A Novel Approach to Analyzing Political Sentiment on Social Media

Devin Conathan
conathan@wisc.edu

Scott Sievert
stsievert@wisc.edu

Joshua Vahala
vahala@wisc.edu

Abstract

The social media platform Twitter provides a great opportunity to analyze sentiment surrounding complicated events at a large scale. We develop a system that embeds new, unseen Tweets into a high-dimensional "tweet-space" that captures complex semantic relations by training on a large sample of Twitter posts ("tweets") related to politics. We use crowd sourcing and active learning to project these high dimensional tweet-vectors onto an interpretable low-dimensional embedding that can be used to efficiently label sentiment. We test the usefulness of our system by examining shifts in sentiment surrounding presidential debates between Donald Trump and Hillary Clinton.

1 Introduction

The social media platform Twitter has evolved through an ever-changing political and social atmosphere. Due to its wide array of content, Twitter offers a unique set of opportunities and challenges for political and social scientists to collaborate with large-scale machine learning researchers.

In particular, the UW–Madison School of Journalism and Mass Communication has been maintaining a large Twitter database described in Section 8.1. They have probed Twitter for analysis on topics such as the public's reaction to mass shootings, presidential debates, and the polarization of political expression [1].

Social and political science researchers seek systems, techniques, and approaches that are both effective in their domain and computationally feasible. We have narrowed our focus to the recent election season due to its unconventionality. Our system allows for arbitrary tweets to be embedded into a high-dimensional space where contextually similar entries are close in proximity. We use active learning to classify tweet sentiment into a low-dimensional embedding for a small set of tweets that are representative of our dataset's full tweet-space. We immediately see the relative sentiment of the new tweet by projecting any new tweet's high-dimensional embedding into this low-dimensional subspace.

We evaluate the success of our tweet-embedding system by inspecting the change in apparent sentiment surrounding the first presidential debate between Hillary Clinton and Donald Trump. We clearly see a polarizing shift in twitter conversation in the days following the debate.

2 Problem Statement

Twitter poses a unique challenge for machine learning and natural language processing (NLP). Basic NLP tools such as bag-of-words perform poorly on Twitter due to the brief and casual nature of the platform; tweets are limited to 140 characters, "emojii", misspellings and informal acronyms are common. Given only the raw tweet data, we seek to generate a dense vector representation that both captures the salient semantic qualities conveyed in the tweet and is robust to the noise typically found in the microblogging atmosphere.

We seek to embed the tweets in a vector space such that the Euclidean distance between any two tweet embeddings quantitatively captures the difference between the tweets in meaning, intention, and topic. Such embeddings would prove to be valuable for machine learning and NLP tasks such as classification and topic modeling to track the volume of specific topics or kinds of tweets over time.

Ideally, these embeddings would be low-dimensional and interpretable by a human. For example, if there were a $\mathbb{R}^2$ embedding where the first dimension corresponded to a "pro-anti" spectrum and the second corresponded to a "Clinton-Trump" spectrum, a human could quickly and easily interpret a visual representation of these embeddings and recognize patterns over time. We use active learning, crowd sourcing and ordinal embedding algorithms to generate such an embedding.

We have built a self-contained and computationally-scalable system that can be trained on a corpus of raw tweets and subsequently yield such embedding(s) from new, unseen tweets. In our specific implementation, we focus on the political arena by training our high-dimensional encoder specifically on a tweet corpus from Donald Trump's candidacy announcement to just past Election Day. We then select a more constrained set of tweets surrounding the Clinton-Trump debates to use for determining this low-dimensional embedding and to measure the change in sentiment surrounding the two candidates surrounding the first debate.

3 Prior work

3.1 Word and Tweet Embeddings

Word embeddings have become a popular and effective model for representing words for machine learning algorithms [2]. It has been demonstrated that the dense vectors these models generate capture meaningful semantic relations between words [3]. It is not immediately obvious how to extend these models, which assign each word a vector, to groups of words (sentences, documents, tweets, etc.). Some have tried implementations that look at weighted averages of word vectors in a group [4]. This approach has merits, but it throws away information such as word order, which is crucial when the input document size is small.

We draw inspiration for our project from two homonymous (*tweet2vec*) implementations from [5] and [6]. Both start by representing each tweet as an $n \times d$ matrix, where d is the total number of unique characters in the training set and n is the max number of characters for a tweet (140). The rows of the $n \times d$ tweet matrix are one-hot encoded vectors where a 1 in the (i, j) position means that the ith character of the tweet is the character that j encodes.

In [5], these tweet matrix representations are fed into a series of convolutional layers which continue into a long short-term memory (LSTM) network and is trained as an encoder-decoder. In [6] they are fed directly into a gated recurrent unit (GRU) network and trained to predict hashtags. In both cases their systems embedded each tweet in a vector space, and they demonstrate the performance and expressiveness of these embeddings through cross-validation and various tasks like sentiment analysis.

