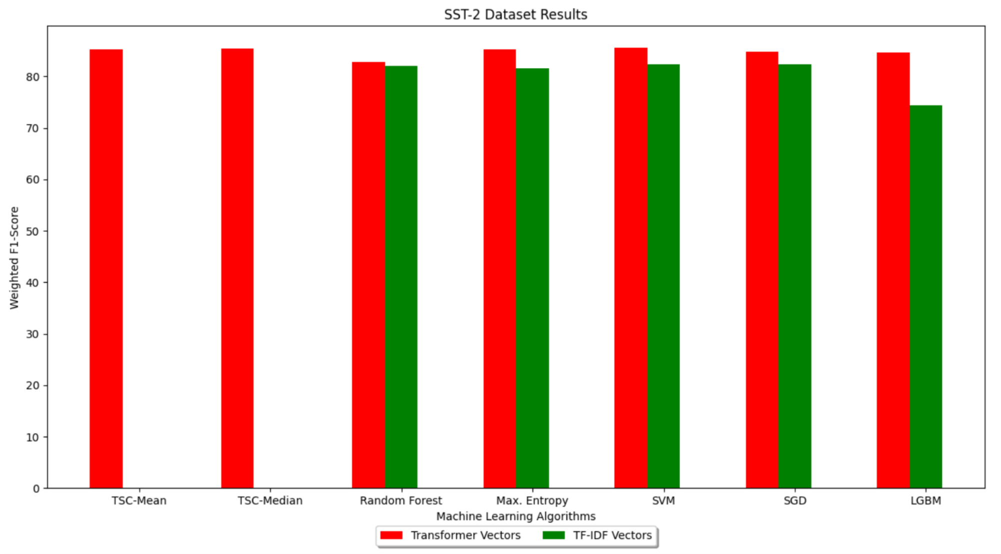


Figure 5.6: Results of SST-2 dataset

The SST-2 dataset contains phrases of the Standford Sentiment treebank. Figure 5.6 shows the results of algorithms on this dataset. The results show that transformer vectors-based algorithms outperformed TF-IDF-based ML algorithms. The SST-2 dataset is a very clean dataset and transformer vectors are able to establish good semantic similarity with this dataset due to that reason TSC-mean and TSC-median both outperformed all TF-IDF-based ML algorithms. The TSC-mean and TSC-median algorithms give 85.34% and 85.49% weighted F1-score respectively. The highest weighted F1-score was achieved by transformer vectors-based SVM which is 85.55% and the lowest score was achieved by TF-IDF-based LGBM which is 74.37%.

As per the Table 5.2, the SST-2 dataset generates balanced tolerance classes due to good semantic similarity and it performed better than TF-IDF-based algorithms. There is a difference of only 1% in weighted F1-score between TSC algorithm and transformer vectors-based SVM. It shows the TSC algorithm performs well and gives comparable and better results with other algorithms.

5.5.6 Experiments with the Sentiment 140 Dataset

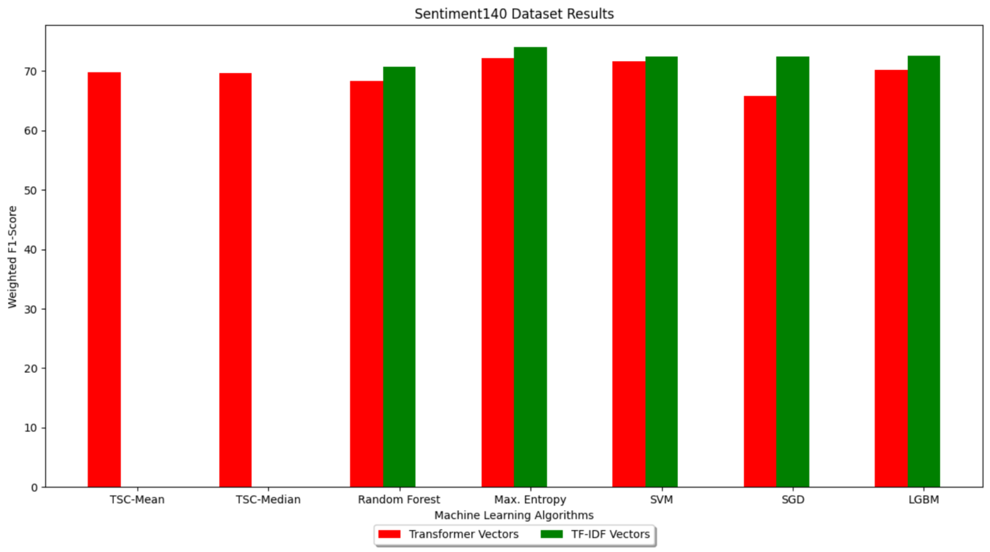


Figure 5.7: Results of Sentiment 140 dataset

The sentiment 140 dataset is derived from the tweets have emoticons. The dataset has two classes. Based on Table 4.2 there are an equal number of positive and negative tweets for this dataset. However, the Table 5.2 shows an imbalance positive and negative tolerance classes for this dataset due to less semantic similarity. The tweets in this dataset were collected from all around the world. The authors only considered emoticons to label the tweets which can be misleading since some tweets could be interpreted as having sarcastic meaning.

The Figure 5.7 shows the results of TF-IDF-based and transformer vectors-based algorithms. The lowest weighted F1-score is achieved by transformer vectors-based SGD which is 65.80%. The highest weighted F1-score achieved by TF-IDF-based ME which is 73.99%. The weighted F1-score for the TSC-mean and TSC-median is

69.78% and 69.62% respectively. which is almost 70% after rounding off. The results of TF-IDF-based ML algorithms are still comparable with transformer vectors-based algorithms.

