A Hybrid Approach to Insincere Questions Detection

Mengxuan Lyu, Jinyue Feng
University of Toronto
April 6, 2019

1 Abstract

As Q&A sites become more prevalent, it is critical to detect and reveal patterns of misleading or discriminative questions. In this project, we studied Quora with the focus on questions with non-neutral tone, rhetorical questions, discriminative questions, and questions with sexual contents as shock values. In natural language processing, a lot of research has been conducted over sarcasm, and insincere questions share substantial similarities with it from linguistics interpretations. Therefore, we transferred and further extend state-of-art approaches of sarcasm detections. In this paper, we proposed a baseline model which is composed of CNN-LSTM and a hybrid model argumented with corpus statistics. Both models utilise ELMo(Peters et al., 2018) as the embedding layer and fine-tuned it within the framework provided by Allennlp[1]. Second, we conducted an in-depth corpus analysis over Quora questions, such as punctuation scores, part-of-speech tagging, named entities, and sentiment scores. Third, we performed evaluations with precision-recall metrics and compare the results with the intended hypothesis. Finally, our hybrid model claimed to outperform the baseline model. Codes are available on Github[2]

[1] https://allennlp.org/
[2] https://github.com/ShawnLYU/Quora-Insincere-Questions-Classification

2 Introduction

Quora is an emergent Q&A site where people could ask, answer, and edit questions to share knowledges. However, despite Quora's policy of *"Be Nice, Be Respectful"*, misleading or discriminative questions of various types still exist. In this project, we will primarily focus on questions with non-neutral tone, rhetorical questions, discriminative questions and questions that use sexual content as shock value. After reviewing sentiment analysis studies, we have found that sarcasm detection is an extensively studied topic in this field. However, there are few studies on the classification of insincere or toxic questions. A closer look at the linguistic backgrounds of sarcasm and insincere questions revealed that these two phenomena share many similarities, which we will discuss in details in section 3.1. Therefore, we are motivated to transfer and extend sarcasm detection approaches to our project.

The goals of this project are:

1. to develop a model that detects insincere questions to improve the quality of online conversations;
2. to experiment on how different word embeddings and classification techniques suit this particular problem;
3. to present and visualize corpus statistical information with an emphasis on how this information would affect the classification results;
4. to explore the linguistic nature of insincere question expressions.

2.1 Project Design

2.1.1 Text to Vector

The first part of our project would be mapping text data into numerical space for the further downstream task of classification. Here we adopt three ways of converting.

- Textual Space

  This process involves preprocessing of texts including tokenization and PoS-tagging, feature extraction such as counting the number of adjectives, and application of machine

learning algorithms such as Support Vector Machine (SVM) and Random Forests (RF) for classification. We will analyze features that have the highest impact on insincerity detection to reveal some statistical information concerning the expression profile of insincere questions.

- Embedding Space

  This approach requires the usage of libraries and the implementation of CNN-LSTM models using TensorFlow and PyTorch. We will have a closer examination of these techniques in the following literature review. In addition to the attempt of achieving satisfactory predictions, we will also visualize the relationships among word vectors to help explore the semantical information about insincere questions.

- Modularization and Integration

  As we will discuss in the following sections, the combination of different modularized components in a classification process may yield better performance than any module alone. We intend to deploy innovative methods to merge the textual space and embedding space modules.

2.1.2 Classification

The second part would be implementing different models, including SVM, RF, and neural networks as well, to perform classification over mapped data.

3 Literature Review

3.1 Relations to Sarcasm

As we have introduced before, a closely related topic to insincere question classification is sarcasm detection, which is a frequently researched area in sentiment analysis(Joshi and Bhattacharyya, 2017). Sarcastic messages and insincere questions share many similarities regarding their linguistic nature:

1. **Sentiment Involvement.** Both sarcasm and rhetorical questions involve non-neutral

sentiments under the disguise of a propositional or interrogative structure (Joshi and Bhattacharyya, 2017; Schmidt-Radefeldt, 1977).

2. **Presence of Indicative Words.** One type of insincere questions contain expressions of exclusive absoluteness, such as *"if not"*; other rhetorical questions have non-deontic modal verbs and some other special particles that strongly indicate its persuasive, not interrogative, nature (Schmidt-Radefeldt, 1977). Similarly, sarcastic texts also involve indicative words, such as *'like'* in *"like you understand"* (Joshi and Bhattacharyya, 2017).
3. **Dropped Negation.** Many sarcastic sentences and rhetorical questions are meant to make a negative statement despite the absence of negation words (Joshi and Bhattacharyya, 2017; Schmidt-Radefeldt, 1977). For instances, "Having a cold is so fun" means "Having a cold is not fun at all", and "Can such a man be innocent?" means "Such a man cannot be innocent."
4. **Intended Victim.** Sarcastic comments can be used to mock a victim, and insincere questions can be discriminative or disrespectful against certain groups of people. The harmful components in both situations may be implicit or explicit.
5. **Violation of Truthfulness.** Sarcastic comments that violate truthfulness resemble insincere questions that are based on false premises. To understand such languages, the listener needs to know what is the true background information and how the truthfulness is violated(Joshi and Bhattacharyya, 2017). This can be a hard problem in natural language processing as it largely depends on knowledge about certain people or certain topics. However, it is also possible that the untruthful texts have syntactic or semantic characteristics that can become features of machine learning classification models.

Based on these linguistic characteristics, we hypothesize that methods that are successful in sarcasm detection will achieve satisfactory performance on insincerity detection. However, we also expect some challenges in transferring the classification approaches from sarcasm to insincerity detection problem. First and foremost, the five properties listed above do not directly serve as classification features for either problem. They merely prove the similarity between the two problems and form the basis of our hypothesis from a linguistic point of view. Second, sarcastic patterns that have been trained into accurate statistical classifiers for sarcasm detection

do not directly apply to insincerity detection. For example, sarcastic messages often feature a conflict between opposing sentiments in the same sentence (Joshi and Bhattacharyya, 2017), which may not be a significant property of insincere questions. We would have to develop feature sets specifically designed for insincerity detection using both statistical feature selection methods and knowledge-based heuristics. Third, although deep learning-based approaches have achieved state-of-art performance and gained popularity in sarcasm detection (Joshi and Bhattacharyya, 2017), such studies typically focus more on the analysis of the models than the scientific nature of the phenomena. In other words, we observe a gap between the linguistic basis and the learning algorithms. The lack of explanations of why neural networks perform well on sarcasm detection increases the difficulty of transferring sarcasm detection models to our problem domain; however, we also consider this as an opportunity for us to contribute to the knowledge base of insincerity detection.

3.2 Statistics-Focused Features

Statistics-focused features in sentiment classification follow an established pattern where four types of features are frequently examined, namely term frequencies, parts of speech counts, the presence of opinion words, and negations(Medhat et al., 2014). The most commonly used statistical feature selection methods include point-wise mutual information, Chi-square, and latent semantics indexing (Medhat et al., 2014). Generally speaking, these methods measure the statistical relationship of word occurrence and class identification. These feature selection methods, combined with classifiers such as SVM, have been shown to be effective in certain sentiment analysis tasks (Medhat et al., 2014).

