Sparse Structures for Robust Sentiment Analysis of Text

Steven Gunarso and Nicholas Lin

Abstract— **In natural language processing, the task of sentiment analysis attributes a positive or negative categorization to a piece of text. This can be especially useful in the area of social media to detect hate speech or otherwise gather opinions from users. However, user generated content on social media does not often conform to English standards for grammatical structure, spelling, or word usage. Therefore, the need for a robust form of sentiment analysis is needed. In this paper, we will propose a novel sparse word matrix representation structure for sentiment analysis.**

I. INTRODUCTION

A. Background

Sentiment analysis is a task that attempts to extract an opinion or position from a piece of text. Often times, this categorization is either positive or negative. Otherwise known as opinion mining, sentiment analysis is useful for quickly extracting the opinions of users on certain subjects. Especially useful on social media platforms, sentiment analysis can be used to gauge user's opinions on recent events or particular topics. Within the field of Natural Language Processing (NLP), sentiment analysis is solved using a variety of techniques such as Support Vector Machines (SVM), Transformers, and other machine learning (ML) algorithms. In the traditional sentence level sentiment analysis of text, a corpus of text (document, paragraph, page, etc.) is divided into its corresponding sentences. Each sentence can then be used to assign a corresponding categorization of positive or negative.

Social media data presents a unique challenge to the sentiment analysis task. The reality of social media is that often times the text a user generates does not follow grammatical rules. It may be an incomplete sentence, be grammatically incorrect, or contain typos. These challenges make the task of sentiment analysis for social media platforms particularly challenging and also present the need for a robust approach towards text categorization. The nuances of social media data also cause pre-trained language models, such as Google BERT, to have decreased performance.

B. Related Work

Previous work in natural language processing and sentiment analysis has seen various methodologies and techniques. Among these, Google BERT proposed by Devlin et al.[1] is widely known as the most popular natural language processing technique with many works being built off of BERT. Google BERT is a bidirectional encoder created in 2019 by Google Research. While BERT has astounding performance in areas such as question answering, named entity recognition, etc. it has been noted in Jones et al. [2] that BERT can significantly struggle in the robust setting. The accuracy of BERT in sentiment analysis when the dataset contains single letter typos drops from 90% to 45%. It has been noted that pretrained language models will struggle against typos since the dataset they have been pretrained on is non-robust.

Sparse fine tuning of BERT has been explored in Cui et al. [3] with a sparse self-attention mechanism added upon the pretrained BERT Base model. In this paper, the authors demonstrate that the sparse layer can boost BERT performance in sentiment analysis, question answering, and natural language inferencing. However, the paper does not explore the new sparse layer performance on robust examples.

There has been attempts at extracting sparse representation vectors in NLP models. While algorithms, such as word2vec and GloVe, extract sparse representations, they fail to generalize and capture semantic meaning from text [4]. As a result, when faced with adversarial examples, the models perform significantly worse.

Most NLP practitioners use a large pretrained model, such as BERT or GPT-n, to generate CLS embeddings to train their own classifiers. While this achieves a reasonable performance, these embeddings rely on dense representation vectors [4]. Because of this, this ML pipeline is often unable to handle adversarial examples at an acceptable level.

C. Justification

NLP models tend to perform disproportionately poorly on adversarial examples (eg. typos, word substitutions, synonyms). As opposed to computer vision (CV) tasks, the input in an NLP problem is in the form of a large, but discrete space. While some of the current algorithms used by many ML practitioners, such as word2vec and GloVe, attempts to extract sparse representation vectors from text in the form of "embeddings," they fail to represent and evaluate context and homonyms very well. Other distributional approaches produce word representations that are dense and low-dimensional in nature. When these examples are exposed to adversarial or out-of-distribution (OOD) examples, they tend to overfit the decision boundary as commonly seen in adversarial training. This is a step backwards from generalization and induces memorization. As language evolves, especially in social media, these models will not be able to adapt to these changes. There is an obvious need for a more robust method to perform sentiment analysis. Pretrained models are also computationally expensive and large to add sparsity onto. In this paper, we will explore a lightweight domain specific language model to compare its performance on robust examples.

II. FORMULATION

A. Conditions, Goals, Novelty

This experiment will attempt to produce a novel approach to extracting sparse representations that is robust in the face of adversarial examples. We will use the publicly-available dataset from Twitter to train our own language model and compare this model against other pretrained models (eg. Google's BERT, OpenAI's GPT-n models) when exposed to adversarial examples. The task that we will evaluate the model against is binary sentiment classification. That is, we expect a classification of positive or negative for the sentiment of the tweets. In the real world application, this classification corresponds to the area of content moderation and hate speech detection. The goal is to use robust structures to improve model performance across classifying adversarial examples.

The following section details the metrics that will be evaluated to compare NLP models.

1) F-1 Score: The f-1 score computes the harmonic mean of recall and precision. The formula for the f-1 score is:

$$f_1 = 2 * \frac{recall * precision}{recall + precision}$$

This metric allows a more comprehensive evaluation of performance across imbalanced datasets.

2) Generalization Gap: The generalization gap metric is extremely important as it measures the difference between a model's performance on training data and performance on test data. The formula for generalization gap is:

$$E_{gap} = |E_{train} - E_{gen}|$$

where training error is:

$$E_{train} = \frac{1}{n}\sum_{i=1}^{n} V(f(x), y)$$

and generalization error is:

$$E_{train} = \frac{1}{n}\int V(f(x), y)p(x, y)dxdy$$

3) Perplexity: Perplexity is a measure of the performance of a model to predict the test data. The most common definition of perplexity is the model that has the highest probability to the test set. The formula for perplexity is:

$$PP(W) = \sqrt[N]{\frac{1}{P(w_1, w_2, ..., w_N)}}$$

B. Adversarial Examples

There are two main types of adversarial examples that we will consider for natural language processing. We will examine the performance of the sparse structure approach against each of these types of adversarial examples individually to determine the robustness of using sparse examples.