5.5.7 Experiments with the SemEval2017 Dataset

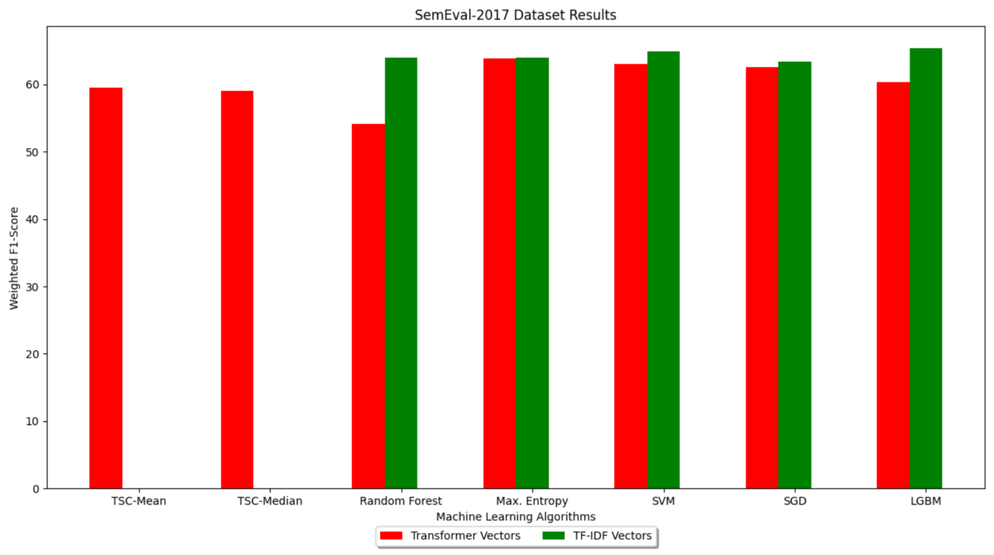


Figure 5.8: Results of SemEval2017 dataset

The SemEval2017 has three classes positive, negative and neutral. Table 4.2 shows the initial class distribution for this dataset. There are very few samples for the negative class in this dataset and due to that reason the tolerance class distribution is imbalanced as well as shown in Table 5.2. There are fewer negative classes when compared to other categories. SemEval2017 contains tweets around the globe as a result, there is less semantic similarity between the tweets.

The Figure 5.8 shows the performance of algorithms on this dataset. The TSC-Mean achieves 59.56% weighted F1-score and TSC-Median achieves 59.09% weighted F1-score. The TF-IDF-based LGBM performed better than all others. It achieved 65.33% weighted F1-score. The transformer vectors-based RF performed less compared to all other algorithms. It achieved 54.17% weighted F1-score. The ME gives the same results for both vectorization methods.

5.5.8 Experiments with the Sanders corpus

The Sanders corpus has four classes and it is a relatively small dataset. It has positive, negative, neutral, and irrelevant hand-labeled tweets. This dataset is focused on tweets related to technology based companies like Google, Microsoft. The Sanders corpus has more neutral and irrelevant tweets compared to the positive and negative tweets. The class boundaries between neutral irrelevant is unsharp. This makes the classification task more interesting between these two classes. Figure 5.9 shows that the three transformer vectors-based ML algorithms perform better than TF-IDF-based ML algorithms. The RF and SVM perform better with TF-IDF vectors compared to the transformer vectors.

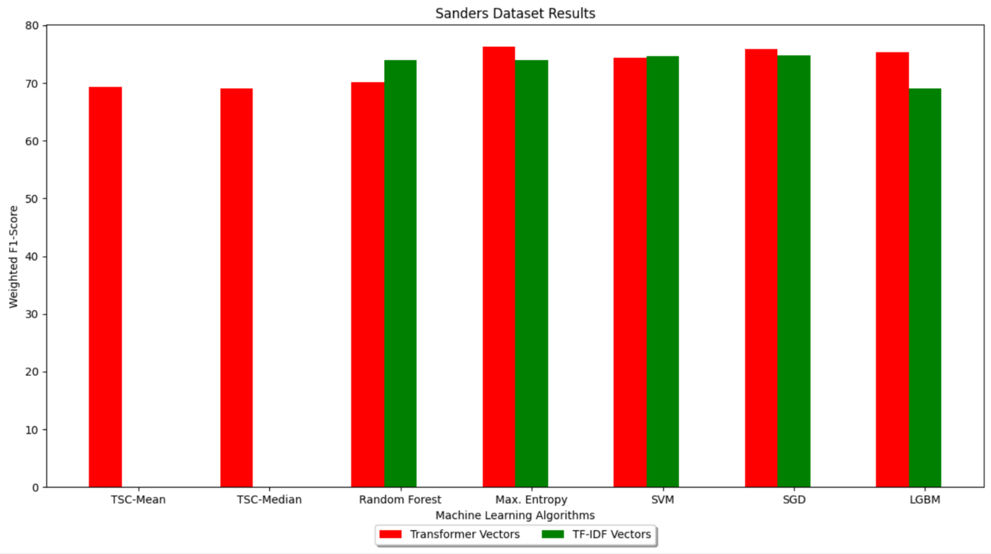


Figure 5.9: Results of Sanders corpus

The TSC-mean and TSC-median achieve 69.32% and 69.10% weighted F1-score respectively. The transformer vectors-based ME outperform all other algorithms. It achieved 76.27% weighted F1-score. The lowest weighted F1-score was achieved by TF-IDF-based LGBM. It achieved 69% weighted F1-score. The Sanders corpus shows the advantage of transformer vectors-based ML algorithms compared to TF-IDF-based ML algorithms.

5.5.9 Experiments with the UCI Sentence Dataset

The UCI Sentence dataset consists of strong positive and negative sentences and it has an average of 12 words per sentence. This dataset is very small and it has a balanced class distribution as shown in the Table 4.2. As per Table 5.2 the tolerance classes are also balanced for this dataset which shows a very high semantic similarity between transformer vectors. Figure 5.10 represents the performance of algorithms for this dataset.

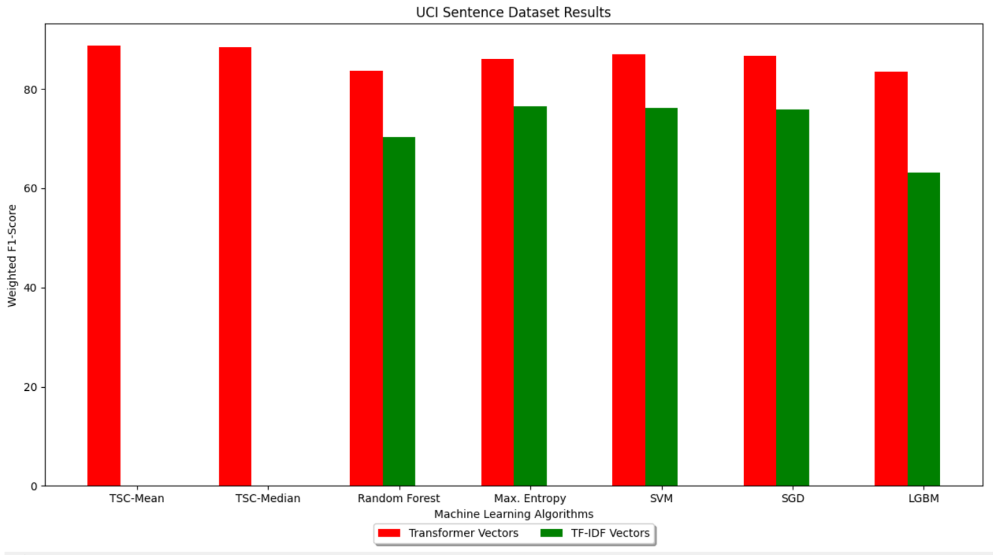


Figure 5.10: Results of UCI Sentence dataset

The transformer vectors-based algorithms outperform all TF-IDF-based ML algorithms. The TSC-mean outperforms all other algorithms in this dataset. It achieves 88.73% weighted-F1-score and TSC-median achieves 88.40% weighted F1-score. The transformer vectors-based SVM performs better than other ML algorithms. It achieves 87.05% weighted F1-score. The TF-IDF-based LGBM achieves the lowest weighted F1-score which is 63.21%. The UCI Sentence dataset is the smallest dataset among all datasets as a result TF-IDF vectors are not able to produce sufficient sized vocabulary and algorithms-based on these vectors are not able achieve good results. The authors of this dataset considered only strong sentiment reviews which play important role in establishing semantic similarity for the transformer vectors. This dataset shows the importance of the size of the dataset and semantic similarity for the TSC algorithm.