In our study, we focus more on presenting corpus statistics and analyzing how such statistical information contribute to a neural-network-based classification process. A particularly relevant study was conducted by Barbosa and Feng (2010), where the authors presented a 2-step sentiment analysis method that only utilized an SVM model for the machine learning process, but its feature extraction methods allowed for robustness against noisy data. The researchers developed two feature sets designed to provide abstract representations of short texts: meta-features (including PoS tags and prior subjectivity and polarity) and syntactic features (namely frequencies of certain

types of characters). The detection process was divided into two parts: subjectivity detection and polarity detection of the subjective texts. The experiment followed a traditional approach where the classifiers with the highest information gain were created using Weka, and the learning process was completed using SVM algorithm(Barbosa and Feng, 2010). Although this method no longer offers state-of-art performance, we still consider the feature selection methods valuable to our project. We intend to adapt the design in this study to obtain measurements of corpus statistics to feed as features into our neural network models.

3.3 Word Embeddings

To embed our text data into vector space for the downstream tasks, we adopted word embeddings in addition to the statistics-focused features.

A study on word embedding-based features (Joshi et al., 2016) provides insights on how to combine word embeddings with feature selection. This work examined how features calculated from word embedding vectors could augment existing feature sets including n-grams, dictionary-based features, syntactical features, and pattern-based features. More specifically, the researchers attempted to use word vector distances to detect context incongruity independent of sentiment changes. The results indicated that word embedding-based features enhanced performance (Joshi et al., 2016). This method might be useful in our project for two reasons: 1) insincere questions may also contain context incongruity because of conflict between asking a question while making a statement; 2) questions that have discriminative or sexual language may be detected based on the semantic similarities to a set of sensitive keywords. In this system, known sources of errors include multiple-sense-induced embedding issues, incapability to identify contextual information, and non-sarcastic metaphors(Joshi et al., 2016). We expect the latter two points have a lower impact on insincerity detection than sarcasm detection because contextual information and metaphors are less relevant to insincere questions than sarcastic messages.

3.4 Neural Networks Models

Here we review several studies on sarcasm detection and sentiment analysis in the hope of transferring core concepts and methods to insincere question classification.

Compared to long texts that contain more contextual information, short texts sentiment analysis can be more challenging, and thus methods such as bag-of-words or n-grams are shown to be less effective (Barbosa and Feng, 2010; dos Santos and Gatti, 2014). (dos Santos and Gatti, 2014) proposed a deep CNN model that constructed word embeddings from character-level to sentence-level for sentiment analysis. The character-level embeddings captured morphological information and the word-level embeddings catch syntactical and semantical information (dos Santos and Gatti, 2014). Finally, the sentence-level representations were constructed upon concatenated character-level and word-level features. The word-level embeddings came from Word2Vec embeddings, and the character-level and sentence-level were obtained using convolutional layers that successively extracted local features in windows and maxed over these windows to get the global representation (dos Santos and Gatti, 2014). According to the results, the extracted features at sentence-level typically concentrated on several sentiment-bearing keywords. On the other hand, the contributions of character-level representations were not closely examined, so to what degree morphological and shape information benefited sentiment analysis was not clear. Our project will primarily focus on semantical and syntactical analysis, but we would also like to explore the advantages of character-level embeddings given enough time and resources.

Poria et al. (2017) presented a CNN architecture that contained in parallel a baseline 2-layer CNN model that directly process the text and three pre-trained models covering three types of clues (in particular, sentiments, emotions, and personalities) that could be beneficial for sarcasm detection. These pre-trained models were trained on their benchmark datasets and then used for feature extraction on the sarcastic tweets datasets. Unlike the CNN model that specialized in obtaining textual information in-depth (dos Santos and Gatti, 2014), the four-component CNN model proposed by Poria et al. (2017) horizontally extracted information from different aspects. Although the baseline features (directly extracted from the word vectors) showed better predictions than all pre-trained models, the combination of the baseline and pre-trained models yielded the best performance (Poria et al., 2017). To merge the outputs of the four parts, the researchers tested two methods: combining the outputs using SVM and appending extracted features from the pre-trained models into the baseline CNN hidden layers (Poria et al., 2017).

The former method showed better results possibly because appending the extracted features into the baseline models compromised their meaningfulness.

Another sarcasm detection study (Ghosh and Veale, 2016) presented a CNN-LSTM-DNN neural networks model for semantic modeling, and this model architecture was particularly relevant to our project design. Consider that the information-bearing parts of text could exist anywhere, the choice of convolutional neural networks and recurrent neural networks was reasonable as both CNN and RNN were capable of capturing sequential temporal information. On top of the LSTM layers, the researchers added a fully-connected DNN layer to map the features into a more separable space. The combination of CNN, LSTM, and DNN showed more superior results than recursive SVM, two-layer CNN, and two-layer RNN models (Ghosh and Veale, 2016). Consider the similarity between the tasks of this study and our project, we intend to construct our neural networks architecture following this CNN-LSTM pattern.

4 Experimental Setup

4.1 Overview

When designing this program, we implemented each part of the model into modules. As shown in Figure 1, Quora question samples that were in textual form were firstly embedded into numerical space with ELMo. Next, these embedded vectors were fed into a CNN-LSTM network, which served as our baseline model. Additionally, statistical features including PoS-tags, punctuation marks, named entities, and sentiment scores were computed using existing libraries. The output of the baseline model and statistical features were then merged using fully connected layers with dropout. The feature vector computed by the final hidden layer were fed to a softmax classifier. Evaluation metrics were applied at the end of the experiments.

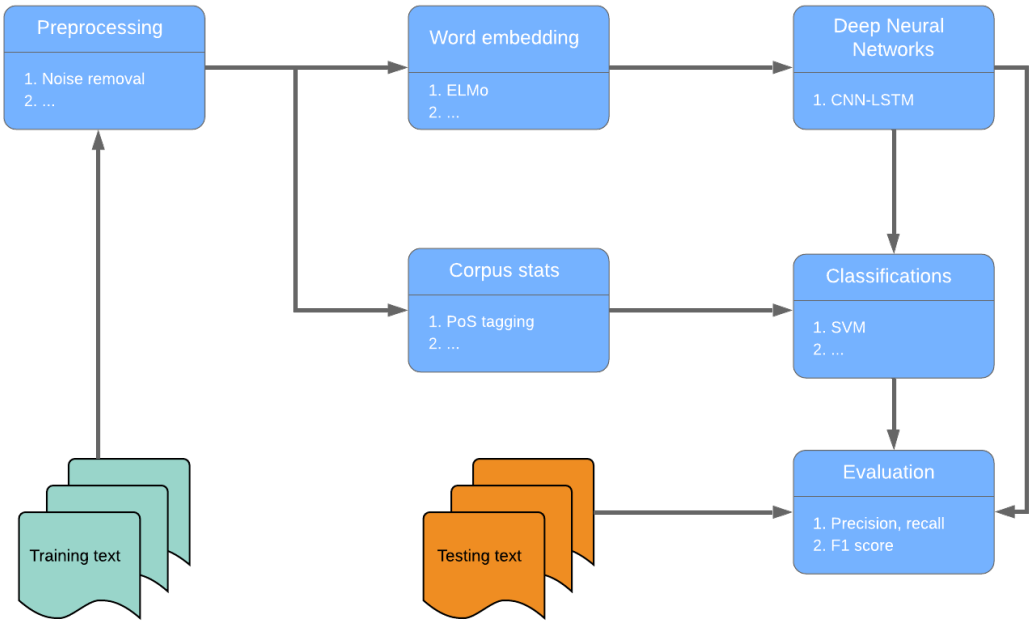


Figure 1: Program design

4.2 Data Preparation

This project originated from a Kaggle challenge. Kaggle provided a dataset of 1,306,122 questions with three fields [3], namely *qid* (a question id), *question_text* (the content of a question) and *target* (where insincere questions were labeled as 1).

