Classification Of Ad Tone in Political Video Advertisements Under Class Imbalance and Low Data Samples

Mohit Baskota
Dept. of Computer Science
Iowa State University
Ames, United States
mbaskota@iastate.edu

***Abstract*—Ad tone defines the aim of a political video advertisement, which can be either to promote a specific candidate, to attack the candidates or to contrast the candidates. Depending upon the aim, a political video advertisement can be classified into either promote, attack or contrast class. Analysis of ad tone in political video advertisements can provide more insights about the political campaign to political science researchers. Political campaigns are investing more and more on online platforms, which creates a large amount of political video advertisements. Manual classification of ad tones in political video advertisement is time-consuming, labor intensive and not scalable. Hence, there is a need for an efficient and effective classification model for automatic classification of the ad tones in political video advertisements. The available labeled dataset is very small in size and suffers from class imbalance. Due to this reason, the performance of the minority class is poor compared to the majority class. Moreover, due to the way the different classes are defined, all three classes decompose into sub-parts and suffer from class overlapping problem. There has been an attempt in automatic classification of political ad tones, but it does not take class imbalance into account. We investigate a couple of data augmentation techniques to overcome the class imbalance problem and the effectiveness of deep learning models on ad tone classification using text-based features. In our experiments, the best deep learning model offers a better F1 score of 0.570 on the minority class compared to the F1 score of previous work, which is 0.527. However, the performance is still unsatisfactory. We design hand-crafted features specific for ad tone classification using Support Vector Machine as the classifier. Our proposed approach gives the best weighted average F1 score of 0.860 on the entire test set and F1 score of 0.657 on the minority contrast class.**

I. INTRODUCTION

Shifting from the traditional print and broadcast media, political campaigns are now spending heavily on online platforms to reach their voters. In presidential election 2012, campaigns spent around $94.4M on web advertisements compared to $8.5M on broadcast and $2.4M on print media [1]. With the help of micro-targeting, political campaigns can reach a specific group of audiences depending on demographic groups, interests, and several other criteria [2]. This is why, in the present days, there are more focus on online platforms for the advertisements than the traditional media. The expenditure on digital advertisements was significantly higher in the presidential

election campaign in 2016 compared to 2012 [3]. According to Google's transparency report, political campaigns have spent around $64M in advertisements on Google's platform since May 31, 2018, and around 18 percent of them are video advertisements [4]. The number of political video advertisements in platforms like YouTube is increasing and automatic analysis of the contents of the political video advertisements is of great importance to political science researchers.

Analysis of the contents in the political video advertisement was started by Wesleyan Media Project (WMP) [5]. It covers all the broadcast television political advertisements from the year 2010. The code-book, WMP - 2010 [6] is a document developed by political science researchers. It contains an extensive set of questions about a political video advertisement such as the **tone of the ad**, sponsors of the ad, issues mentioned in the ad, appearance of the American flag in the ad, and several others [7]. With manual coding, the researchers successfully answer all the questions from the codebook. However, manual coding is tremendously time consuming. With the significant increase in political video advertisements in online platforms, their method of manual coding is not scalable. Therefore, coding of online video advertisements needs to be automated.

Among the several questions from the code-book, we focus on answering the following question using an automated approach. **"In your judgment, is the primary purpose of the ad is to promote a specific candidate, attack a candidate, or contrast the candidates?"** Given a political video advertisement, our aim is to classify the tone of the ad as either promote, contrast or an attack.

While the codebook provides the question, it fails to give the definition of each ad tone. Before proceeding to classify the ads to their respective ad tone category, we first seek to get the definitions of all the ad tone categories. The work in [8], with consultation of political science researchers, gave the concrete definition of all the ad tone categories. Following is their definition of all the ad tone categories and we follow the same definition for our work.

Attack ad is an ad that contains only attacks against others such as one of the following cases.

1. Candidate A attacks candidate B in the same party.
2. An organization attacks a candidate.
3. An organization attacks a party.

Promote ad is an ad that fits one of the following criteria.

1. Candidate A promotes himself/herself than others.
2. Candidate A attacks candidate B in another party.
3. Candidate A attacks another party.

Contrast ad is an ad that promotes one candidate while attacks another candidate in the same party.

To the best of our knowledge, there are no other datasets available for online political video ads with labeled ad tones apart from the one created in [8]. The corpus consists of 773 online political video ads divided into 498 promote ads, 191 attack ads, and 84 contrast ads. The dataset is very small in size and has class imbalance. The class imbalance occurs when data samples in the dataset are not uniformly distributed across different classes. The class with more data samples compared

to others is called the majority class and the ones with fewer data samples are called the minority class. As most of the classification models are dependent on data statistics, they may be biased towards the majority class and have very poor performance on the minority classes. In our dataset, the promote class is the majority class. The attack and contrast classes are the minority classes. Even if we collect many more ads and label them manually, class imbalance still exists because the election campaigns tend to focus more on promote and attack ads compared to contrast ads.

Classification of ad tones in the online political video advertisement is challenging not only due to the limited availability of labeled data and class imbalance problem. If we revisit the definition of all the ad tones, we can observe that it suffers from class decomposition and overlapping problem [9]. The classes are decomposed into small sub-parts such that some sub-part overlaps between the classes. For example, in the definition of promote class, an ad belongs to the promote class even if it attacks a candidate from another party. The only subtle difference between this kind of promote ad and attack ad is that attack ad involves attacking candidates in the same party. This makes the classification of political ad tones harder to solve if relying only on word patterns used.

To address class imbalance, we investigated two oversampling methods for data augmentation. The first method uses a Recurrent Neural Network (RNN) [10] based model to generate synthetic text from the contrast class. The second method uses language translation to translate text from speech in ads from the contrast class to an intermediate language and translated it back to the base language, creating new samples in the minority class.

For ad tone classification, we investigated deep learning language models. We propose a neural network model which has better performance on the minority contrast class with F1 score of 0.570. It is approximately 5 percent higher than the F1 score of the current classification model proposed by [8]. We trained for word embeddings [11] using our dataset and also implemented pre-trained word2vec [12][13] with the aim to overcome the class imbalance problem. However, both the models over fit the training set and did not perform well.

We designed hand-crafted features and used Support Vector Machine (SVM) to classify the ad tones. To address class imbalance, we oversample the minority classes using SMOTE [17]. This model performs best among all the models with an average F1 score of 0.860 on the entire test set and 0.657 on the contrast class.

The paper is organized as follows. Section II provides a brief summary of the related work. In Section III describes our proposed approach. Experimental design, results and analysis are described in Section IV. The conclusion and the description of the future work are described in Section V.

II. Related work

Classification of ad tones in political ads: As per our best knowledge, the first and only prior work related to classification of political ad tones is described in “Automated coding of political video advertisements” [8]. This work gives the definitions of all three classes of ad tones in political video ads. Multiple machine learning models (e.g., SVM, K-nearest neighbors, Naive Bayes) were applied to textual features extracted from audio and Optical Character Recognition (OCR) from images. Good performance on the entire dataset was achieved, but the performance is low on the contrast class with the best F1 score of 0.527. The low performance on the contrast

class is due to limited number of training samples and class imbalance. The authors of [8] did not investigate any techniques to tackle the class imbalance problems.