5.5.10 Experiments with the AG-News Dataset

The AG-News dataset is a news classification task with four different news categories. The news dataset has correct grammar and spelling which is an advantage of news classification over tweet classification. The correct grammar and spelling are very helpful to establish semantic similarity between the news data. Table 4.3 shows that all four categories are balanced at the dataset level. Table 5.3 shows a similar number of tolerance classes for all four news categories which show the semantic similarity of the TSC-mean algorithm. The Figure 5.11 shows the performance of all algorithms for this dataset.

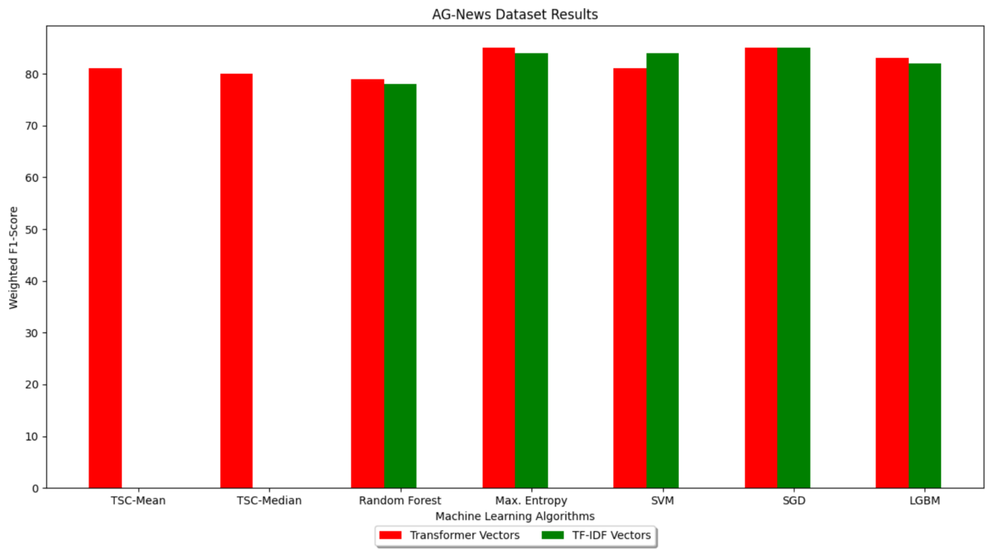


Figure 5.11: Results of AG-News dataset

The transformer vectors-based algorithms perform better than TF-IDF-based ML algorithms except SVM and SGD. The TF-IDF-based SGD outperform all other algorithms. The TF-IDF-based SVM achieves 84% weighted F1-score meanwhile transformer vectors-based SVM gets 81% weighted F1-score. The SGD gets 84.53% and 84.87% of weighted F1-score with TF-IDF and transformer vectors respectively. The TF-IDF-based RF gets the lowest weighted F1-score which is 78.19%. The TSC-mean and TSC-median achieve 80.77% and 80.37% weighted F1-score respectively. The AG-News dataset has 40 words per the news which is higher than SST-2 and UCI-sentence dataset. The TF-IDF vectors have the advantage of larger size and better vocabulary. However, the transformer vectors produce better semantic similarity and are able to give very comparative results with TF-IDF-based algorithms. There are more number of transformer vectors-based ML algorithms which performs better than their TF-IDF vector versions for example RF, ME and LGBM.

5.5.11 Experiments with the 20-Newsgroups Dataset

The 20-Newsgroups classification is document-level classification. Each row of this data represents one document. It has 209 words per document and have twenty news categories makes it a very interesting classification problem. It has the highest class among all other datasets. Finding balanced data distribution and tolerance classes is challenging in this dataset. Figure 5.12 shows the performance of all algorithms on this dataset. Table 5.4 shows the tolerance class distribution for this dataset

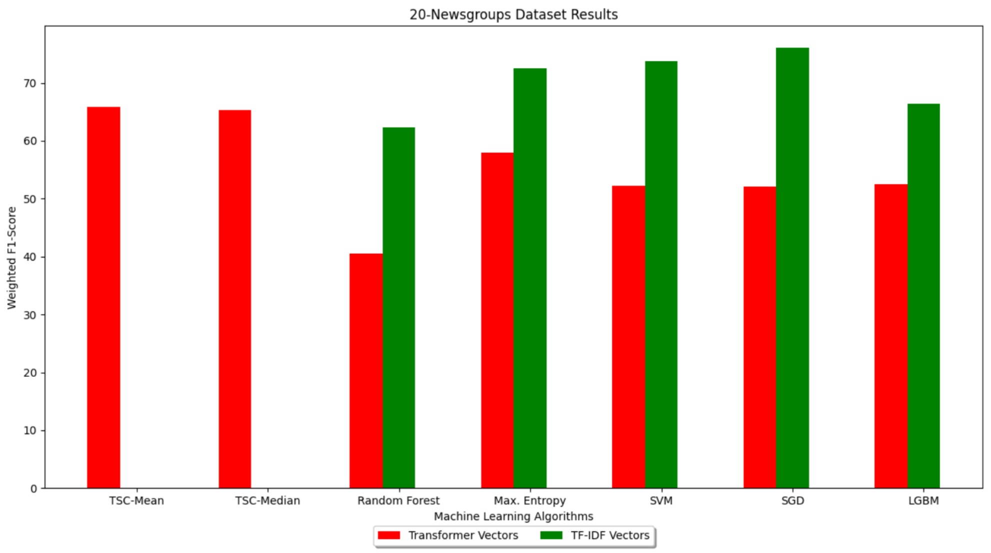


Figure 5.12: Results of 20-Newsgroups dataset

The TF-IDF-based ML algorithms perform better than transformer vectors-based algorithms due to very long length of the document size and each document represented by 1x768 dimensional transformer vectors meanwhile TF-IDF-based algorithms has the advantage of a rich vocabulary which makes it a better choice over the transformer vectors. The result shows that TSC-mean and TSC-median outperformed all other transformer vectors-based ML algorithms and achieved 66% and 65% weighted F1-score respectively. The TF-IDF-based SGD performs best compared to all other algorithms. It achieves 76% weighted F1-score. The transformer vector-based RF gets the lowest weighted F1-score which is 41%.