For the entire project, we relied on pandas library for data analysis because of its easiness to use and high performance. After a general analysis of the dataset, we figured out that most of the questions were around 50 tokens in length, and there were only 22 sentences that were longer than 100 tokens. Thus we removed these 22 sentences to improve training efficiency. These process left us with 1,044,839 training examples.

We purposely did not treat our data with the commonly used stop-word removal step for our analysis, because previous studies have shown that stop-word removal in Twitter sentiment analysis tasks could hinder model performance given that the typical stop-word lists were not applicable to Twitter corpus (Giachanou and Crestani, 2016). Similarly, stop words in short Quora questions may also serve as useful information for classification.

Another pre-processing step we conducted was correcting one specific set of noisy data,

[3] https://www.kaggle.com/c/quora-insincere-questions-classification/data

namely questions involving mathematical formulas. The problem of wrongly labeled mathematical questions in this dataset was a known issue according to the Kaggle discussion forum. Since the presence of mathematical formulas could be a strong indicator of sincere questions, we wanted our model to accurately capture this information.

After all the preparation steps, we divided the dataset into a training set of 835827 questions, a validation set of 209011 questions, and a testing set of 261257 questions.

4.3 Corpus Statistics

For corpus statistics analysis, we used Pandas for data handling, spaCy, TextBlob, and NLTK for natural language processing, SciPy for statistical testing, and Matplotlib for plotting.

We looked into the statistical information in four major aspects: punctuation marks, part-of-speech tags, named entities, and sentiment scores. Although spaCy PoS-tagger includes PUNCT (punctuation) as one lexical category, we developed our own list of punctuation marks for finer-grained details. The PoS tags used followed the Universal Dependencies scheme (pos, 2017), and the named entities list followed the categorization provided by spaCy Named Entity Recognition model trained on OntoNotes 5 corpus (spa, 2019). For each specific item in these categories except for sentiment scores, we computed the frequencies of an item in each sentence. For sentiment scores, we directly used TextBlob library. After collecting the statistics, we conducted the Kolmogorov–Smirnov test (KS test) to compare their distributions in positive and negative samples. Since we used two-tail tests, the p-value threshold for significance is 0.01. All the tested features are listed in Table 1. For all the significant features, we also generated a series of graphs to visualize the distributions.

Using the KS test for discrete distributions was a compromised choice because the test was less powerful if the distributions were not continuous. The large sample size was also problematic as it reduced the power of significance testing. However, one benefit of this approach was that the conservative reduction of the number of features allowed us to conduct more comprehensive corpus statistics analysis.

To visualize extracted statistical features about punctuation, parts-of-speech tag, and named entities at corpus level, we created bar plots for the counts of the occurrences over the entire

	punctuation marks	PoS tags	named entities	sentiment
Statistically Significant Features	all punctuation marks - (from string.punctuations) commas periods quotation marks question marks other punctuation marks	ADJ: adjective ADP: adposition ADV: adverb CCONJ: coordinating conjunction DET: determiner NOUN: noun NUM: numeral PART: particle PRON: pronoun PROPN: proper noun VERB: verb	PERSON - People, including fictional. NORP - Nationalities or religious or political groups. ORG - Companies, agencies, institutions, etc. GPE - Countries, cities, states. LOC - Non-GPE locations, mountain ranges, bodies of water. DATE - Absolute or relative dates or periods. CARDINAL - Numerals that do not fall under another type.	sentiment polarity
others	exclamation marks	AUX: auxiliary INTJ: interjection SCONJ: subordinating conjunction SYM: symbol X: other	FAC - Buildings, airports, highways, bridges, etc. PRODUCT - Objects, vehicles, foods, etc. (Not services.) EVENT - Named hurricanes, battles, wars, sports events, etc. TIME - Times smaller than a day. PERCENT - Percentage, including "%". MONEY - Monetary values, including unit. QUANTITY - Measurements, as of weight or distance. WORK_OF_ART - Titles of books, songs, etc. LAW - Named documents made into laws. LANGUAGE - Any named language. ORDINAL - "first", "second", etc.	

Table 1: Features for KS tests

corpus. To visualize the distributions at the sentence level, we used boxplots. For certain features of which the distributions are extremely skewed, our visualization methods may not provide the best representation. Therefore we incorporated different types of plots to complement each other to partially solve this problem. For example, we created a pie chart for punctuation marks to show the relative distribution in one sample population, and joint KDE plots for sentiment scores.

4.4 Experiments and Expectations

Our experiments contain two primary goals: one is to investigate the performance of a similar architecture to state-of-art sarcasm detection models on insincerity classification; the other is to explore how pre-extracted corpus-statistical and sentiment features contribute to the classification results.

Based on the similarities of the linguistic natures of sarcasm and rhetorical questions, we expected that our baseline CNN-LSTM model will achieve good performance on the task

because similar models are proven to be successful in sarcasm detection. Once we achieve reasonable results with our baseline model, we will proceed to add different components of the pre-extracted data into the neural network and evaluate their effects on the results. Since information such as sentiment scores directly correlates with the indicators of insincere questions, we anticipate that directly feeding such information to the network, instead of relying on word embeddings as semantic representation, should improve our model's performance. Without empirical results, it is difficult to predict whether statistical features such as punctuation marks counts will augment our model. But consider that the distributions of extracted statistical features were significantly different in labeled and unlabeled data population, we expect some improvements in the evaluation metrics.

Additionally, to understand the nature of insincere questions, we will use the statistical tests and visualization methods described in the previous sections to conduct the analysis. We will compare and contrast the two populations to identify what aspects of the text tend to indicate insincerity.

5 Methodology

5.1 Problem Definition

Let the dataset be $X = \{X_1, X_2, \cdots, X_N\}$ so that $X_i = \{t_1, t_2, \cdots, t_k\}$, where t_i is the i^{th} token, $i \in [1,k]$. Besides, we also have $Y = Y_1, Y_2, \cdots, Y_N$ where $Y_i \in \{0,1\}$ so that 1 indicates the sample is labeled as insincere. Firstly each textual sample would be embedded using both ELMo and corpus statistics (shown as model S) as:

$$\begin{aligned} R_i &= [ELMo1, ELMo2, \cdots, ELMo_k] \\ C_i &= S(X_i) \end{aligned} \tag{1}$$

For model M taking X_i as input, it would compute the output as $\widetilde{Y}_i = M(X_i) = M(R_i, C_i), \widetilde{Y}_i \in \{0,1\}$. Therefore, the performance of our model would be evaluated by:

$$
\begin{aligned}
Precision(X_i) &= Pr(Y_i = 1 \mid \widetilde{Y}_i = 1) \\
Recall(X_i) &= Pr(\widetilde{Y}_i = 1 \mid Y_i = 1) \\
F1 &= 2 * \frac{Precision(X_i) Recall(X_i)}{Precision(X_i) + Recall(X_i)}
\end{aligned} \tag{2}
$$

The aim of this project is to use corpus statistics to augment the performance of standard CNN-LSTM for insincerity classification.

5.2 Word Embedding

To better exploit information embedded in the short contexts of Quora questions, we applied Embeddings from Language Models(ELMo) to embed our text data into vector space for the downstream analysis instead of naive one-hot encoding.

A critical obstacle of language modeling is caused by the curse of dimensionality. To address this problem, in 2003, Bengio et al. (2003) proposed a feed-forward Neural Network Language Model (NNLM) to learn a densely distributed representation(Hinton et al., 1986) for words, where both the word feature vectors and the parameters of that probability function were learned at the same time. NNLM is designed to learn $f\left(w_t, \cdots, w_{t-n+1}\right) = \hat{P}\left(w_t | w_1^{t-1}\right)$, $w_t \in V$, where $w_1 \cdots w_T$ is the word sequence and V is a large but finite vocabulary. This objective is divided into two steps: firstly, the previous N words would be mapped into real vectors by a share projection matrix, which represents each word of the vocabulary with a distributed feature vectors; then, the sequence of word feature vectors would be mapped into a probability distribution over V. As a result, this model yielded better perplexity compared to N-gram models and inspired more researchers to look into distributed representations of words.