Classification under class imbalance: The techniques that deal with imbalanced data classification fall into two categories – algorithmic [14] and data manipulation techniques [14]. The algorithmic approach focuses on enhancing and designing the algorithm to handle the imbalance in the data naturally. Data manipulation techniques handle the imbalance in the data by modifying the distribution of the data to create a balance in the training set before any machine learning algorithm is applied. Oversampling of the minority classes and under sampling of the majority classes are approaches [15] and [16]. Synthetic Minority Over-sampling Technique (SMOTE) [17] is a very popular technique that creates a balanced data set using random oversampling of the minority class. In this technique, we select the K-neighbors of a sample from minority class. For each neighbor the difference between its feature vectors and minority data sample is calculated. The difference is then multiplied by some random number between 0 and 1 and then the result is added to the original feature vector. This causes the selection of K random point between the selected sample and its K-neighbors. This approach will help to create a balanced dataset.

Data augmentation in the text: Data augmentation has been used widely in the area of vision to deal with the class imbalance problem [18] [19]. However, there has been very limited use of data augmentation techniques in natural language processing (NLP). One of the techniques of data augmentation in NLP is using synonym replacement [20] [21] [22]. However, in recent years RNN [10] and Generative Adversarial Network (GAN) [23] have been used to generate good quality synthetic texts [24]. Another simple technique of data augmentation can be language translation which has been proven to be helpful to achieve better performance in the recent Kaggle challenge for Toxic Comment classification [25]. They translated English documents to another intermediate language and translated them back to English to augment the training set. While translation helped them to achieve better performance, it is not always the case and depends largely at the problem being solved.

Video classification using text-based features: The idea of using text-based features for video classification has been extensively used [26][27]. The advantage of using text-based features is that we can utilize a large number of researches that have been conducted in document classification for the video classification problem. The textual features of the video can be extracted from the texts that appear in the key-frames of the video using OCR techniques [28] [27]transcribing the speech in the video [29][30][31] or other human annotated data like tags, title, description and more. The work in [32] described a video characterization technique using image and language understanding. They used speech recognition to obtain the transcript and use Term frequency Inverse document frequency (TF-IDF) to measure the relative importance of words for the text extracted from the videos. Work in [8] used text features obtained from both OCR and speech recognition to classify the ad tone in political video advertisements.

Language modeling using deep neural networks: Following the impressive advances in computer vision, recent research in NLP are increasingly focusing on deep learning methods [33]. Traditional machine learning models in NLP depend heavily on handcrafted features while deep learning based models enable automatic feature representation. Authors in [34] showed that a simple deep learning model can outperform most state-of-the-art techniques in NLP tasks such as

Named-Entity Recognition (NER). Models based on neural networks have been performing better in various NLP tasks following the success of word embeddings [35] [11] and deep learning methods [36]. Word embeddings or distributional vectors capture the similarity between the words. They are pre-trained on a large unlabeled dataset by predicting a word based on its context [11][13]such that the word vectors can capture contextual information. They have been very efficient in capturing contextual similarity among the words. They are generally used as the first layer in a deep learning model. Word embeddings can be randomly initialized and trained on a given dataset or can pre-trained on a large unlabeled dataset. The fact that they can be pre-trained on a very large dataset and can be transferred to another problem is sometimes found to improve the performance when the size of the labeled dataset is small [37]. Google's word2vec [12] was trained on 100 million words from Google news dataset and has a vocabulary of 3 million words with embedding dimension of 300 features.

III. PROPOSED WORK

Supervised machine learning requires sufficiently large and balanced dataset to achieve good performance. Our dataset consists of 773 videos and has a class imbalance ratio for the contrast, promote, and attack class of approximately 1:2:6. The small labeled dataset and the imbalance ratio greatly reduce the overall classification performance. Moreover, due to the class imbalance, the classifier favors more towards the majority class, resulting in very low performance for the minority class. To overcome this problem, we investigate a couple data augmentation techniques. We then extract text-based features from the videos and investigate various existing language models to generate additional text features for classification. Finally, we propose hand-crafted features for ad tone classification. After feature extraction, we oversample features of both the minority classes using SMOTE [17], creating a balanced training dataset for supervised machine learning classifiers. We describe the process of text extraction from the videos, the data augmentation techniques, the feature extraction process, and various classification models.

A. Text extraction from the video

The two primary sources of the text in political video ads are the key-frames and the audio. We extract the text from both audio and key-frames and use it as the feature.

1. Audio text extraction

   We use speech recognition software [38] to create a transcript from the audio extracted from the video. Firstly, we extract audio from the video ad. Then we use Google's speech to text API [38] to extract the text from the audio source. Google speech to text API only supports audio shorter than 50 seconds. When the video is longer than 50 seconds, we split the generated audio into several parts of about 50 seconds each and send each part to Google speech to text API for text extraction. The extracted text from all the parts is then concatenated to create a single text document corresponding to the video ad.

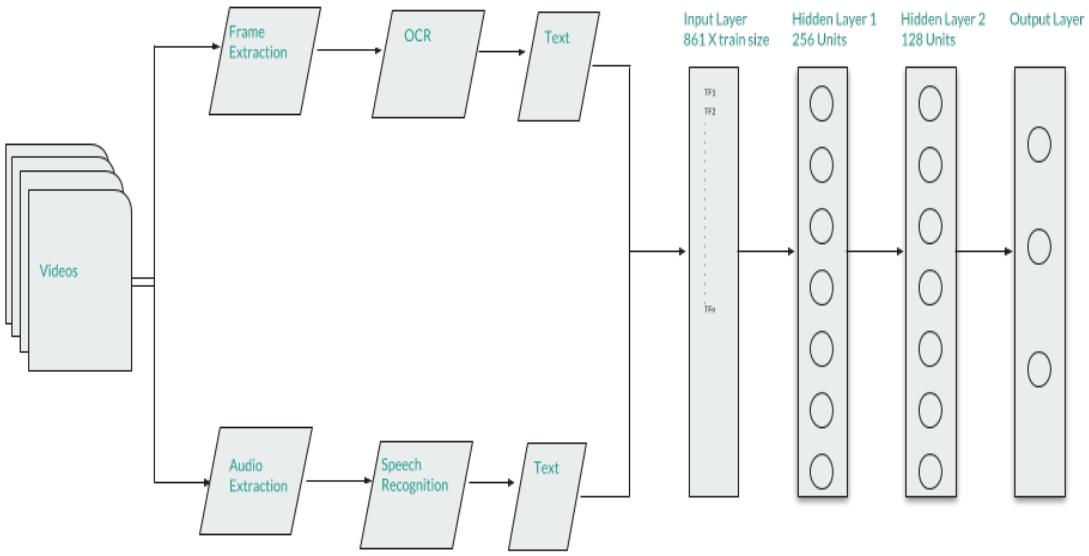

Figure 1: Overview of end-to-end process for model 1.

2. OCR text extraction from key-frames

   OCR text extraction from the frames generated from the video is more complex. One of the major problems is that we can generate frames from a video at various rates. This can cause two problems. 1) With a high frame extraction rate, we will end up with many frames with similar texts on them. As a result, it will be very time-consuming and will incur a monetary cost associated with OCR if using a paid third-party service like Google Vision. We will also end up with many duplicate texts. 2) If we generate frames with a smaller rate, we might lose some frames with the texts in it. To address these situations, we use an open source python library PySceneDetect [39] for scene detection. With scene detection, we limit the number of frames generated and select the frames that are important. We use Content detector mechanism from PySceneDetect with the threshold of 13, which separates scenes using changes in color and intensity between neighboring frames. After scene detection, we take first, middle and last frames of each scene as key frames and use Google Vision API [40] to get the text from the frames. We follow the same process for all the detected scenes. Finally, after extracting texts from all the detected scenes, we take the union of the sentences to create an OCR text document for the video to remove duplicate sentences.

B. Data Augmentation Techniques

We investigate two techniques for augmentation to increase the number of samples in the dataset to create a balanced dataset.