5.5.12 Overall Analysis

The TSC algorithm shows comparative and better performance in some datasets. Figure 5.13 shows the comparison of TSC algorithm with TF-IDF-based ML algorithms for all datasets. The TSC algorithm outperforms all ML algorithms in SST-2 and UCI Sentence datasets. Table 5.5 includes numerical data of weighted F1-score for all datasets with TSC algorithm and TF-IDF-based ML algorithms. The TSC algorithm uses SBERT vectors which give the best performance since they are able to establish good semantic similarity. The results show that TF-IDF-based ML algorithms give the best results in longer sentences or document-level classification tasks like IMDB and 20-newsgroups datasets. Due to long sentences, the SBERT vectors are not performing better due to having a fixed size of 1x768 dimensional vector for each document or sentence. The size of TF-IDF vectors depends on the vocabulary of that dataset which means having longer sentences makes intense vocabulary for the TF-IDF approach which is an advantage over transformer vectors.

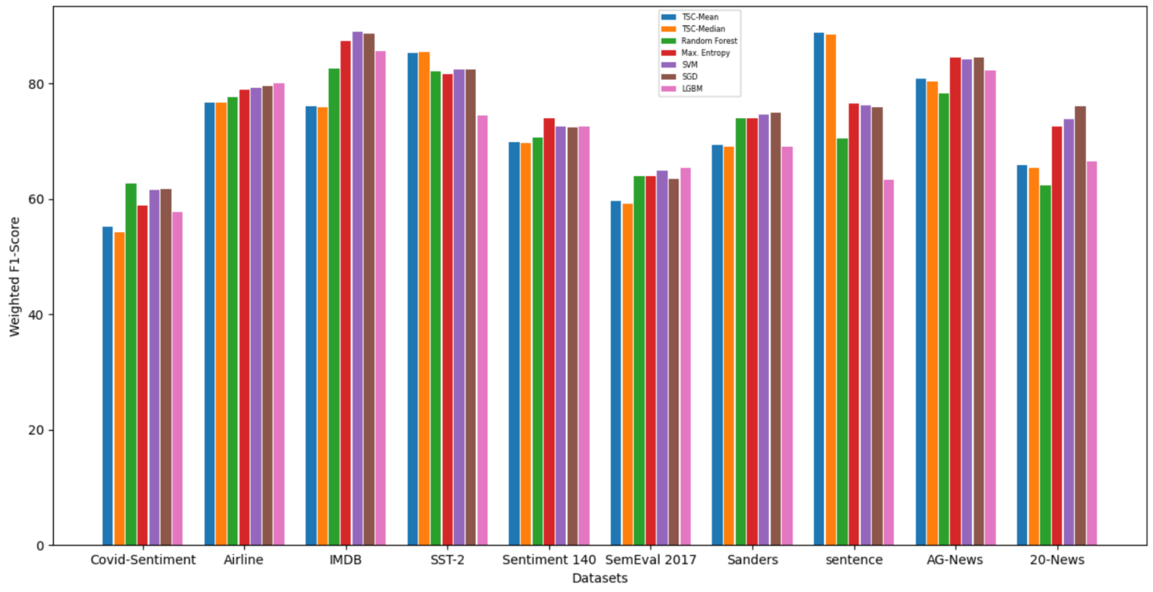


Figure 5.13: Overall comparison with TF-IDF-based approach with TSC algorithm

The TSC-mean performs better than TSC-median in some datasets hence its preference for analysis. Figure 5.14 shows the comparison between TSC algorithm and other ML algorithms for all datasets. Table 5.6 represents numerical values of weighted F1-score for every dataset and algorithms. The TSC algorithm performs outperforms all other algorithms in IMDB, UCI Sentence, and 20-Newsgroups datasets. The ML algorithms have transformer vectors but from different models. The TSC algorithm has SBERT based vectors meanwhile other ML algorithms have BERT vectors. The

Table 5.5: TF-IDF-based weighted F1-score Results

Dataset	**TSC-mean**	**TSC-median**	**RF**	**ME**	**SVM**	**SGD**	**LGBM**
Covid-Sentiment	55.13	54.28	**62.62**	58.88	61.47	61.65	57.76
U.S. Airline	76.71	76.71	77.70	78.90	79.23	79.53	**79.97**
IMDB	75.99	75.89	82.59	87.39	**88.90**	88.59	85.59
SST-2	85.34	**85.49**	82.12	81.65	82.36	82.36	74.37
Sentiment140	69.78	69.62	70.69	**73.99**	72.49	72.44	72.60
SemEval 2017	59.56	59.09	63.95	63.97	64.83	63.42	**65.33**
Sanders corpus	69.32	69.10	73.96	74.01	74.66	**74.86**	69
UCI Sentence	**88.73**	88.40	70.40	76.48	76.15	75.82	63.21
AG-News	80.77	80.37	78.19	84.44	84.19	**84.53**	82.29
20-Newsgroups	65.84	65.35	62.34	72.59	73.77	**76.04**	66.43

BERT and SBERT both are not able to give the best results in longer sentences but SBERT achieves slightly better semantic similarity compared to the BERT model due to that reason the TSC algorithm outperforms other ML algorithms in three datasets.

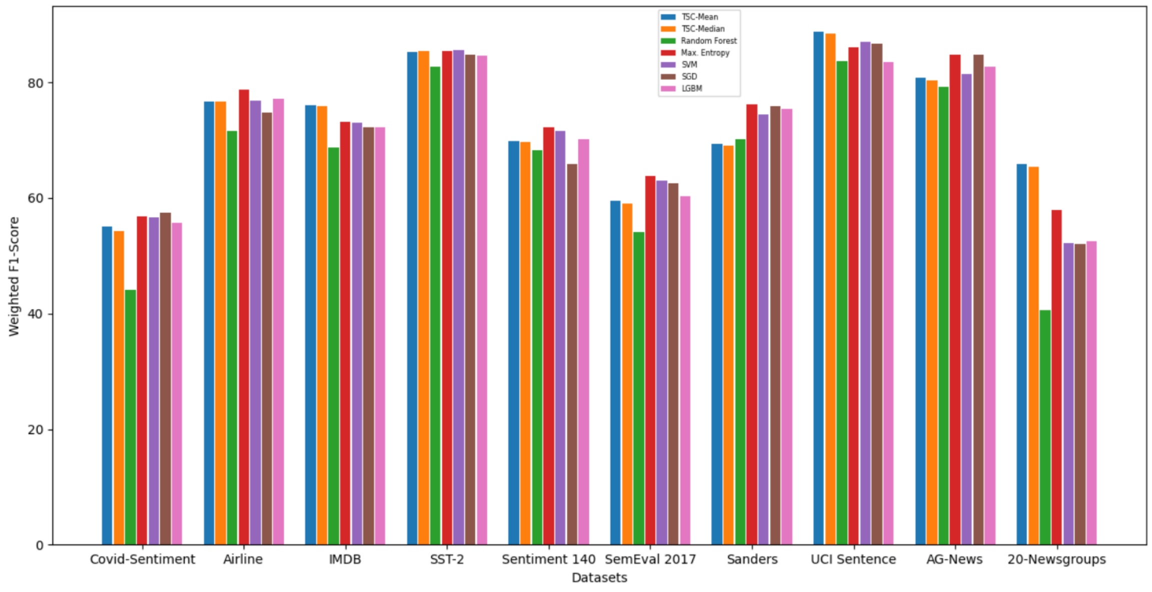


Figure 5.14: Overall comparison with Transformer vectors-based approach with TSC algorithm