Recognizing the need to include distributed representations of words introduced by Hinton et al. (1984), Mikolov et al. (2013) analyzed NNLM and proposed two models to learn the words' continuous vector representations. Originally, NNLM mainly consisted of three layers: words were first mapped into embeddings at projection layer; then embedded vectors flowed into an ordinary hyperbolic tangent hidden layer; finally, the output layer represented the probability

distribution over V. Therefore, most of the computation happened in the output layer. To learn the continuous word representations with higher accuracy and lower computational cost from a huge corpus, the first model Mikolov et al. proposed was the Continuous Bag-of-Words Model (CBOW) which removed the non-linear hidden layer. In CBOW, embedded words would be first projected and then averaged into the same position, after which an output layer would be presented with hierarchical softmax. They also provided another model that was similar to CBOW called Continuous Skip-gram Model to predict the context given a word. The result turned out to be promising as the model did not only extract the syntactic regularities but also revealed subtle semantic information of words.

As many previous models tried to embed words into same vectors, they failed to model polysemy properly. Peters et al. (2018) therefore proposed the Embeddings from Language Models (ELMo) representations which is a new contextualized word representations to explore syntactic and semantic features of words and also how they vary in different contexts by embedding each token as a function of the input space. Normally, a language model would try to predict the next token given the previous context: $p(t_1,t_2,\ldots,t_N) = \prod_{k=1}^{N} p(t_k|t_1,t_2,\ldots,t_{k-1})$. Instead, a backward language model processes the sentence reversly, trying to predict the previous token provided with the furture context: $p(t_1,t_2,\ldots,t_N) = \prod_{k=1}^{N} p(t_k|t_{k+1},t_{k+2},\ldots,t_N)$. ELMo adopted a bidirectional language model(biLM) which combines both forward and backward language models: $\sum_{k=1}^{N}(\log p\left(t_k|t_1,\ldots,t_{k-1};\Theta_x,\overrightarrow{\Theta}_{LSTM},\Theta_s\right) + \log p\left(t_k|t_{k+1},\ldots,t_N;\Theta_x,\overleftarrow{\Theta}_{LSTM},\Theta_s\right))$. Therefore, for a L-layer biLM using LSTM, there will be $2L+1$ representations for each token t_k:

$$\begin{aligned} R_k &= \left\{\mathbf{x}_k^{LM}, \overrightarrow{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM} | j = 1,\ldots,L\right\} \\ &= \left\{\mathbf{h}_{k,j}^{LM} | j = 0,\ldots,L\right\}, \end{aligned} \tag{3}$$

where $\mathbf{h}_{k,0}^{LM}$ is token layer while $\mathbf{h}_{k,j}^{LM} = [\overrightarrow{\mathbf{h}}_{k,j}^{LM}; \overleftarrow{\mathbf{h}}_{k,j}^{LM}]$ refers to biLSTM layers. Then, ELMo would integrate all representations into one single vector for each specific downstream NLP tasks with $\mathbf{ELMo}_k^{\text{task}} = E\left(R_k;\Theta^{task}\right) = \gamma^{\text{task}} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}$.

In our project, ELMo was applied and the task-specific weighting of all three layers of representations shown as the equations 1 in Peters et al. (2018) (*i.e.* character-convnet output

and two LSTM outputs) would be find-tuned.

5.3 Neural Networks

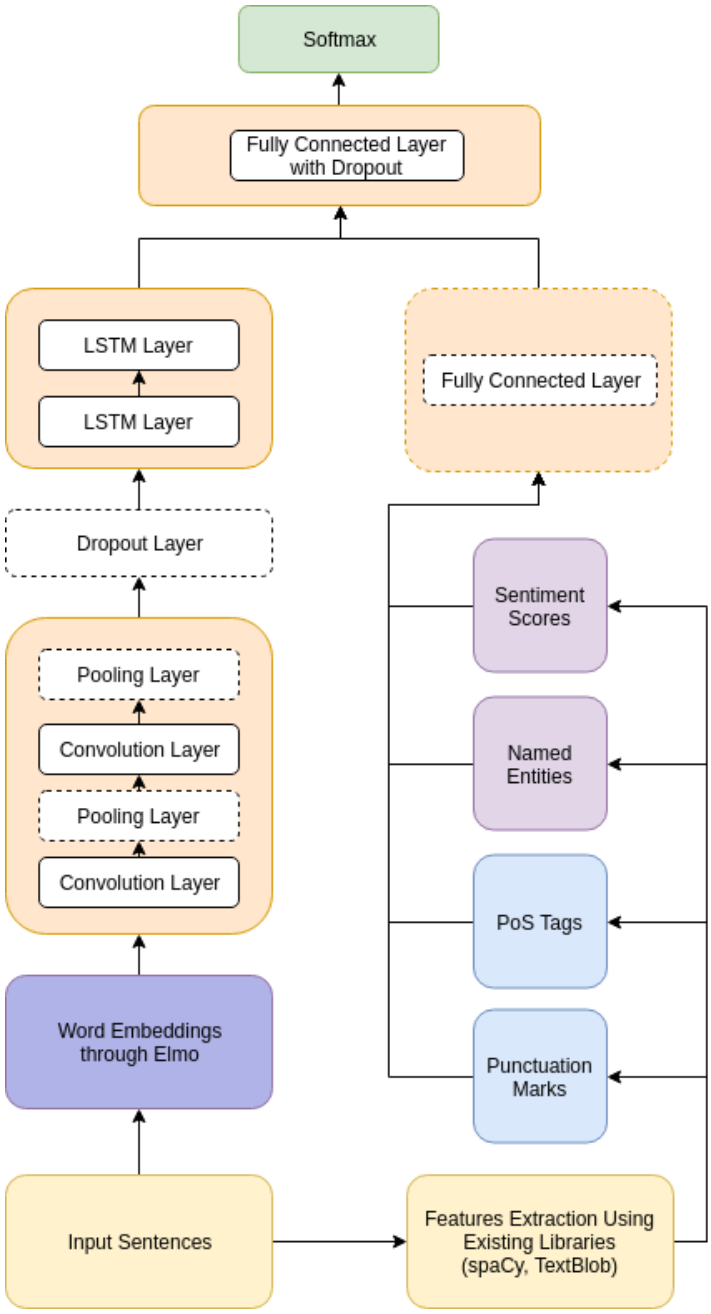


Figure 2: Illustration of neural networks architecture

In this figure, the same types of components are shown in the same colors. The blocks with dashed lines correspond to layers that typically appear in neural networks, but possibly hinder performance according to previous studies. We decided to construct a baseline model without these layers and add them in later to compare the performance if time allows.

Based on our review of neural network models in the field of sentiment analysis, we decided

to adopt a CNN-LSTM-DNN model as illustrated in Figure 2.

Studies have shown that compared to completely distributing word representations, modularized components with specific tasks could improve the overall performance of a CNN model (Poria et al., 2017). As we discussed in the previous sections, we have established several characteristics that indicate a Quora question to be insincere; therefore, the idea of combining various components each responsible for detecting different classification clues can be valuable for our project. In our design, the network is generally divided into two components: a baseline CNN-LSTM model taking word vectors as inputs and extracted syntactic and semantic features from existing libraries. Ideally, we will train the baseline model first, and feed the additional features (punctuation marks, named entities, PoS tags, and sentiment scores) one at a time to explore their effects on model performance.