1. Synthetic text generation

   The word level recurrent neural network architecture in [41] is inspired by DeepMoji [42]. The first layer of this architecture is a 100-dimensional word embedding. This embedding layer is followed by LSTM layers. The outputs of all the layers are then fed into the

attention layer which weighs the most important temporal features. The embedding layer and the LSTM layers are skip connected into the attention layer, which helps to ease back propagation by preventing vanishing gradients. The output will be the probabilities of the candidate words for the next word.

We trained the networks with different configurations by varying the number of LSTM layers from 1 to 4 with 128 and 256 cell size. The training data comprises of the texts extracted from the audio of contrast class videos as explained previously. We kept the maximum length of the generated text to be 200 as our dataset does not have documents that have more than 200 words. We fixed the embedding dimension as 100 and trained both bidirectional and unidirectional RNN.

2. Language translation

   Motivated by the improvement in the performance by using language translation in toxic comment classification task [25], we created new data samples for the contrast class using Google Translate API [43]. We translated the text from English to first in French then did the second translation of the translated French text to Spanish. Finally, we translated the Spanish text back to English to generate a new sample for the contrast class.

C. Feature Extraction

All our classification models use text-based features. We describe the process of feature extraction, extracting text features from the video. We describe our proposed feature engineering method to create domain-specific handcrafted features, which is specific to our problem.

1. Bag of words

   Bag of words is a simple approach of representing text data for the machine learning algorithm. It is called a bag of words because, in this model, a document is represented with the words it contains, disregarding the grammar and the order of the words. We combine the text we extracted from audio and OCR, remove the stop words from it and convert all the words into the lower case. Then we extract word level n-grams from the whole corpus and create a vocabulary. Now from each ad, we calculate the TF-IDF for all the word level n-grams in the document to create a feature vector. So, for a given n-gram word from the vocabulary in a given document, its index in the feature vector consists of its TF-IDF value. We investigate a total of 3 different word level n-grams: 1-gram, 2-gram, and 3-gram.

2. Encoded text sequence

   Machine learning requires a numeric representation of the text and cannot work directly on the text data. So, we create the encoding of the text sequence of the document to create a feature vector. We first, combine both the audio and OCR texts extracted in the previous step from the whole dataset and remove stop words. Tokenization is one of the important steps in natural language processing. It splits a document into a list of words. We use Keras Tokenizer API to extract the tokens [44]. We limit the number of words in the vocabulary to be 2000 such that the less frequent words are not included in the vocabulary. The

vocabulary consists of only the top 2,000 frequent words. After tokenization, we will get the index for each word in the vocabulary. The document length of each ad is variable, and we cannot have a variable length feature vector. Since no document in our corpus has the total number of words greater than 200, we fix the length of the feature vector to be 200. Now for each ad, we concatenate the extracted text from the audio and the OCR creating a document that represents an ad. Then we use Keras texts_to_sequences API [44] to encode each word in the document with its index in the vocabulary creating an encoded text sequence. We finally pad the sequence with leading zeros to create the fix-sized feature vector.

3. Feature engineering

The definitions of the three classes have very subtle but important difference. To differentiate among the classes depends on whether the candidates/organizations belong to the same party. By looking at just the words and the sentiment of an ad, one cannot simply tell whether it is an attack, promote or contrast ad. This is because there is a decomposition of the class into subparts and there is also a class overlapping problem. For example, a promote ad does not only promote a candidate. If the ad attacks a candidate from another party, it is still considered as a promote ad. It is differentiated from the attack ad by just this one simple difference. An attack ad involves attacking a candidate in the same party while a promote ad involves attacking a candidate from a different party. For the contrast ad, there needs to be the case that the candidate being promoted, and the candidate being attacked both are in the same party. So, the contrast class overlaps between the promote class and the attack class.

The information about the political parties involved, the number of candidates that are being attacked, and whether the candidates belong to the same party or the different party are very important features. We perform Named Entity Recognition (NER) on the text extracted from audio and frames using Google Natural Language API [45] to extract the names of the persons. As the API is not accurate, we do additional text matching to find the names of the candidates. We create a JSON configuration file that has all the candidate names in the dataset mapped to their respective political party. This way we can extract the name of the candidates mentioned in the ad with their corresponding political party.

Extracting the name of the candidate being attacked and the one being promoted in the ad is non-trivial. In each political ad, the sponsor of the ad must be provided. The ad sponsor is very important to find out the candidates that are being attacked or promoted in the ad. Each sponsor normally supports and attacks a list of the candidates. Although in a given ad, a sponsor tends to support one of the candidates and attack one or many. We use a political website [46] to extract the information about the sponsors. This website has very good information about the sponsors including the candidates they promoted and attacked in an election with the expenditure. We create a JSON configuration file of all the sponsors in the 2016 presidential election together with the names of the candidates they promoted and attacked. We then perform simple text matching in the extracted texts to identify the sponsor. Once the sponsor is found, we then take the name of the candidate identified previously and match the configuration file of the corresponding sponsor to find out who are attacked or promoted. We finally create a feature vector with the following feature-set:

total number parties detected, count of the Republican candidates, count of the Democrat candidates, count of the candidates that are attacked, count of the candidates that are promoted.

D. Classification Models

We investigate the classification models that were not considered in the previous work [8]. We first focus on the classification models based on neural networks. We investigate a simple feed-forward neural network and then further expand our model to use the word embedding trained on our dataset. We then extend the model to use the pre-trained word embedding rather than the word embedding trained on our dataset. Finally, we experiment with SVM with our proposed features and feature level data augmentation.

1. Model 1: Fully connected neural network

   We use the bag of words TF-IDF features of word n-grams described in the previous section. As we have three classes, the output layer has three units. We use softmax as the activation function to get the prediction probabilities of the three classes. The first hidden layer connected to the input layer has 256 hidden units with Relu as the activation function. We use dropout for regularization. Our second layer has 128 hidden units with Relu as the activation function and is connected to the output layer. Figure 1 gives the overview of end-to-end process for model 1.

2. Model 2: Fully connected neural network with word embedding

   Our hypothesis is that the classifier will have more contextual information to learn from the data with the help of word embedding as compared to a simple bag of words model. We train word embedding on our dataset using word2vec with the skip gram model with a window size of 5 and a feature dimension of 50. We use the same model described above but the input layer, in this case, is connected to our learned embedding layer and the embedding layer is flattened and connected to the first hidden layer. We use the embedding as the feature vector instead of a bag of words for this model. The rest of the network is the same as that of Model 1.

3. Model 3: Fully connected neural network with pre-trained word embedding

   We also expand our embedding model to use pre-trained word embedding. Our hypothesis is that due to low data samples, the word embedding trained with our dataset is not able to learn as much contextual information required for the classifier. So, we use pre-trained word2vec embedding, which was trained on 100 billion words from a Google News dataset. The dataset has 3 million words and phrases and the length of the embedding vector is 300. The network structure is the same as that of Model 2 except that we replace the input embedding layer with the pre-trained word2vec embedding.

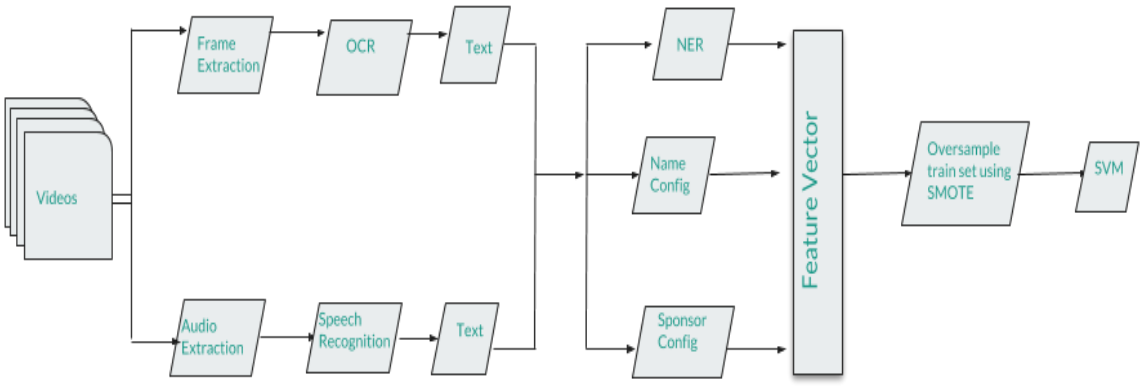

Figure 2 End-to-end process for model 4.