As per Table 5.6, the TSC algorithm performs second best in SST-2 dataset. In SST-2, Transformer vectors-based SVM achieves the highest weighted F1-score which is 85.55% and TSC-median gets 85.49% which is an ignorable difference. After

Table 5.6: Transformer vectors-based weighted F1-score Results

Dataset	**TSC-mean**	**TSC-median**	**RF**	**ME**	**SVM**	**SGD**	**LGBM**
Covid-Sentiment	55.13	54.28	44.02	56.83	56.60	**57.52**	55.62
U.S. Airline	76.71	76.71	71.63	**78.79**	76.86	74.74	77.10
IMDB	**75.99**	75.89	68.79	73.19	72.99	72.24	72.29
SST-2	85.34	85.49	82.79	85.35	**85.55**	84.86	84.69
Sentiment140	69.78	69.62	68.30	**72.19**	71.59	65.80	70.19
SemEval 2017	59.56	59.09	54.17	**63.82**	62.99	62.55	60.28
Sanders corpus	69.32	69.10	70.19	**76.27**	74.42	75.90	75.38
UCI Sentence	**88.73**	88.40	83.73	86.06	87.05	86.73	83.44
AG-News	80.77	80.37	79.30	84.75	81.44	**84.87**	82.73
20-Newsgroups	**65.84**	65.35	40.52	57.89	52.27	52.05	52.54

rounding off the results of the Table 5.6, TSC algorithm performs second best in U.S. Airline, SST-2, and Sentiment 140 datasets.These results demonstrate the efficacy of the TSC algorithm on the chosen datasets

Chapter 6

Conclusion and Future Work

In this thesis, we implemented a modified form of a tolerance-based algorithm (TSC) to classify the sentiment polarities as well as news categories from text. TSC algorithm is a supervised algorithm that was designed to perform text classification with tolerance near set (TNS). It is the first time where TNS is used for text classification. The TSC algorithm is developed with the help of the Python programming language which provides very powerful scientific computing packages like numpy, pandas, and scipy. We applied two strategies TSC-mean and TSC-median in our extensive experiments. The performance of TSC algorithm was tested with various datasets related to natural language with some datasets having movie and restaurant reviews from customers, others having tweets and news headlines. The choice of the dataset was motivated by the fact that this data has undergone extensive pre-processing and is well-researched in terms of several published studies. We also hand-crafted a dataset from Twitter regarding opinions on Covid-19 across the globe. TSC algorithm achieves the best performance even in 20 newsgroups dataset which has twenty categories for classification. In this work, we have compared the TSC algorithm with the best-supervised algorithms that include ensemble learning, gradient decent, and gradient boosting strategies. The results show the TSC algorithm gives better and comparative results with such highly optimized algorithms.

We have compared the results of TSC algorithm with ten different datasets and five other supervised algorithms. TSC algorithm outperforms reported ML algorithms with two datasets using TF-IDF vectors and three datasets using transformer vectors. TSC algorithm outperforms other ML algorithms in the UCI Sentence dataset. In terms of execution time, the TSC algorithm takes a maximum of 780 minutes to compute the distance matrix in the IMDB dataset which has the highest number of training examples among all other datasets and it takes only 24 minutes to compute the distance matrix for the UCI Sentence dataset. The advantage of TSC algorithm is that we have to create a distance matrix only once for each dataset. This matrix can then be reused by storing it in the local machine with numpy file format. The TSC algorithm takes a maximum of 25 minutes to obtain results with the IMDB dataset with higher tolerance values due to our optimized approach, and 3 minutes for each tolerance value in the UCI Sentence dataset. The execution time of TSC algorithm is highly dependent on the size of the dataset.

As future work, optimization of TSC algorithm in terms of computing tolerance classes is a natural extension of the current work. It is also important to apply the TSC algorithm to other larger data sets mentioned in the related works section of this thesis. One of the major issues in dealing with these datasets is the cleaning and removing unnecessary text. The dataset with clean text gives better results compared to the original dataset, especially in tweets. The TSC algorithm takes much more time in making distance matrix which can be reduced by applying parallel computing with CUDA[1]. The TSC can also tested with Jaccard similarity to get comparative analysis with cosine distance. The Self-Supervised Learning [28] is new emerging area of research where we can explore it's possibility for text classification.

[1]https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html

Appendix A

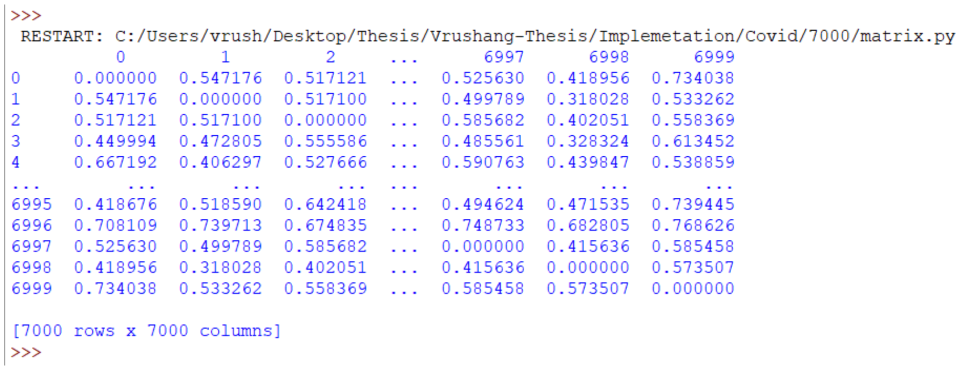
Selected Code Snapshots

A.1 Appendix

A sample distance matrix and the steps of the TSC algorithm are given in this Appendix. In addition, python code for generating vector embedding and ML pipeline is given.