Among RNN implementations, LSTMs are advantageous because such architecture is easier to train and suffers less from vanishing or exploding gradients. The combination of CNN and RNN networks is also beneficial for several reasons. First, convolutional layers help to extract abstract and compact features to be used as inputs to LSTM network (Chan and Lane, 2015), which in turn compensates for the disadvantage of fixed filter width of convolutional layers, allowing for better analysis of long-distance relationships over texts with various length. Additionally, the convolution layers reduce feature variations so that CNN-LSTM work more efficiently (Ghosh and Veale, 2016).

To combine different components, we plan to feed the output of LSTM layers and the pre-extracted features into a fully connected layer to match the features into a more separable space. Another design is to first feed the extracted features into a fully connected layer, and then concatenate the output of this layer with the output of LSTMs as the input vector for the final fully connected layer. We will then use softmax for classification and compute cross-entropy as the loss function.

5.4 Evaluating Metrics

In the 1,044,839 training samples, there are 980,167 samples labeled as sincere while 64,672 of them are labeled as insincere. Therefore, $accuracy = \frac{the\ number\ of\ correctly-classified\ samples}{the\ number\ of\ all\ samples}$ is not

appropriate for evaluating the model. In this section, targeting at this biased dataset, different evaluation metrics will be discussed and applied.

There are two distinct values of *target* in our dataset: 0 indicates a sincere question while 1 indicates an insincere one. In this project, we define samples that are labeled as 1 to be positive while those labeled as 0 to be negative. Hence we could calculate our model's precision, recall, and F1 score accordingly. Provided that, these statistical metrics could be applied for evaluations.

6 Experimental Analysis

6.1 Corpus-statistical Insights

To investigate the nature of insincere questions from a statistical perspective, we will start our discussion by looking into the corpus statistics experiment results. Although the ground-truth labels contained noises, we refer to labeled questions as "insincere questions" or "positive samples" and unlabeled questions as "sincere questions" or "negative samples" in the following discussion.

6.1.1 Punctuation Marks

In addition to serving as grammatical structures and the guidance of pauses, punctuation marks convey information about emphasis on specific content and reflect people's psychological states (Thorndike, 1948). In accordance to our hypothesis, the usage of punctuation was significantly different between sincere questions and insincere questions.

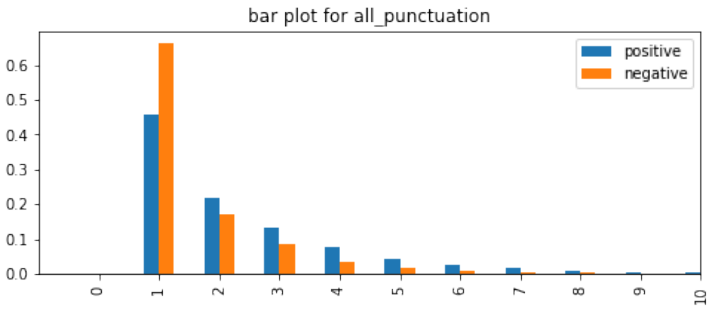


Figure 3: Bar plot of punctuation marks distribution in positive and negative samples

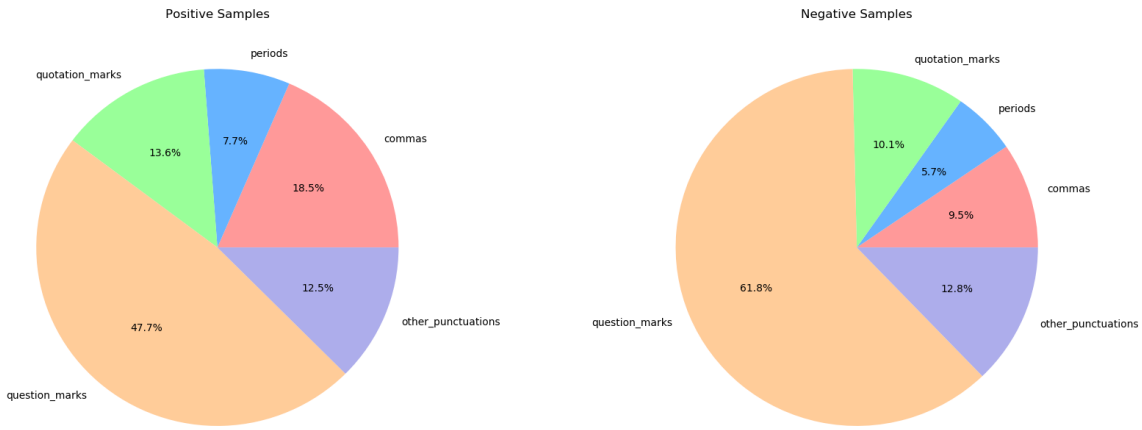


Figure 4: Pie charts for proportions of punctuations in positive and negative samples

Figure 3 indicated a general tendency for insincere questions to contain more than one punctuation marks and figure 4 showed the pattern of insincere questions displaying a more diverse distribution of types of punctuation marks. Combining the information of these two graphs, we conclude that sincere questions were more likely to contain question marks, which indicate an inquisitive tone. On the other hand, insincere questions had higher proportion of commas and periods, possibly because more pauses and descriptions were needed when making a statement compared to asking a questions. The distinctive patterns of punctuation usage were evidence that a feature vector containing punctuation marks counts was meaningful to the classification problem. To help visualize the feature vectors, we performed principal component analysis on 2000×2 random samples as shown in figure 5.

6.1.2 Parts of Speech Tags

Parts of speech counts are widely used as features in sentiment analysis tasks (Medhat et al., 2014) and have shown positive contribution to sarcasm detection (Barbosa and Feng, 2010). In our study, parts of speech tags clearly beared discriminative power.

Based on an examination of PoS tags distribution at both corpus level (see plots in Appendix A) and sentence level (see plots in Appendix B), we discovered the following patterns:

1. For most types of tags, including adjective and adverbs, each sincere question tended to

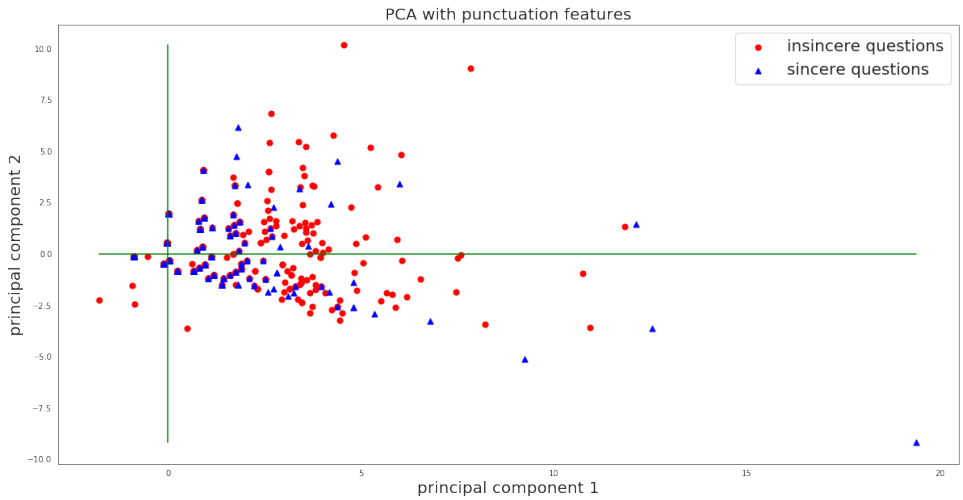


Figure 5: PCA of punctuation features

contain less than two occurrances, whereas insincere questions were more likely to show higher counts. In other words, PoS tags showed more compact distributions in negative samples.