4. Model 4: SVM with hand crafted feature and oversampling

   We generate the feature vector as described in the feature engineering portion of the previous section. We use SMOTE [17] as the oversampling technique to overcome the problem of class imbalance. With the generated feature vectors for all the ads, we use a library in python for SMOTE to create synthetic samples of the minority classes – attack and contrast and train the expanded training dataset with SVM. We created 177 and 249 synthetic samples in attack and contrast class respectively creating a balanced training set with 295 samples in each class.

IV. Experiments and Analysis

A. *Dataset and Ground Truth*

1. Dataset

   Per our knowledge, there is no other publicly available labeled dataset apart from the one created by [8]. We take the ground truth from the available dataset and download the video from the actual source [47][48]. The ground truth for the videos are given by the definitions as described in Section I. The dataset contains 773 ads in total with 498 in promote class, 191 in attack class and 84 in contrast class. Figure 3 shows the distribution of ads in the dataset.

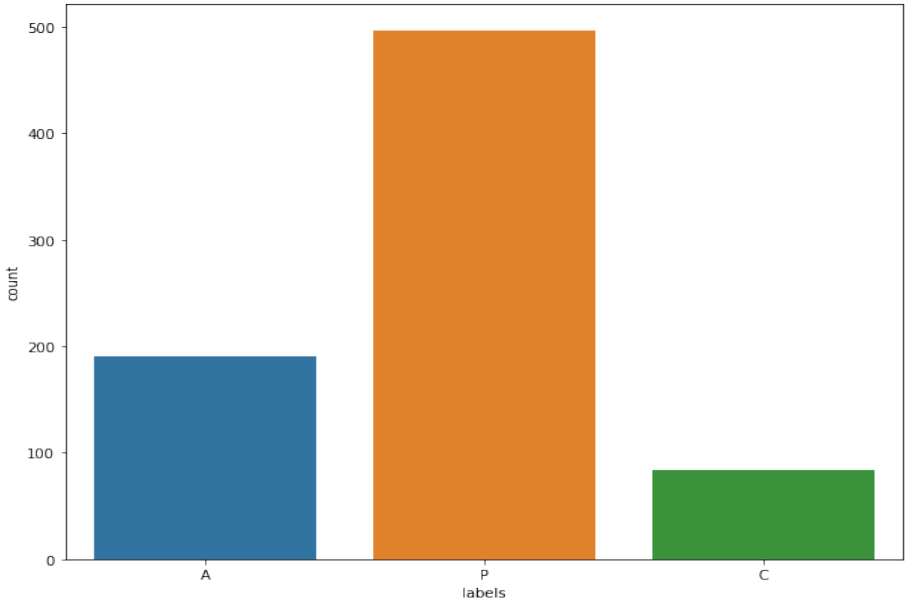

Figure 3: Distribution of ads over different classes. P: Promote class, A: Attack class, C: Contrast class

2. Training and Test Sets

   For the first three neural network models, that are based on neural network, we split the dataset with 80% of the data in training set and remaining 20% in the test set, keeping the original ratio of three classes in each set. For the last method that uses SVM and handcrafted features, we split the dataset such that 60% of the data is in training set and 40% of the data in the test set keeping the original ratio of three classes in each set. We increase the test size in this case because we will perform oversampling of minority class in the training set. That way we will have an increased number of samples in both the test and training set.

3. Parameter Tuning

   We experimented with various hyper-parameters for the first three neural network models. We changed the number of hidden layers from 1 to 4 with varying number of hidden units: 64, 128, 256 and 512.

   We used the grid search algorithm to find the best parameters for our last model that is based on SVM. We select the parameter C = [0.1, 1, 10, 100, 1000], Kernel = ['linear', 'poly', 'rbf', 'sigmoid'], Gamma = [0.001, 0.01, 0.1, 1.0, 10,], Decision function shape = ['one vs one', ‘one vs rest’] for the grid search.

4. Performance Metrics

   To compare our result with the previous work, we follow the same performance metrics that are used by the previous work. We use the weighted Precision (P), recall (R), and F1 score to measure the performance of all the models. The three metrics are defined below

where N_i is the number of ads in ith class, P_i, R_i, $F1_i$ is the precision, recall and F1 score for ith class, respectively.

$$P = \frac{\sum_{i=1}^{3} N_i * P_i}{\sum_{i=1}^{3} N_i}; R = \frac{\sum_{i=1}^{3} N_i * R_i}{\sum_{i=1}^{3} N_i}; F1 = \frac{\sum_{i=1}^{3} N_i * F1_i}{\sum_{i=1}^{3} N_i}$$

B. *Classifier Training*

The three neural network-based models were all trained for 100 epochs with a batch size of 10. We used categorical cross-entropy as the loss function and stochastic gradient descent algorithm for the optimizer.

For our proposed model 4, we used the best parameter found from the grid search. We used 'rbf' kernel with C=1000 and gamma = 0.001 to train the model.

We trained all the models using 10–fold cross-validation in the training set. The training set was first divided into 10 sets of equal size. The model was then trained on the 9 of those sets and the remaining one was used for the validation. Then we take the performance of the classifier in the test set. We repeat this same process for 10 times and report the average performance on the test sets.

C. Experiment Results and Analysis

1. Data Augmentation Techniques

1. Synthetic text generation

   The best result from the synthetic text generation is shown in Table 1. As we can see that the generated text does not have good quality. The text generated is very short and it does not convey any meaning. The state-of-the-art text generation is usually performed on large training dataset with millions of documents. Our dataset has only 84 documents and the generator was not able to learn on such small dataset and generate good quality texts. As the text generated from this method is not good, we do not use it as our data augmentation technique.

2. Language translation

   We applied the aforementioned translation method to the data in the contrast class. The examples of original text and the translated text are shown in Table 2. The final text has some new synonym words and change in the sentence structure. We translated all the samples in the contrast class, doubling the size of the contrast class. We added the new samples to the training set and trained the best performing SVM classifier from the previous work. We do not get any noticeable improvement in the performance. The performance remained approximately the same. This is because there was not a substantial increase in the size of the contrast class. Moreover, the newly generated samples did not differ much in regards to the words in the original text. The classification model uses a bag of words as the feature and change in the sentence structure is not that important to it.

is is and rubio the and watched ted cruz country cruz cruz cruz cruz country now ' s m the , ,
i ' s s fo r a do to liners to the the an be the the reagan this a of clinton the members of the the the into to we to you on his that will to the bill military the would that in made trump in and clinton want security trump trump trump it ' under and to to it fight it to ' s for the i this this message ,
and is for the jim colored and roadside cruz is and for the to resolution and are in ted trump cruz one the the the that to and utmost line and trump the a time of and amnesty author laborers that vote the to marco rubio politicians trump trump rubio trump have a values the ' s t rules it ' m s s and to one here ' s m president ted cruz cruz ' s conservatives our and conservatives cruz cruz cruz cruz
' s i let cruz cruz cruz
- cruz cruz cruz cruz john
ted cruz ' taxes

Table 1: Sample text generated from the contrast class using synthetic text generation method

Original Text	Translated Text
justice scalia's death us the next president just who donald trump pick person from suggested the alabama supreme court ban everything is on the line now our freedoms america the battleground do you trust you trust to fill our federal courts values for ted cruz,cruz for president let's take our country back,	the death of Scalia of Justice, the next president, of which Donald Trump has chosen the person of the supreme court of Alabama, everything is in danger now, our freedoms, the United States, the battlefield on which you trust to fulfill the values of our federal courts for ted cross, cross for the president. recover our country,

got a republican feel take the first step monday evening for a major political party party is doing carly fiorina a conservative outside who has a bold new blueprint to take our country back carly fiorina a conservative leader we can trust to take our country back before barley for america,	I have the feeling that the Republicans are taking the first step on Monday night, because an important political party is Carly Fiorina, an outside conservative, with a bold new plan to get our country back. America

Table 2: Original text from contrast class and the text obtained after translation.