A.1.1 Execution Flow

Based on the high level TSC algorithm Flow shown in Figure 3.5, The first step is to load the dataset. We used pandas to load CSV file data into python environment. We generated vectors from the SBERT algorithm and used those vectors to create distance matrix for training set. Figure A.1 shows the distance matrix for the Covid-sentiment dataset. We store this matrix in numpt file format (.npz) to reuse it for experimenting with different tolerance values.



```
>>>
 RESTART: C:/Users/vrush/Desktop/Thesis/Vrushang-Thesis/Implemetation/Covid/7000/matrix.py
             0         1         2  ...      6997      6998      6999
0     0.000000  0.547176  0.517121  ...  0.525630  0.418956  0.734038
1     0.547176  0.000000  0.517100  ...  0.499789  0.318028  0.533262
2     0.517121  0.517100  0.000000  ...  0.585682  0.402051  0.558369
3     0.449994  0.472805  0.555586  ...  0.485561  0.328324  0.613452
4     0.667192  0.406297  0.527666  ...  0.590763  0.439847  0.538859
...        ...       ...       ...  ...       ...       ...       ...
6995  0.418676  0.518590  0.642418  ...  0.494624  0.471535  0.739445
6996  0.708109  0.739713  0.674835  ...  0.748733  0.682805  0.768626
6997  0.525630  0.499789  0.585682  ...  0.000000  0.415636  0.585458
6998  0.418956  0.318028  0.402051  ...  0.415636  0.000000  0.573507
6999  0.734038  0.533262  0.558369  ...  0.585458  0.573507  0.000000

[7000 rows x 7000 columns]
>>>
```

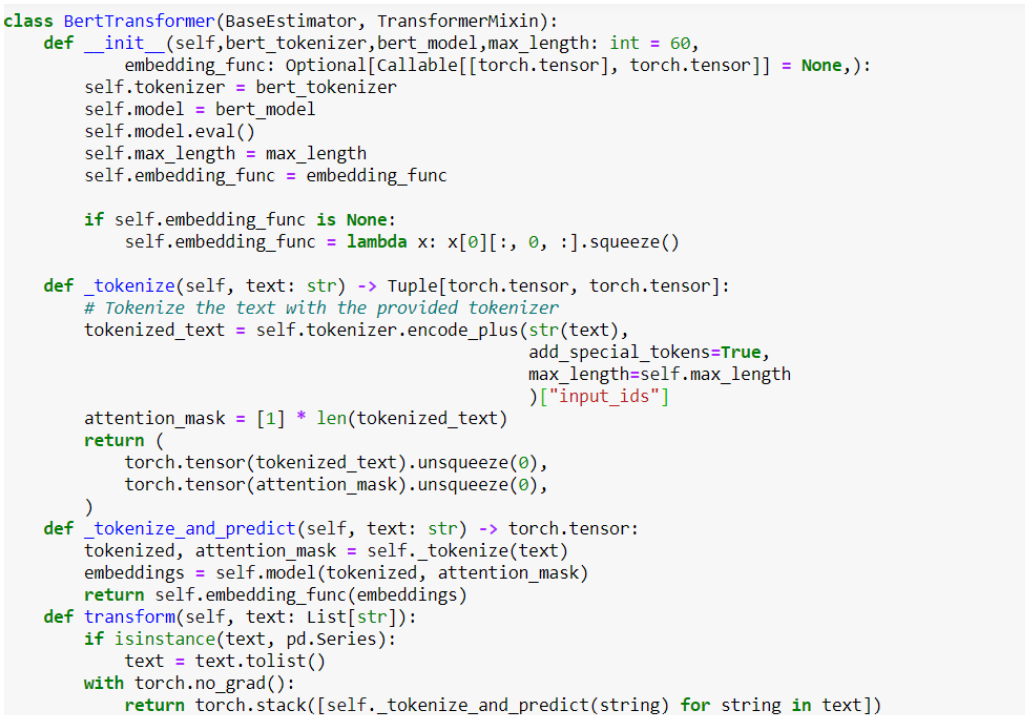
Figure A.1: Distance matrix for Covid-Sentiment dataset

Once the distance matrix is created, then TSC algorithm finds the tolerance classes using the distance matrix. Using the tolerance classes, the tolerance class representative is computed using the mean and median class values of each tolerance class. The decision (text category) of the tolerance class representative is determined based on the majority voting process. At this point learning process is completed and testing step of algorithm begins.

The test dataset loaded similar to the training set of the dataset. Then the distance is calculated between the test samples with the representative class stored from the learning process. The text classification of the test sample is based on which representative class is the nearest to the test sample (minimum Cosine distance). By comparing the predicted text class with the original text category of each test sample in the dataset, the weighted F1-score of the algorithm is computed.

A.1.2 Pipeline for BERT and Scikit-learn

For comparison, we used BERT vectors with Scikit-learn-based classical Ml algorithms. Figure A.2 shows python code to get vector embedding with BERT. Figure A.3 shows the python code to make ML pipeline to combine BERT vectors with classical ML models.



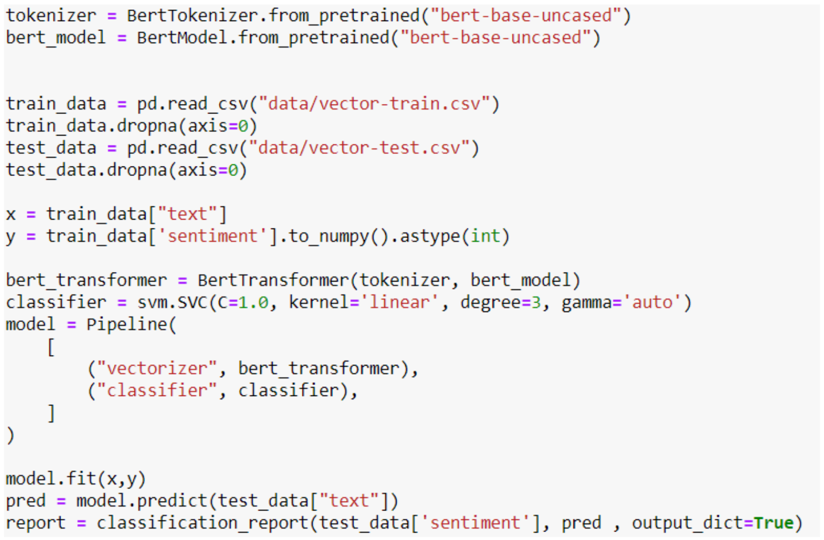
```python
class BertTransformer(BaseEstimator, TransformerMixin):
    def __init__(self,bert_tokenizer,bert_model,max_length: int = 60,
            embedding_func: Optional[Callable[[torch.tensor], torch.tensor]] = None,):
        self.tokenizer = bert_tokenizer
        self.model = bert_model
        self.model.eval()
        self.max_length = max_length
        self.embedding_func = embedding_func

        if self.embedding_func is None:
            self.embedding_func = lambda x: x[0][:, 0, :].squeeze()

    def _tokenize(self, text: str) -> Tuple[torch.tensor, torch.tensor]:
        # Tokenize the text with the provided tokenizer
        tokenized_text = self.tokenizer.encode_plus(str(text),
                                                    add_special_tokens=True,
                                                    max_length=self.max_length
                                                    )["input_ids"]

        attention_mask = [1] * len(tokenized_text)
        return (
            torch.tensor(tokenized_text).unsqueeze(0),
            torch.tensor(attention_mask).unsqueeze(0),
        )
    def _tokenize_and_predict(self, text: str) -> torch.tensor:
        tokenized, attention_mask = self._tokenize(text)
        embeddings = self.model(tokenized, attention_mask)
        return self.embedding_func(embeddings)
    def transform(self, text: List[str]):
        if isinstance(text, pd.Series):
            text = text.tolist()
        with torch.no_grad():
            return torch.stack([self._tokenize_and_predict(string) for string in text])
```