2. The presence of adjective, adverbs, proper nouns seemed to be the strongest indicators of insincerity. This was expected because both adjectives and adverbs were sentiment-bearing, and proper nouns were suggestive of certain themes that frequently appeared in insincere questions.
3. Numbers seemed to be more associated with sincere questions possibly because of the relation of numbers to facts and math-related questions.

PCA of PoS-tags feature vectors is shown in figure 6. Consistent with our analysis above, the data points of sincere questions were much more concentrated than those of insincere quetions.

6.1.3 Named Entities

Compared to punctuation marks and PoS tags, the distributions of named entities were much more sparse. Therefore, the statistical plots of named entity count were less informative. However, we consider named entities as valuable information for the classification problem because named entities may associate with certain topics that are frequently mentioned in insincere questions. For example, "Trump" was the 9th most frequent unigram with 5212 occurrances in the 64671 labeled questions.

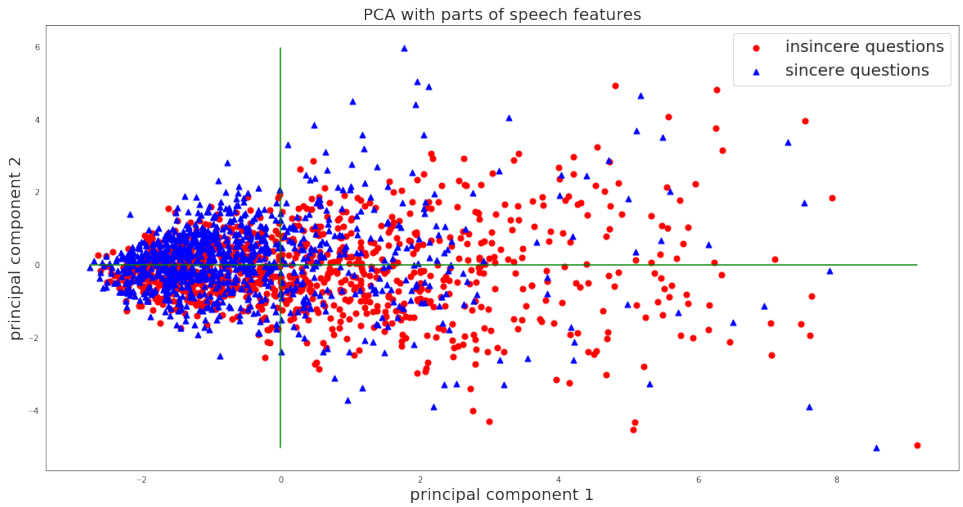


Figure 6: PCA of PoS features

Among all the named entity count that passed the KS test, NORP (nationalities or religious or political groups) counts showed the highest level of distinction between the two sample populations (figure 7). As we described in the introduction, one major type of insincere questions were questions that were discriminative against a group of people. NORP counts partially captured this information.

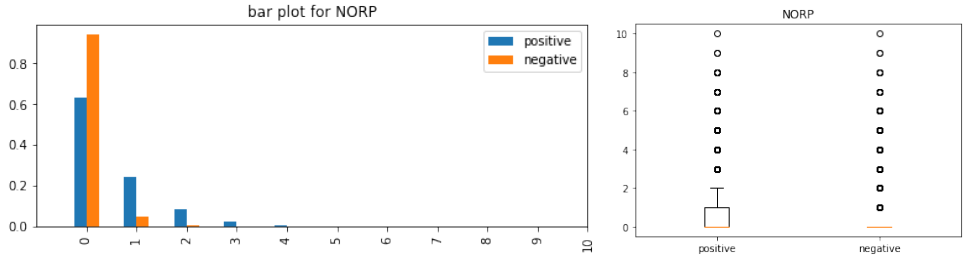


Figure 7: NORP distributions in positive and negative samples

Consider that the data was sparse, PCA of named entities features (figure 8) expectedly showed less distinguishable pattern compared to the previous two types of feature vectors.

6.1.4 Sentiment Scores

As we have discussed before, sentiment and polarity are important in the detection of insincere questions. Although we mostly relied on the neural networks models to extract semantical information, we still conducted statistical analysis on the sentiment scores to get a more staight-forward picture of how sincere and insincere questions differ in terms of sentiments. Expectedly,

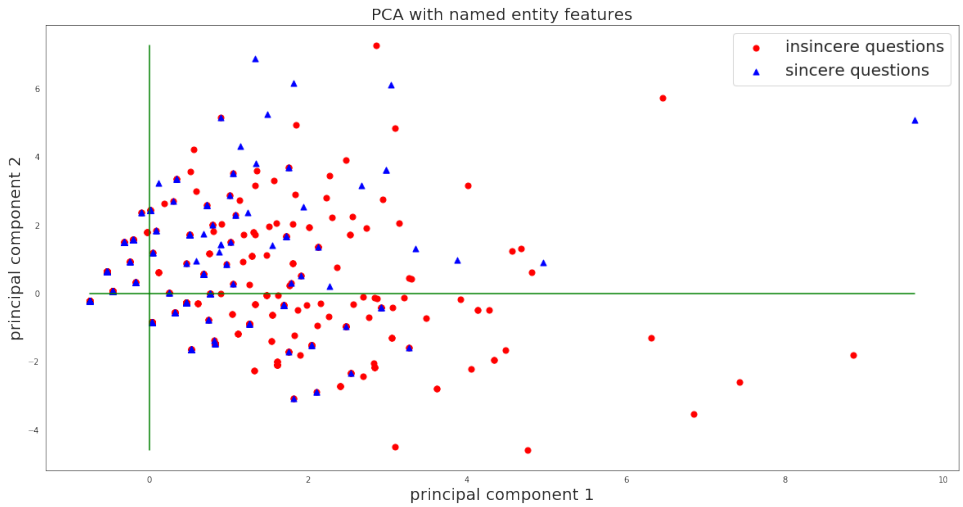
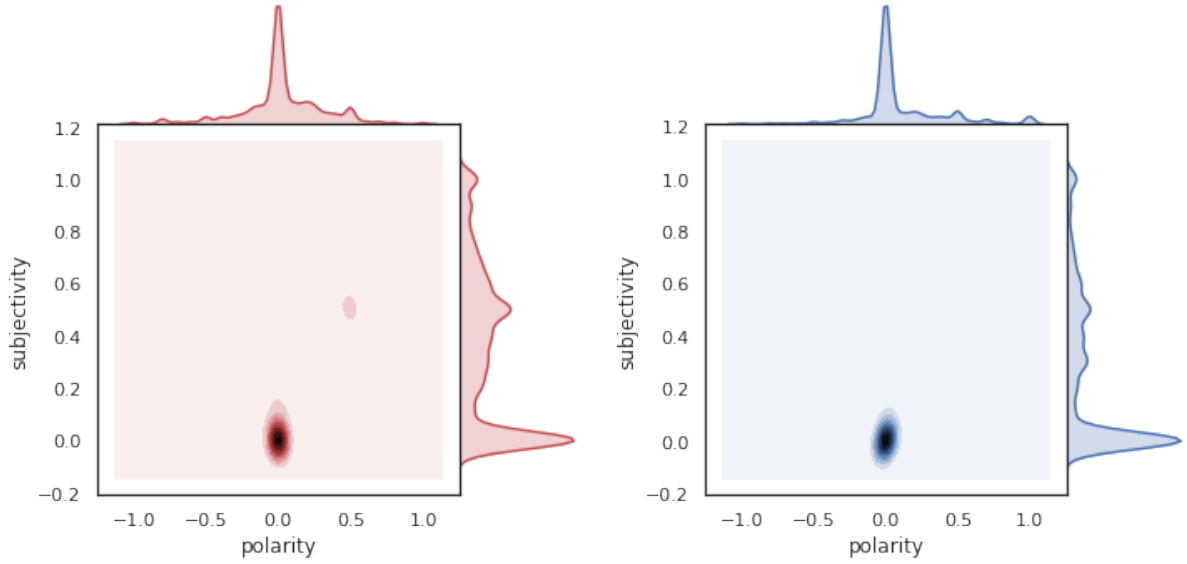


Figure 8: PCA of named entities features

both sentiment and polarity scores extracted from TextBlob library passed the KS test and were considered significantly different in our experiment setting.