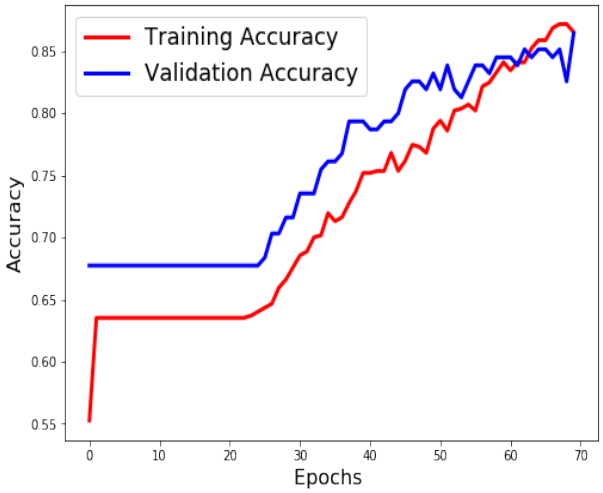

Figure 4: Training and validation accuracy across all epochs for Model 1

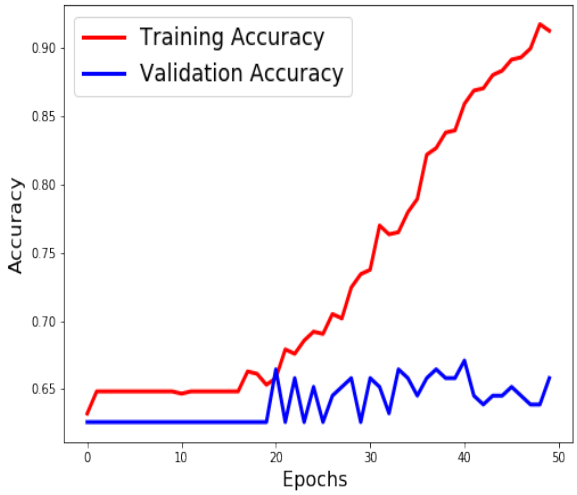

Figure 5: Training and validation accuracy across all epochs for Model 2

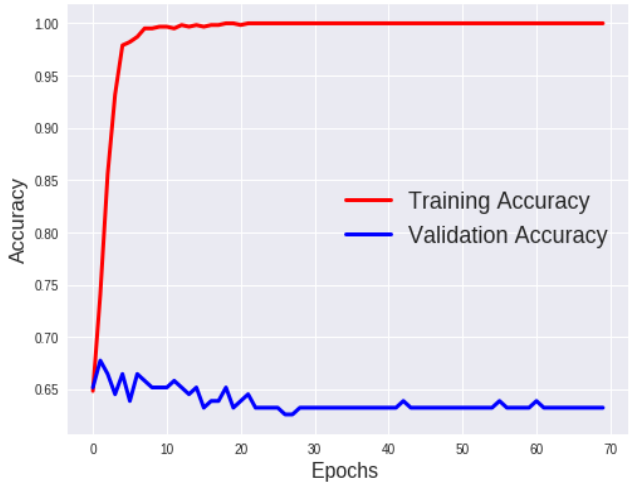

Figure 6: Training and validation accuracy across all epochs for Model 3

2. *Classification Models*

1. Neural network-based models (Model 1, Model 2, Model 3)

   Table 3 shows the result of the model 1 and compares it with model 4 and the best models from [8]. The F1 score of model 1 on the entire set is 0.846 which is a slight improvement over Rule-based method and SVM (1,2)- W. However, the F1 score on the contrast class was improved approximately by 5 percent compared to SVM based models. It was not able to improve the F1 score on the contrast class compared to Rule-based method. Figure 4 shows the training and validation accuracy for model 1. We can see that it converges at epoch 70. If we go beyond that the difference between training and validation start to grow bigger and the model will over-fit on the training data. We report the performance of model 1 at epoch 70. This model does perform better compared to SVM based models but not with a huge difference. The reason is that the distribution of the words among the classes are very similar as shown in Figure 5 and with have very low data samples and imbalanced dataset.

   We do not report the performance of model 2 and model 3 because they did not perform well compared to any other models. Model 2 converges at epoch 20 with very low performance and starts overfitting on the training data rapidly as shown in Figure 5. Model 3 performs worse as it starts overfitting on the training data right away as shown in Figure 6. The reason for the bad performance of the word embedding model is because the audio text we extracted from political ads is not syntactical as a normal audio conversation. The ad has various segments with each segment talking about different topics. Therefore, the word embedding trained on our dataset performed better than the pre-trained word2vec because it was able to learn the contextual similarity of the words in our dataset but pre-trained was not able to. So, the major problem in this case also is the low number of data samples to train the word embedding.

2. Support vector machine with hand crafted feature and oversampling (Model 4)

   Our proposed model 4 achieves the best performance compared to all the other models as shown in table 3. It achieves the best F1 score of 0.860 on the entire set. It achieves best F1 score on promote and contrast but the attack class. While the performance of the attack class drops by 3 percentage as compared to SVM 5- ch, our model does better on the contrast class by 12 percentage compared to SVM based models. While there is an improvement in performance, we do not improve greatly. This is because of the scenarios like when we detect candidates that are being attacked and promoted in the ad, a lot of times when an ad is attacking a candidate, there is a section at the end mentioning the candidate who approves it. In that case, the ad will detect both promoted and attacked candidates, but the ad is attacking the candidate. Table 4 shows the classification result on test set. The rows with red indicate that the actual class label and predicted class labels are different.

   We performed better than the Rule-Based method but not very significantly. However, the advantage of our method over the Rule-Based is that we do not have to hard code all the rules. We just have to add the name of the candidates and the sponsor in the configuration.

Models	Entire set			Promote			Attack			Contrast		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
Model 4 (SVM with our handcrafted features)	0.874	0.850	**0.860**	0.941	0.895	**0.917**	0.824	0.786	0.804	0.596	0.733	**0.657**
Model 1	0.852	0.841	0.836	0.869	0.932	0.899	0.821	0.781	0.788	0.825	0.443	0.576
Rule Based	0.846	0.832	0.838	0.903	0.918	0.910	0.834	0.634	0.720	0.534	0.774	0.632
SVM 5 - ch	0.849	0.853	0.845	0.873	0.96	0.909	0.916	0.765	**0.831**	0.679	0.505	0.527
SVM (1,2)-W	0.823	0.827	0.819	0.894	0.943	0.895	0.843	0.725	0.823	0.566	0.688	0.523

Table 3: Performance of our best performing models compared with that of two best performing models [8]: SVM 5-ch (5 character n-grams) and SVM (1-2)-Word n-grams.

P, R, F1 denote precision, recall and F1 score, respectively. SVM performance are 10-fold cross validation on the entire dataset. Model 1 performance was measured on a separate test set.