Figure A.2: BERT Embedding python code



```python
tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")
bert_model = BertModel.from_pretrained("bert-base-uncased")


train_data = pd.read_csv("data/vector-train.csv")
train_data.dropna(axis=0)
test_data = pd.read_csv("data/vector-test.csv")
test_data.dropna(axis=0)

x = train_data["text"]
y = train_data['sentiment'].to_numpy().astype(int)

bert_transformer = BertTransformer(tokenizer, bert_model)
classifier = svm.SVC(C=1.0, kernel='linear', degree=3, gamma='auto')
model = Pipeline(
    [
        ("vectorizer", bert_transformer),
        ("classifier", classifier),
    ]
)

model.fit(x,y)
pred = model.predict(test_data["text"])
report = classification_report(test_data['sentiment'], pred , output_dict=True)
```

Figure A.3: ML pipeline code for scikit-learn

Bibliography

[1] Johan Bollen and Alberto Pepe. Modeling public mood and emotion: Twitter sentiment and socioeconomic phenomena. In *Proceedings of the 5th International AAAI Conference on Weblogs and Social Media (ICWSM'11*, page 450–453, 2011.

[2] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[3] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[4] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Céspedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.

[5] Emily Chen, Kristina Lerman, and Emilio Ferrara. Tracking social media discourse about the covid-19 pandemic: Development of a public coronavirus twitter data set. *JMIR Public Health and Surveillance*, 6(2):e19273, 2020.

[6] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(ARTICLE):2493–2537, 2011.

[7] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[8] Nádia Félix Felipe da Silva, Luiz FS Coletta, Eduardo R Hruschka, and Estevam R Hruschka Jr. Using unsupervised information to improve semi-supervised tweet sentiment classification. *Information Sciences*, 355:348–365, 2016.

[9] Yan Dang, Yulei Zhang, and Hsinchun Chen. A lexicon-enhanced method for sentiment classification: An experiment on online product reviews. *IEEE Intelligent Systems*, 25(4):46–53, 2009.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[11] Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. Adaptive recursive neural network for target-dependent Twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 49–54, 2014.

[12] Anastasia Giachanou and Fabio Crestani. Like it or not: A survey of twitter sentiment analysis methods. *ACM Computing Surveys*, 49(2):46, 2016.

[13] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *Processing*, 150, 01 2009.

[14] Ammar Hassan, Ahmed Abbasi, and Daniel Zeng. Twitter sentiment analysis: A bootstrap ensemble framework. In *2013 international conference on social computing*, pages 357–364. IEEE, 2013.

[15] C. J. Henry. *Near Sets: Theory and Applications.* PhD thesis, 2011.

[16] Tu Bao Ho and Ngoc Binh Nguyen. Nonhierarchical document clustering based on a tolerance rough set model. *International Journal of Intelligent Systems*, 17:199–212, 2002.

[17] Xia Hu, Jiliang Tang, Huiji Gao, and Huan Liu. Unsupervised sentiment analysis with emotional signals. In *Proceedings of the 22nd international conference on World Wide Web*, pages 607–618, 2013.

[18] Clayton Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 8, pages 216–225, 2014.

[19] Rajesh Jaiswal and Sheela Ramanna. Detecting Overlapping Communities Using Ensemble-based Distributed Neighbourhood Threshold Method in Social Networks. *Intelligent Decision Technologies Journal, IOS Press*, 15(2):251–267, 2021.

[20] Zhao Jianqiang and Gui Xiaolin. Deep convolution neural networks for twitter sentiment analysis. *IEEE Access*, PP, 01 2018.

[21] Salud M Jiménez-Zafra, M Teresa Martín-Valdivia, Eugenio Martínez-Cámara, and L Alfonso Ureña-López. Combining resources to improve unsupervised sentiment analysis at aspect-level. *Journal of Information Science*, 42(2):213–229, 2016.

[22] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28:11–21, 1972.

[23] Vahid Kardan and Sheela Ramanna. Tolerance Methods in Graph Clustering: Application to Community Detection in Social Networks. In *International Joint Conference on Rough Sets*, pages 73–87. Springer, 2018.

[24] Jasleen Kaur and Jatinderkumar R Saini. A study of text classification natural language processing algorithms for indian languages. *The VNSGU Journal of Science Technology*, 4(1):162–167, 2015.

[25] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30:3146–3154, 2017.

[26] Dimitrios Kotzias, Misha Denil, Nando De Freitas, and Padhraic Smyth. From group to individual labels using deep features. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 597–606, 2015.

[27] Efthymios Kouloumpis, Theresa Wilson, and Johanna Moore. Twitter sentiment analysis: The good the bad and the omg! In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 5, pages 538–541, 2011.

[28] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[29] Ken Lang. Newsweeder: Learning to filter netnews. In *Machine Learning Proceedings 1995*, pages 331–339. Elsevier, 1995.

[30] Jimmy Lin and Alek Kolcz. Large-scale machine learning at twitter. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 793–804, 2012.

[31] Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150, 2011.

[32] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[33] Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. Deep learning based text classification: A comprehensive review, 2021.

[34] Hoora Moghaddam and Sheela Ramanna. Harvesting Patterns from Textual Web Sources with Tolerance Rough Sets. *Patterns Journal, Cell Press, Elsevier*, 1(1):1–20, 2020.

[35] Georgios Paltoglou and Mike Thelwall. Twitter, myspace, digg: Unsupervised sentiment analysis in social media. *ACM Trans. Intell. Syst. Technol.*, 3(4), September 2012.

[36] B Pang and L Lee. Opinion mining and sentiment analysis. foundations and trends (r) in information retrieval, 2 (1-2), 1-135, 2008.

[37] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. *arXiv preprint cs/0205070*, 2002.

[38] Vrushang Patel and Sheela Ramanna. Tolerance-based short text Sentiment Classifier. In *International Joint Conference on Rough Sets, Lecture Notes in Artificial Intelligence*, pages 239–245. Springer, 2021.

[39] Zdzislaw Pawlak. Rough Sets: Theoretical Aspects of Reasoning About Data. Kluwer Academic Publishers, Norwell, MA, USA, 1992.

[40] Zdzislaw Pawlak and Andrzej Skowron. Rudiments of rough sets. *Information Sciences, Elsevier*, 177(1):3–27, 2007.

[41] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[42] J. Peters and S. Naimpally. Applications of near sets. *Notices of the American Mathematical Society*, 59:536–542, 2012.

[43] J.F. Peters. Near sets. General theory about nearness of objects. *Applied Mathematical Sciences*, 1(53):2609–2029, 2007.