The curves on the right and top side of each plot are the kernel density estimation. The darker region on the joint plot indicates higher density. The polarity score is a float within the range [-1.0, 1.0]. The subjectivity is a float within the range [0.0, 1.0] where 0.0 is very objective and 1.0 is very subjective.

Figure 9: KDE jointplots of sentiment scores for labeled questions (shown in red on the left) and unlabeled questions (shown in blue on the right)

Figure 9 shows the joint KDE distribution of polarity and subjectivity scores in labeled and unlabeled samples. We observed that the unlabeled samples were more concentrated around (0, 0) which indicates a complete neutral tone, whereas the labeled samples showed both higher chances of displaying non-neutral polarity and subjectivity. However, the two populations were

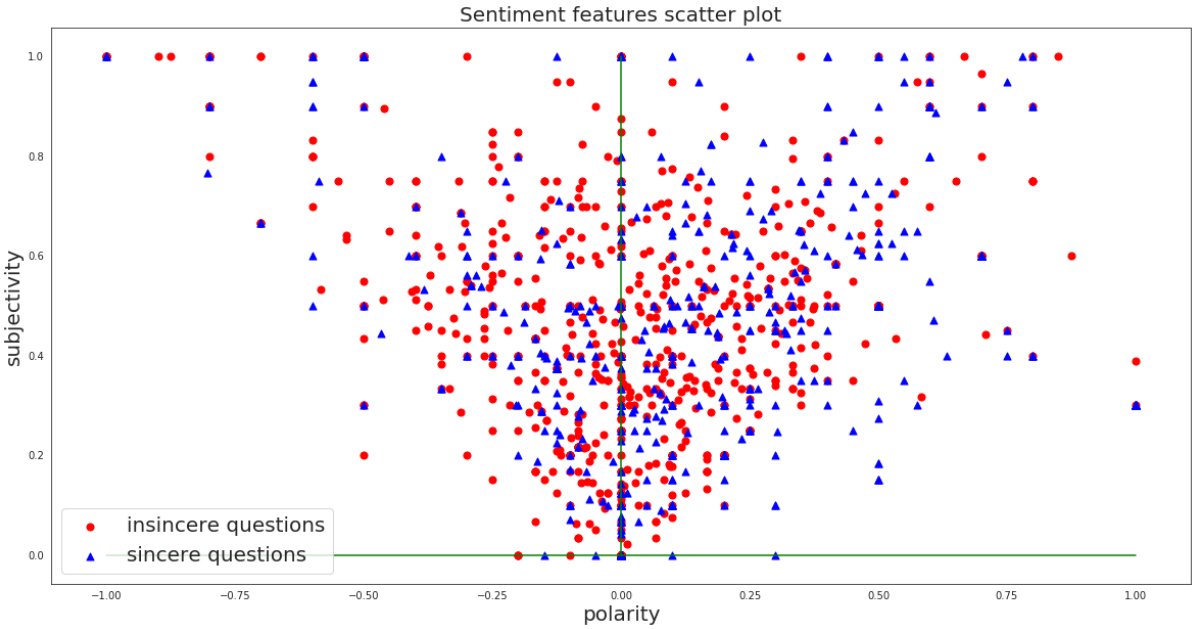


Figure 10: Scatter plot of sentiment scores

unexpectedly similar as shown in figure 10. We do not consider this statistical analysis results as contradiction to our hypothesis that sentiments were crucial in insincere question detection. Rather, we believe that it might be too simplistic, and therefore problematic, to represent the sentiment of a question using two scores assigned by an existing library.

6.2 Discussions

extrapolate our findings:

high-level insight:

We propose the following approaches that may have further improvements: 1. neural networks for sentiments score 2. larger size of model (LSTM hidden sizes, multiple CNN filters, multiple layers of DNN) 3. potentially use SVM at the very last step may have some influences

References

Universal pos tags, 2017. URL https://universaldependencies.org/u/pos/.

Annotation specifications schemes used for labels, tags and training data., 2019. URL https://spacy.io/api/annotation.

Luciano Barbosa and Junlan Feng. Robust sentiment detection on twitter from biased and noisy data. *Coline 2010: Poster Volume*, 2010.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

William Chan and Ian Lane. Deep convolutional neural networks for acoustic modeling in low resource languages. *In IEEE International Conferene on Acoustics, Speech and Signal Processing*, 2015.

Cicero Nogueira dos Santos and Maira Gatti. Deep convolutional neural networks for sentiment anlysis of short texts. *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 69–78, 2014.

Aniruddha Ghosh and Tony Veale. Fracking sarcasm using neural network. *Proceedings of NAACL-HLT*, 2016.

Anastasia Giachanou and Fabio Crestani. Like it or not: A survey of twitter sentiment analysis methods. *ACM Comput. Surv.*, 49(2):28:1–28:41, June 2016. ISSN 0360-0300. doi: 10.1145/2938640. URL http://doi.acm.org/10.1145/2938640.

Geoffrey E Hinton, James L McClelland, David E Rumelhart, et al. *Distributed representations*. Carnegie-Mellon University Pittsburgh, PA, 1984.

Geoffrey E Hinton et al. Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, volume 1, page 12. Amherst, MA, 1986.

Aditya Joshi and Pushpak Bhattacharyya. Automatic sarcasm detection: a survey. *ACM Computing Surveys*, 2017.

Aditya Joshi, Vaibhav Tripathi, Kevin Patel, Pushpak Bhattacharyya, and Mark Carman. Are word embedding-based features useful for sarcasm detection? *arXiv:1610.00883v1*, 2016.

Walaa Medhat, Ahmed Hassan, and Hoda Korashe. Sentiment analysis algorithms and applications: a survey. *Ain Shams Engineering Journal*, 2014.

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

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.

Soujanya Poria, Erik Cambria, Devamanyu Hazarike, and Prateek Vij. A deeper look into sarcastic tweets using deep convolutional neural networks. *arXiv:1610.08815v2*, 2017.

Jurgen Schmidt-Radefeldt. On so-called rhetorical questions. *Journal of Pragmatics*, 1977.

E. L. Thorndike. The psychology of punctuation. *The American Journal of Psychology*, 61(2): 222–228, 1948. ISSN 00029556. URL `http://www.jstor.org/stable/1416968`.