S. N	Total Parties count	Republican Candidates count	Democrat Candidates count	Total Candidates attacked	Total Candidates promoted	Actual Class	Predicted Class
1	1	2	0	1	1	Attack	Contrast
2	1	3	0	3	0	Attack	Attack
3	1	2	0	1	1	Attack	Contrast
4	1	1	0	1	0	Attack	Attack
5	1	2	0	1	1	Attack	Contrast
6	1	1	0	1	0	Attack	Attack
7	1	2	0	1	1	Attack	Contrast
8	1	5	0	5	0	Attack	Attack

9	1	1	0	1	0	Attack	Attack
10	1	1	0	1	0	Attack	Attack
11	1	1	0	1	0	Attack	Attack
12	1	1	0	1	0	Attack	Attack
13	1	1	0	1	0	Attack	Attack
14	0	0	0	0	0	Attack	Promote
15	1	1	0	1	0	Attack	Attack
16	1	1	0	1	0	Attack	Attack
17	2	1	1	2	0	Attack	Attack
18	1	0	1	1	0	Attack	Attack
19	1	1	0	1	0	Attack	Attack
20	1	1	0	1	0	Attack	Attack
21	0	0	0	0	0	Attack	Promote
22	2	4	1	4	1	Attack	Contrast
23	1	1	0	1	0	Attack	Attack
24	1	0	1	1	0	Attack	Attack
25	1	1	0	1	0	Attack	Attack
26	1	2	0	1	1	Attack	Contrast
27	1	1	0	1	0	Attack	Attack
28	1	1	0	1	0	Attack	Attack
29	1	2	0	2	0	Attack	Attack
30	1	1	0	1	0	Attack	Attack
31	1	1	0	1	0	Attack	Attack
32	1	2	0	1	1	Attack	Contrast
33	1	1	0	1	0	Attack	Attack
34	1	1	0	1	0	Attack	Attack
35	1	1	0	1	0	Attack	Attack
36	1	1	0	0	1	Attack	Promote
37	1	3	0	2	1	Attack	Contrast
38	1	0	1	1	0	Attack	Attack
39	1	1	0	1	0	Attack	Attack
40	1	1	0	1	0	Attack	Attack
41	1	2	0	1	1	Attack	Contrast

42	0	0	0	0	0	Attack	Promote
43	1	2	0	2	0	Attack	Attack
44	1	1	0	1	0	Attack	Attack
45	1	1	0	1	0	Attack	Attack
46	1	1	0	1	0	Attack	Attack
47	1	1	0	1	0	Attack	Attack
48	1	1	0	1	0	Attack	Attack
49	1	1	0	1	0	Attack	Attack
50	1	1	0	1	0	Attack	Attack
51	1	1	0	1	0	Attack	Attack
52	1	0	1	1	0	Attack	Attack
53	1	1	0	1	0	Attack	Attack
54	1	1	0	1	0	Attack	Attack
55	1	2	0	1	1	Attack	Contrast
56	1	0	1	1	0	Attack	Attack
57	1	1	0	1	0	Attack	Attack
58	1	1	0	1	0	Attack	Attack
59	1	3	0	2	1	Attack	Contrast
60	1	0	1	1	0	Attack	Attack
61	1	0	1	1	0	Attack	Attack
62	1	1	0	1	0	Attack	Attack
63	1	2	0	1	1	Attack	Contrast
64	1	0	1	1	0	Attack	Attack
65	1	1	0	1	0	Attack	Attack
66	1	2	0	1	1	Attack	Contrast
67	1	0	1	1	0	Attack	Attack
68	1	1	0	1	0	Attack	Attack
69	1	1	0	1	0	Attack	Attack
70	1	0	1	1	0	Attack	Attack
71	1	2	0	1	1	Attack	Contrast
72	1	1	0	0	1	Promote	Promote
73	1	1	0	0	1	Promote	Promote
74	1	0	1	0	1	Promote	Promote

75	1	1	0	1	0	Promote	Attack
76	1	0	1	0	1	Promote	Promote
77	1	1	0	0	1	Promote	Promote
78	1	1	0	0	1	Promote	Promote
79	1	1	0	0	1	Promote	Promote
80	2	1	1	1	1	Promote	Promote
81	1	1	0	0	1	Promote	Promote
82	1	0	1	0	1	Promote	Promote
83	2	1	1	1	1	Promote	Promote
84	1	1	0	0	1	Promote	Promote
85	1	1	0	0	1	Promote	Promote
86	1	0	1	0	1	Promote	Promote
87	1	0	1	0	1	Promote	Promote
88	1	1	0	0	1	Promote	Promote
89	1	1	0	0	1	Promote	Promote
90	1	1	0	0	1	Promote	Promote
91	1	1	0	0	1	Promote	Promote
92	1	0	1	0	1	Promote	Promote
93	1	1	0	0	1	Promote	Promote
94	0	0	0	0	0	Promote	Promote
95	1	0	1	0	1	Promote	Promote
96	1	1	0	0	1	Promote	Promote
97	1	1	0	0	1	Promote	Promote
98	1	0	1	0	1	Promote	Promote
99	1	0	1	0	1	Promote	Promote
100	2	1	1	1	1	Promote	Promote
101	1	1	0	0	1	Promote	Promote
102	1	1	0	1	0	Promote	Attack
103	0	0	0	0	0	Promote	Promote
104	1	0	1	0	1	Promote	Promote
105	1	1	0	0	1	Promote	Promote
106	1	2	0	1	1	Promote	Contrast
107	1	1	0	0	1	Promote	Promote

108	0	0	0	0	0	Promote	Promote
109	1	1	0	0	1	Promote	Promote
110	1	0	1	0	1	Promote	Promote
111	2	1	1	1	1	Promote	Promote
112	1	0	1	0	1	Promote	Promote
113	1	0	1	0	1	Promote	Promote
114	1	1	0	0	1	Promote	Promote
115	1	1	0	0	1	Promote	Promote
116	0	0	0	0	0	Promote	Promote
117	1	0	1	0	1	Promote	Promote
118	0	0	0	0	0	Promote	Promote
119	2	1	1	1	1	Promote	Promote
120	1	0	1	0	1	Promote	Promote
121	1	1	0	0	1	Promote	Promote
122	1	1	0	0	1	Promote	Promote
123	1	1	0	0	1	Promote	Promote
124	2	1	1	1	1	Promote	Promote
125	1	0	1	0	1	Promote	Promote
126	1	1	0	0	1	Promote	Promote
127	0	0	0	0	0	Promote	Promote
128	1	0	1	0	1	Promote	Promote
129	1	2	0	1	1	Promote	Contrast
130	2	1	1	1	1	Promote	Promote
131	1	1	0	0	1	Promote	Promote
132	2	1	1	1	1	Promote	Promote
133	1	0	1	0	1	Promote	Promote
134	1	1	0	0	1	Promote	Promote
135	1	1	0	0	1	Promote	Promote
136	1	1	0	0	1	Promote	Promote
137	1	0	1	0	1	Promote	Promote
138	1	0	1	0	1	Promote	Promote
139	1	1	0	0	1	Promote	Promote
140	1	0	1	0	1	Promote	Promote