[44] J.F. Peters. Near sets. Special theory about nearness of objects. *Fundamenta Informaticae*, 75(1-4):407–433, 2007.

[45] J.F. Peters. Tolerance near sets and image correspondence. *Int. J. of Bio-Inspired Computation*, 1(4):239–245, 2009.

[46] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[47] G. Poli, E. Llapa, J. Cecatto, J Saito, J. Peters, S. Ramanna, and M Nicoletti. Solar flare detection system based on tolerance near sets in a GPU-CUDA framework. *Knowledge-Based Systems Journal, Elsevier*, 70:345 – 360, 2014.

[48] L. Polkowski, A. Skowron, and J. Zytkow. Tolerance Based Rough Sets. In *Lin, T.Y., Wildberger, M. (eds.) Soft Computing: Rough Sets, Fuzzy Logic, Neural Networks, Uncertainty Management, Knowledge Discovery*, pages 55–58, San Diego, 1994. Simulation Councils Inc.

[49] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, pages 1–26, 2020.

[50] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[51] Sheela Ramanna, James F. Peters, and Cenker Sengoz. Application of tolerance rough sets in structured and unstructured text categorization: a survey. In *Thriving Rough Sets*, pages 119–138. Springer, 2017.

[52] Sheela Ramanna and Ashmeet Singh. Tolerance-based approach to audio signal classification. In *Proceedings of 29th Canadian AI Conference, LNAI 9673*, pages 83—88, 2016.

[53] Ankita Rane and Anand Kumar. Sentiment classification system of twitter data for us airline service analysis. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, volume 1, pages 769–773. IEEE, 2018.

[54] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

[55] Stephen Robertson. Understanding inverse document frequency: on theoretical arguments for idf. *Journal of documentation*, 6:521–533, 2004.

[56] Sara Rosenthal, Noura Farra, and Preslav Nakov. Semeval-2017 task 4: Sentiment analysis in twitter. In *Proceedings of the 11th international workshop on semantic evaluation (SemEval-2017)*, pages 502–518, 2017.

[57] Hassan Saif, Miriam Fernandez, Yulan He, and Harith Alani. On stopwords, filtering and data sparsity for sentiment analysis of twitter. In *LREC 2014, Ninth International Conference on Language Resources and Evaluation*, pages 810–817, 2014.

[58] Hassan Saif, Yulan He, and Harith Alani. Alleviating data sparsity for twitter sentiment analysis. In *2nd Workshop on Making Sense of Microposts: Big things*

come in small packages at the 21st International Conference on theWorld Wide Web (WWW'12), pages 2–9. CEUR Workshop Proceedings (CEUR-WS. org), 2012.

[59] Hassan Saif, Yulan He, Miriam Fernandez, and Harith Alani. Semantic patterns for sentiment analysis of twitter. In *International Semantic Web Conference*, pages 324–340. Springer, 2014.

[60] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[61] Cenker Sengoz and Sheela Ramanna. Learning relational facts from the web: A tolerance rough set approach. *Pattern Recognition Letters, Elsevier*, 67(P2):130–137, 2015.

[62] Richa Sharma, Shweta Nigam, and Rekha Jain. Opinion mining of movie reviews at document level. *arXiv preprint arXiv:1408.3829*, 2014.

[63] Ashmeet Singh and Sheela Ramanna. Application of tolerance near sets to audio signal classification. In B. Zielosko, U. Stanczyk, and L. C. Jain, editors, *Advances in Feature Selection, and Data and Pattern Recognition*. Springer International Publishing, 2018. *DOI 10.1007/978-3-319-67588-6 13*.

[64] Andrzej Skowron and Jaroslaw Stepaniuk. Tolerance Approximation Spaces. *Fundamenta Informaticae*, 27(2,3):245–253, August 1996.

[65] Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 455–465, 2013.

[66] Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.

[67] T.Alusaifeer, S.Ramanna, and C.Henry. GPU implementation of MCE approach to finding near neighbourhoods. In *Proceedings of the International Conference on Rough Sets and Knowledge Technology (RSKT2013)*, Lecture Notes in Computer Science, pages 251–262. Springer, 2013.

[68] Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. Learning sentiment-specific word embedding for twitter sentiment classification. volume 1, pages 1555–1565, 06 2014.

[69] Diego Terrana, Agnese Augello, and Giovanni Pilato. Automatic unsupervised polarity detection on a twitter data stream. In *2014 IEEE International Conference on Semantic Computing*, pages 128–134. IEEE, 2014.

[70] Mike Thelwall, Kevan Buckley, Georgios Paltoglou, Di Cai, and Arvid Kappas. Sentiment strength detection in short informal text. *Journal of the American society for information science and technology*, 61(12):2544–2558, 2010.

[71] Peter D Turney. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. *arXiv preprint cs/0212032*, 2002.

[72] Arshia Ulaganathan and Sheela Ramanna. Granular Methods in Automatic Music Genre Classification: A Case Study. *Journal of Intelligent Information Systems, Springer*, 52(1):85–105, 2019.

[73] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[74] Gang Wang, Jianshan Sun, Jian Ma, Kaiquan Xu, and Jibao Gu. Sentiment classification: The contribution of ensemble learning. *Decision support systems*, 57:77–93, 2014.

[75] Piotr Wasilewski, James F. Peters, and Sheela Ramanna. Perceptual tolerance intersection. *Transactions on Rough Sets Journal*, 13:159–174, 2011.

[76] Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[77] Marcin Wolski. Perception and classification. A Note on Near sets and Rough sets. *Fundamenta Informatica*, 101:143–155, 2010.

[78] Bing Xiang and Liang Zhou. Improving twitter sentiment analysis with topic-based mixture modeling and semi-supervised training. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 434–439, 2014.

[79] Alec Yenter and Abhishek Verma. Deep cnn-lstm with combined kernels from multiple branches for imdb review sentiment analysis. In *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*, pages 540–546. IEEE, 2017.

[80] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing. *ieee Computational intelligenCe magazine*, 13(3):55–75, 2018.

[81] E.C. Zeeman and O.P. Buneman. Tolerance spaces and the brain. *Towards a Theoretical Biology, 1*, pages 140–151, 1968. , Published in C.H. Waddington (Ed.), Towards a Theoretical Biology. The Origin of Life, Aldine Pub. Co.

[82] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *arXiv preprint arXiv:1509.01626*, 2015.

[83] Ziqiong Zhang, Qiang Ye, Zili Zhang, and Yijun Li. Sentiment classification of internet restaurant reviews written in cantonese. *Expert Systems with Applications*, 38(6):7674–7682, 2011.

[84] Shaojie Zhu, Bing Xu, Dequan Zheng, and Tiejun Zhao. Chinese microblog sentiment analysis based on semi-supervised learning. In *Semantic Web and Web Science*, pages 325–331. Springer, 2013.