A Bar plots of PoS Distribution at Corpus Level

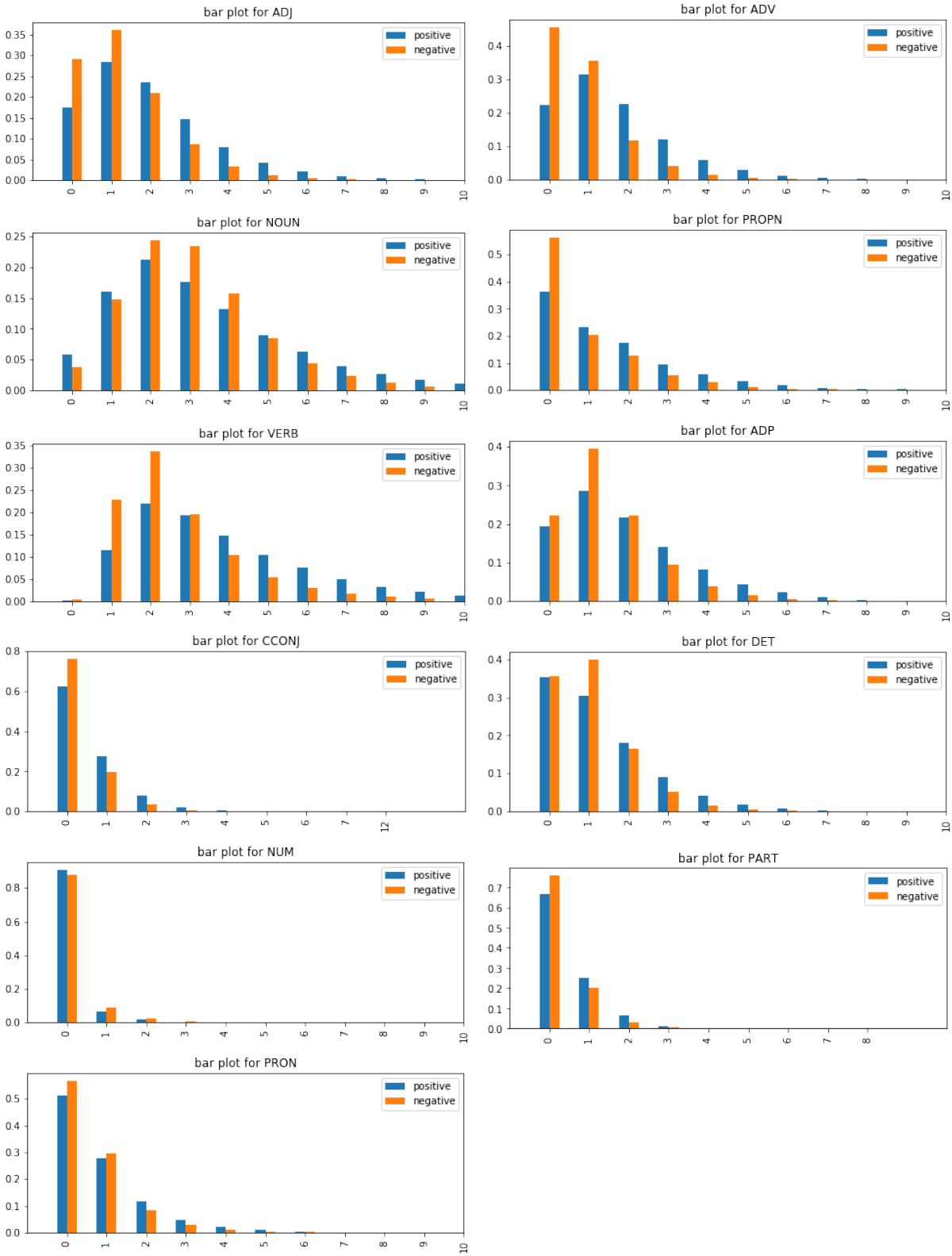


Figure 11: Bar plot of PoS distribution in positive and negative samples

B Boxplots of PoS Distribution at Sentence Level

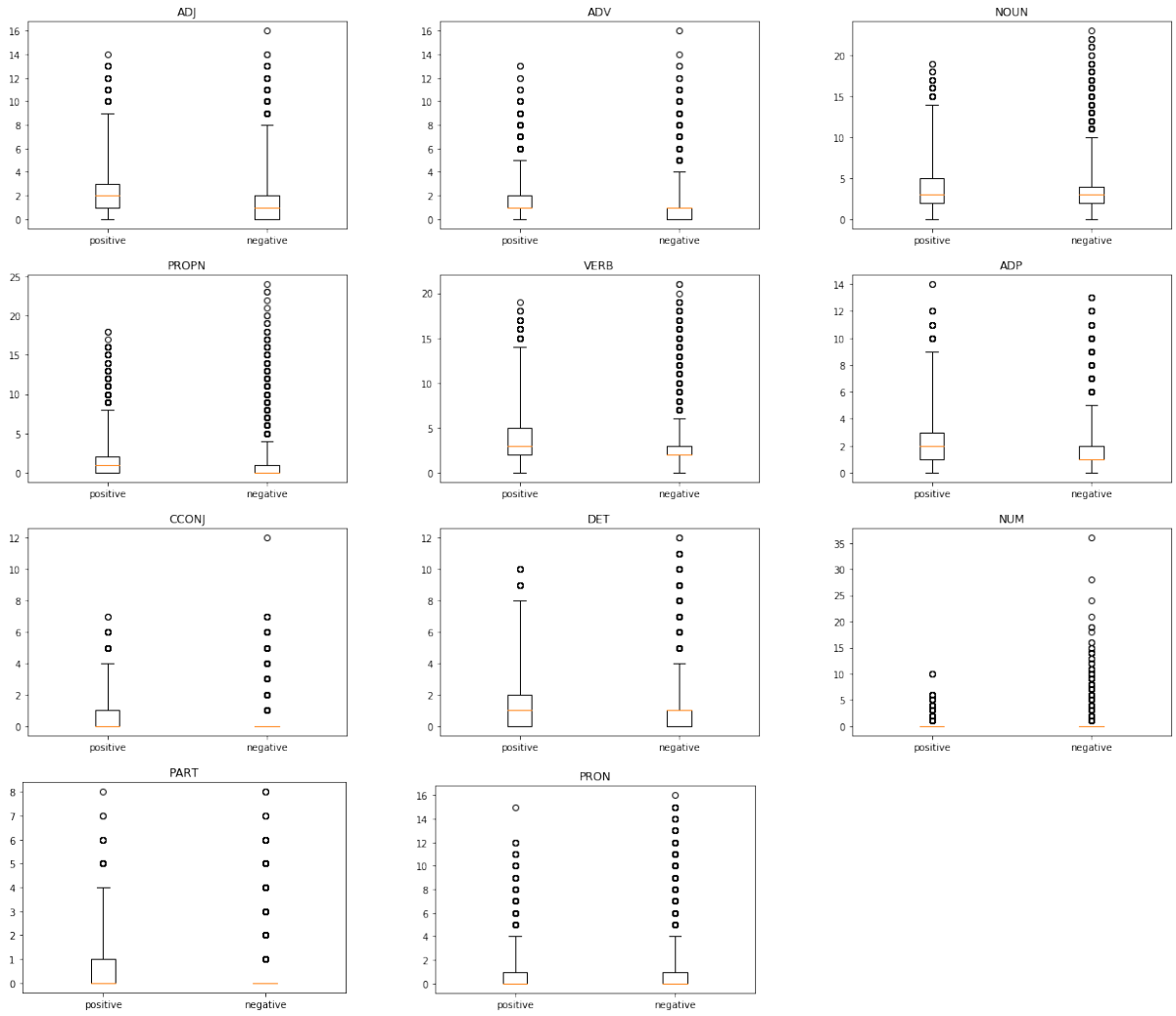


Figure 12: Boxplot of PoS distribution in positive and negative samples