141	1	0	1	0	1	Promote	Promote
142	1	0	1	0	1	Promote	Promote
143	1	1	0	0	1	Promote	Promote
144	1	1	0	0	1	Promote	Promote
145	1	1	0	0	1	Promote	Promote
146	1	0	1	0	1	Promote	Promote
147	1	1	0	0	1	Promote	Promote
148	1	1	0	0	1	Promote	Promote
149	1	0	1	0	1	Promote	Promote
150	1	0	1	0	1	Promote	Promote
151	1	0	1	0	1	Promote	Promote
152	1	0	1	0	1	Promote	Promote
153	1	0	1	0	1	Promote	Promote
154	1	0	1	0	1	Promote	Promote
155	1	0	1	0	1	Promote	Promote
156	0	0	0	0	0	Promote	Promote
157	1	1	0	0	1	Promote	Promote
158	1	2	0	1	1	Promote	Contrast
159	1	0	1	0	1	Promote	Promote
160	1	1	0	0	1	Promote	Promote
161	1	1	0	0	1	Promote	Promote
162	1	1	0	0	1	Promote	Promote
163	1	1	0	0	1	Promote	Promote
164	1	1	0	0	1	Promote	Promote
165	1	1	0	0	1	Promote	Promote
166	1	0	1	0	1	Promote	Promote
167	1	0	1	0	1	Promote	Promote
168	1	0	1	0	1	Promote	Promote
169	1	0	1	0	1	Promote	Promote
170	1	1	0	0	1	Promote	Promote
171	1	1	0	0	1	Promote	Promote
172	1	1	0	0	1	Promote	Promote
173	0	0	0	0	0	Promote	Promote

174	1	1	0	0	1	Promote	Promote
175	1	1	0	0	1	Promote	Promote
176	1	0	1	0	1	Promote	Promote
177	1	1	0	0	1	Promote	Promote
178	1	0	1	0	1	Promote	Promote
179	1	1	0	0	1	Promote	Promote
180	1	0	1	0	1	Promote	Promote
181	0	0	0	0	0	Promote	Promote
182	1	1	0	0	1	Promote	Promote
183	1	1	0	0	1	Promote	Promote
184	1	1	0	0	1	Promote	Promote
185	1	1	0	0	1	Promote	Promote
186	1	0	1	0	1	Promote	Promote
187	1	1	0	0	1	Promote	Promote
188	1	0	1	0	1	Promote	Promote
189	1	0	1	0	1	Promote	Promote
190	1	1	0	0	1	Promote	Promote
191	1	1	0	0	1	Promote	Promote
192	1	0	1	0	1	Promote	Promote
193	1	1	0	0	1	Promote	Promote
194	1	0	1	0	1	Promote	Promote
195	1	1	0	0	1	Promote	Promote
196	1	1	0	0	1	Promote	Promote
197	1	1	0	0	1	Promote	Promote
198	1	1	0	0	1	Promote	Promote
199	1	1	0	0	1	Promote	Promote
200	1	0	1	0	1	Promote	Promote
201	1	0	1	0	1	Promote	Promote
202	0	0	0	0	0	Promote	Promote
203	1	1	0	0	1	Promote	Promote
204	1	1	0	0	1	Promote	Promote
205	1	1	0	0	1	Promote	Promote
206	1	1	0	0	1	Promote	Promote

207	1	1	0	0	1	Promote	Promote
208	1	1	0	0	1	Promote	Promote
209	1	1	0	0	0	Promote	Promote
210	1	1	0	0	1	Promote	Promote
211	1	1	0	0	1	Promote	Promote
212	1	0	1	0	1	Promote	Promote
213	1	0	1	0	1	Promote	Promote
214	1	0	1	0	1	Promote	Promote
215	1	1	0	0	1	Promote	Promote
216	0	0	0	0	0	Promote	Promote
217	1	1	0	0	1	Promote	Promote
218	1	1	0	0	1	Promote	Promote
219	1	1	0	0	1	Promote	Promote
220	1	0	1	0	1	Promote	Promote
221	1	1	0	0	1	Promote	Promote
222	1	0	1	0	1	Promote	Promote
223	1	1	0	0	1	Promote	Promote
224	1	1	0	0	1	Promote	Promote
225	1	1	0	0	1	Promote	Promote
226	1	1	0	0	1	Promote	Promote
227	1	0	1	0	1	Promote	Promote
228	1	1	0	0	1	Promote	Promote
229	1	1	0	0	1	Promote	Promote
230	1	1	0	0	1	Promote	Promote
231	1	0	1	0	1	Promote	Promote
232	1	1	0	0	1	Promote	Promote
233	1	1	0	0	0	Promote	Promote
234	1	0	1	0	1	Promote	Promote
235	1	0	1	0	1	Promote	Promote
236	1	0	1	0	1	Promote	Promote
237	1	0	1	0	1	Promote	Promote
238	1	1	0	0	0	Promote	Promote
239	1	1	0	0	1	Promote	Promote

240	1	1	0	0	0	Promote	Promote
241	1	1	0	0	1	Promote	Promote
242	1	0	1	0	1	Promote	Promote
243	1	0	1	0	1	Promote	Promote
244	1	1	0	0	0	Promote	Promote
245	1	1	0	0	1	Promote	Promote
246	1	1	0	0	1	Promote	Promote
247	1	0	1	0	1	Promote	Promote
248	1	0	1	0	0	Promote	Promote
249	1	0	1	0	1	Promote	Promote
250	0	0	0	0	0	Promote	Promote
251	0	0	0	0	0	Promote	Promote
252	1	1	0	0	1	Promote	Promote
253	1	1	0	0	1	Promote	Promote
254	1	1	0	0	1	Promote	Promote
255	1	1	0	0	1	Promote	Promote
256	1	1	0	0	0	Promote	Promote
257	1	1	0	0	1	Promote	Promote
258	1	1	0	0	1	Promote	Promote
259	1	1	0	0	1	Promote	Promote
260	1	0	1	0	1	Promote	Promote
261	1	0	1	0	1	Promote	Promote
262	1	1	0	0	1	Promote	Promote
263	1	1	0	0	1	Promote	Promote
264	1	0	1	0	1	Promote	Promote
265	1	0	1	0	1	Promote	Promote
266	0	0	0	0	0	Promote	Promote
267	0	0	0	0	0	Promote	Promote
268	1	1	0	0	1	Promote	Promote
269	1	1	0	0	1	Promote	Promote
270	1	1	0	0	1	Promote	Promote
271	1	1	0	0	1	Promote	Promote
272	1	1	0	0	1	Promote	Promote

273	1	1	0	0	1	Promote	Promote
274	1	1	0	0	1	Promote	Promote
275	1	2	0	1	1	Contrast	Contrast
276	1	2	0	1	1	Contrast	Contrast
277	1	1	0	0	1	Contrast	Promote
278	1	2	0	1	1	Contrast	Contrast
279	1	3	0	2	1	Contrast	Contrast
280	1	2	0	1	1	Contrast	Contrast
281	1	2	0	1	1	Contrast	Contrast
282	2	2	1	0	0	Contrast	Contrast
283	1	2	0	1	1	Contrast	Contrast
284	1	2	0	1	1	Contrast	Contrast
285	1	1	0	0	1	Contrast	Promote
286	1	2	0	1	1	Contrast	Contrast
287	1	2	0	1	1	Contrast	Contrast
288	1	3	0	2	1	Contrast	Contrast
289	1	1	0	0	1	Contrast	Promote
290	1	1	0	0	1	Contrast	Promote
291	1	3	0	2	1	Contrast	Contrast
292	1	2	0	1	1	Contrast	Contrast
293	2	1	1	2	0	Contrast	Attack
294	1	1	0	0	1	Contrast	Promote
295	2	2	1	0	0	Contrast	Contrast
296	1	3	0	2	1	Contrast	Contrast
297	1	2	0	1	1	Contrast	Contrast
298	1	2	0	1	1	Contrast	Contrast
299	1	3	0	2	1	Contrast	Contrast
300	1	2	0	1	1	Contrast	Contrast
301	1	2	0	1	1	Contrast	Contrast
302	1	2	0	1	1	Contrast	Contrast
303	1	2	0	1	1	Contrast	Contrast
304	2	2	1	0	0	Contrast	Contrast
305	1	4	0	3	1	Contrast	Contrast

306	1	1	0	0	1	Contrast	Promote
307	1	2	0	1	1	Contrast	Contrast
308	1	2	0	1	1	Contrast	Contrast
309	1	2	0	1	1	Contrast	Contrast

Table 4: Classification result for model 4 in test set.