1) Typos: In this form of adversarial example, the text contains small typos that to the human eye do not impact the meaning of the sentence. For example, duplicate letters is a form of typos. The sentence, "He is smarrt" contains a typo where the letter 'r' is duplicated within the word smart. Although the human reader can quickly detect typos and understand the sentiment of the text, machine learning algorithms struggle with such examples as their frequency across the training set is minimal. As many approaches view individual words as tokens and unique tokens for different spellings, the result of typos can significantly decrease a model's ability to make a sentiment prediction. Google BERT notoriously struggles against typos in sentiment analysis and can drop from 90% accuracy to 45% accuracy when duplicate letter typos are introduced. Add Reference

2) Semantic Modifications: The second form of adversarial examples are changes to structure of a sentence that do not directly impact meaning. These examples involve transforming sentences from active sentences to passive sentences. For example the statement, "the chef loved the pasta" would be manipulated to be "the pasta was loved by the chef". Again in this form of adversarial example, a human reader can easily detect the difference in meaning, but most language models learn patterns from sentences by their structure and the order in which words (tokens) appear. This form of adversarial example accounts for situations where users may express grammatically incorrect statements but contain obvious sentiment.

C. The Dataset

1) Twitter Data: The twitter dataset contains 15,000 tweets that have been classified to be positive or negative sentiment. In our analysis, we will drop the neutral classification option and make the task binary classification. Tweets are short corpuses of text that a user produces (no more than 280 characters). The Twitter dataset contains 7,781 negative training samples and 8,582 positive training samples. Examples of tweets include "Sooo SAD I will miss you here in San Diego!!!" and "Journey!? Wow... u just became cooler. hehe...." as examples of negative and positive sentiments.

III. METHODS

For the classification task, let $x \in \mathbb{R}^V$ denote the vector representation of a word. V denotes the vector space of the vocabulary (all word tokens). Furthermore, we have a binary response variable $y \in \{-1, +1\}$. The goal of the task is to learn a weight vector $w \in \mathbb{R}^V$ such that the loss function $L(x, w, y)$ is minimized.

$$L(x, w, y) = \log(1 + exp(-yw^T x))$$

Here the logloss function is utilized for learning. The objective function for learning (without any regularization so far) then becomes:

$$\hat{w} = \arg\min_{w} \sum_{i \in I} \log(1 + exp(-y_i w_i^T x_i))$$

where the set I contains all the training examples.

A. Sparsity of w

The first area for sparsity that we will investigate is within the vector weights w. The w vector gives a sense of how much contribution each word in the sentence has towards the actual sentiment. Intuitively, only a few words within a sentence determine its sentiment. We want to provide w as sparse as possible since this indicates that one or two words justify the sentiment of a sentence. Therefore we propose regularization for w on a sentence level. Let $\Phi(x)$ be the regularization function:

$$\Phi(w) = \sum_{i \in I} \sum_{s \in S} \sum_{c \in C} \lambda_w ||w||_2$$

which is the l-1 norm minimization across the weights. Here, I is the set of training examples, S it the set of sentences within each training example and C is the set of parse tree nodes of a particular sentence. The parse tree representation of a sentence is a regularization aspect that will be explored in the next section.

B. Sparsity of x

The next area for sparsity that we will investigate is in the word representation form. There are several popular methods for word representations. One of which is word2vec which creates an embedding of each word that improves cosine similarity between words. This representation is not sparse since it is a compressed representation of words.

$$\Omega(x) = \sum_{i \in I} \sum_{x \in V} \lambda_x ||x||_2$$

In this research, we will attempt to use a Semantic Transformation method designed specifically for semantic analysis to be able to utilize the desirable properties of sparse representations. While sparsity is achieved through "penalties," such as l_1 regularization. A better method that we apply here is the use of an activation function [6]. The activation function is as follows:

$$S(x) = e^{-(\beta x - \gamma)^2} - e^{-(\beta x + \gamma)^2}$$

where β and γ are hyperparameters that regulate the intended sparsity.

C. Loss function for Learning

The final optimization function that we will be solving is:

$$\hat{w} = \arg\min_{w,x} \sum_{i \in I} \log(1 + exp(-y_i w_i^T x_i)) + \Phi(w) + \Omega(x)$$

References

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training ofdeep bidirectional transformers for language understanding,” 2019.

[2] E. Jones, R. Jia, A. Raghunathan, and P. Liang, “Robust encodings: Aframework for combating adversarial typos,” 2020.

[3] B. Cui, Y. Li, M. Chen, and Z. Zhang, “Fine-tune BERT with sparse self-attention mechanism,” inProceedings of the 2019 Conference on EmpiricalMethods in Natural Language Processing and the 9th International JointConference on Natural Language Processing (EMNLP-IJCNLP), (HongKong, China), pp. 3548–3553, Association for Computational Linguistics,Nov. 2019.

[4] D. Carrano, “Geometric Properties of Backdoored Neural Networks,” 2021.

[5] D. Yogatama, "Sparse Models of Natural Language Text," 2015.

[6] H. Wenpeng, M. Wang, B. Liu, F. Ji, H. Chen, D. Zhao, J. Ma, and R. Yan, "Transformation of Dense and Sparse Text Representations," 2019.

[7] S. Vaswani, "Exploiting Sparsity in Supervised Learning,"