V. Conclusion

We investigated various approaches for classification of ad tone in political video advertisements under class imbalance and low data samples. To overcome the class imbalance problem, we investigated data augmentation techniques in the text. We found that creating new data samples using both synthetic text generation and language translation did not perform well. Our dataset is very small to generate good quality synthetic texts using recurrent neural networks.

We found that model 1, which is based on a simple neural network achieves better performance on the entire set and in the minority contrast class compared the previous work. However, model 1 did not perform better than the previous rule-based method on the minority contrast class. We found that the model 2, which has word embedding trained on our dataset performs better than the model 3 which is has pre-trained word2vec for word embedding. Both model 2 and model 3 did not perform better than any other models. They rapidly start overfitting on the training data in a few epochs of training.

We observed that the classification of political ad tones depends on the identification of the candidates that are being attacked and promoted in the ad. Our proposed model, model 4, which takes into account of the candidates that are being attacked and promoted in the ad, performs the best in the entire test set. It performs best on promote and contrast classes but the F1 score on attack class is reduced by approximately 3 percentage compared to previous work. Our proposed method has a clear advantage over rule-based method. Our method does not hard code the name of the candidates and parties as rules. Instead, use configuration to detect names and parties avoiding crafting rules for new dataset.

REFERENCES

[1] "Expenditures on presidential election 2012." [Online]. Available: https://www.opensecrets.org/pres12/expenditures.php.

[2] "About targeting for video campaigns." [Online]. Available: https://support.google.com/youtube/answer/2454017?hl=en.

[3] C. B. Williams, "Digital Advertising Expenditures in the 2016 Presidential Election," *Soc. Sci. Comput. Rev.*, vol. 36, no. 4, pp. 406–421, 2018.

[4] "Transparency report on political advertising on Google." [Online]. Available: https://transparencyreport.google.com/political-ads/overview.

[5] "The Wesleyan Media Project." [Online]. Available: http://mediaproject.wesleyan.edu/.

[6] "WMP Codebook 2010." [Online]. Available: http://mediaproject.wesleyan.edu/wp-content/uploads/2016/09/WMP-2010-releasecodebook_v1.2.pdf.

[7] "About The Wesleyan Media Project." [Online]. Available: http://mediaproject.wesleyan.edu/about/project-background/.

[8] L. Qi, C. Zhang, A. Sukul, W. Tavanapong, and D. A. M. Peterson, "Automated coding of political video ads for political science research," in *Proceedings - 2016 IEEE International Symposium on Multimedia, ISM 2016*, 2017.

[9] J. Stefanowski, "SIST 13 - Overlapping, Rare Examples and Class Decomposition in Learning Classifiers from Imbalanced Data."

[10] S. Hochreiter and J. J. Urgen Schmidhuber, "Long Short term Memory," *Mem. Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[11] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality."

[12] "Google Word2Vec." [Online]. Available: https://code.google.com/archive/p/word2vec/.

[13] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space."

[14] C. Lemnaru and R. Potolea, "LNBIP 102 - Imbalanced Classification Problems: Systematic Study, Issues and Best Practices," 2012.

[15] H. He and E. Garcia, "Learning from imbalanced data," *Ieee Trans. Knowl. Data Engin*, vol. 21, no. 9, pp. 1263–1284, 2009.

[16] Y. Yan, Y. Liu, M. L. Shyu, and M. Chen, “Utilizing concept correlations for effective imbalanced data classification,” *Proc. 2014 IEEE 15th Int. Conf. Inf. Reuse Integr. IEEE IRI 2014*, pp. 561–568, 2014.

[17] N. V Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic Minority Over-sampling Technique,” 2002.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks.” pp. 1097–1105, 2012.

[19] C. Szegedy *et al.*, “Going Deeper with Convolutions.”

[20] George A. Miller, “WordNet: A Lexical Database for English,” *Commun. ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[21] X. Zhang, J. Zhao, and Y. Lecun, “Character-level Convolutional Networks for Text Classification *.”

[22] W. Y. Wang and D. Yang, “That’s So Annoying!!!: A Lexical and Frame-Semantic Embedding Based Data Augmentation Approach to Automatic Categorization of Annoying Behaviors using #petpeeve Tweets *.”

[23] I. J. Goodfellow *et al.*, “Generative Adversarial Nets.”

[24] W. Fedus, I. Goodfellow, A. M. Dai, and G. Brain, “MASKGAN: BETTER TEXT GENERATION VIA FILLING IN THE.”

[25] “Toxic comment classification challange.” [Online]. Available: https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/discussion/52557.

[26] Y. Song, M. Zhao, J. Yagnik, and X. Wu, “Taxonomic classification for web-based videos,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 871–878, 2010.

[27] D. Brezeale and D. J. Cook, “Automatic Video Classification: A Survey of the Literature,” *IEEE Trans. Syst. Man, Cybern. Part C (Applications Rev.*, vol. 38, no. 3, pp. 416–430, May 2008.

[28] M. G. Christel, “Video Classification and Retrieval with the Informedia Digital Video Library System,” 2002.

[29] M. Features, P. Wang, R. Cai, and S. Yang, “A hybrid approach to news video classification multimodal features A Hybrid Approach to News Video Classification with,” no. February, pp. 2–6, 2015.

[30] H. Chatbri *et al.*, “Automatic MOOC video classification using transcript features and convolutional neural networks,” 2016.

[31] A. G. Hauptmann and R. Jin, “Multi-modal Information Retrieval from Broadcast Video using OCR and Speech Recognition.”

[32] M. A. Smith and T. Kanade, “Video skimming and characterization through the combination of image and language understanding techniques,” pp. 775–781, 2002.

[33] T. Young, D. Hazarika, S. Poria, and E. Cambria, “Recent Trends in Deep Learning Based Natural Language Processing.”

[34] R. Collobert, J. Weston, J. Com, M. Karlen, K. Kavukcuoglu, and P. Kuksa, “Natural Language Processing (Almost) from Scratch,” 2011.

[35] T. Mikolov, M. Karafiát, L. Burget, J. " Honza, " ˇ Cernock´ycernock´y, and S. Khudanpur, “Recurrent neural network based language model,” 2010.

[36] R. Socher *et al.*, “Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank.”

[37] Y. Kim, “Convolutional Neural Networks for Sentence Classification.”

[38] “Google Speech To Text.” [Online]. Available: https://cloud.google.com/speech-to-text/.

[39] “PySceneDetect.” [Online]. Available: https://pyscenedetect.readthedocs.io/en/latest/.

[40] “Google Vision API.” [Online]. Available: https://cloud.google.com/vision/.

[41] “TextGenRNN.” [Online]. Available: https://github.com/minimaxir/textgenrnn.

[42] B. Felbo, A. Mislove, A. Søgaard, I. Rahwan, and S. Lehmann, “Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm.”

[43] “Google Translation API.”

[44] “Keras text Processing API.” [Online]. Available: https://keras.io/preprocessing/text/.

[45] “Google Natural Language API.” [Online]. Available: https://cloud.google.com/natural-language/.

[46] “Open Secrets.” [Online]. Available: https://www.opensecrets.org/pacs/.

[47] “PTAA.”

[48] “PCL.” [Online]. Available: http://pcl.stanford.edu/.