In [6], two models are compared: a "word level baseline" and their proposed *tweet2vec* model. The former takes a matrix of word embeddings from an unspecified word embedding model as an input whereas the latter takes the aforementioned matrix of one-hot encoded character embeddings. The motivation for the latter is that tweets often contain misspellings or otherwise rarely seen words, which may not exist in a specific word embedding model. Furthermore, because there is a predetermined small number of character options and a limit on character length for tweets (140), the character embeddings are easier to process.

3.2 Active Learning

Another method of generating embeddings is by gathering comparisons of triplets of samples (e.g., "i is more similar to j than k") and then grouping together the most similar samples. These comparisons rely on human judgment, which is expensive to obtain in practice. There has been much research into generating these embeddings with as few comparisons as possible. There exist adaptive sampling algorithms that rank all $\mathcal{O}(n^3)$ possible triplets (for n samples) by how much information each comparison would yield. By only selecting the most informative comparisons to label, these

algorithms obtain accurate embeddings with as few as $\mathcal{O}(dn \log n)$ comparisons, where d is the dimension of the embedding [7, 8].

Many of these adaptive sampling algorithms are 'sample-agnostic' in that they do not rely on any features the data may have. Active learning algorithms such as uncertainty sampling can further increase the efficiency of these queries by maximizing the information gain from each obtained label [9]. There has been work on finding the information gain for each feature [10] which has shown gains in a separate classification task.

There is some recent work on using active learning and feature information for triplet constraints [11]. In this, they compare and make modifications to the crowd kernel algorithm [7]. In the below work we look into similar modifications to the stochastic triplet embedding (STE) [8] algorithm. We have performed simulations on similar data and do not embed the tweets using the modifications described in Section 5.5, though these modifications are of interest for future experiments.

4 High-Dimensional Embedding via Neural Network

4.1 Overview

We propose a variation on the prior *tweet2vec* implementations more similar to [6] than [5] in that we strip hashtags from tweets and train the embeddings to be predictive of the hashtags. The motivation for this choice is that hashtags often function as a proxy for the emotional or semantic content of the tweet; that is, similar tweets will have similar hashtags because they express similar ideas, and thus their embeddings should be similar.

We contend that the implementation in [6] throws away too much information by disregarding the word embeddings completely, which contain important contextual information. While their analysis showed a remarkable performance bonus from using the character-embedding model over the word-embedding model, little is said about how the word-embeddings were obtained. Small-scale experiments have demonstrated the utility of a Gensim implementation of a word2vec model [12] trained on a relatively small number of tweets (e.g. a model trained on just a single day's worth of tweets was able to capture semantic relations such as basic analogies). We believe that an even better *tweet2vec* model can be built by incorporating *both* the word- and character-embeddings, provided that the former is properly tuned and trained. Further, this hybrid word/character-embedding approach should perform especially well in a restricted domain such as political discourse, where the vocabulary is likely smaller than a more general random sampling of tweets.

4.2 Neural Network Architecture

Our architecture is based upon the merging of word-based and character-based embeddings. The raw tweet input is separated into two different input types to generate these respective embeddings.

The input for the word-based embedding is generated as $W \in \mathbb{R}^{n \times k}$ matrix where $n = 50$ is the max number of words possible in a tweet, and $k = 100$ is the *word2vec* embedding dimension. Each row of W is the word2vec embedding of each word in the tweet. The word2vec embedding is trained on the full political dataset (see Section 8.2) stripped of hashtags, unique urls, and twitter handles.

The input of the character-based embedding is generated as $C \in \mathbb{R}^{n \times r}$ where $n = 50$ is the max number of words possible in a tweet, and $r = 68$ is the number of possible characters allowed in a tweet (26 letters, 32 punctuation marks, and 10 digits are included). Each row of C is then the sum of the one-hot character embeddings of each character in the word (i.e., the word "bad" would be encoded as `[1, 1, 0, 1, 0,...]` where the first four characters correspond to characters "a", "b", "c", and "d").

Each input matrix W and C is fed into a bidirectional Gated Recurrent Unit (Bi-GRU) layer with $4k = 800$ and $8r = 544$ nodes respectively. Each output of these Bi-GRU nodes is merged into a single layer with a 10% dropout to help avoid overfitting. This merged layer output is then fed into a 3000 node layer with a 50% dropout and a ReLU activation. We finish the architecture with a fully connected 300 node layer with linear activation and a 1000 node output layer with softmax activation. The 300 node layer outputs our high-dimensional embedding as a vector in $\mathbb{R}^{300}$. The output layer represents the probability that each of the 1000 hashtags considered in our training set would be used

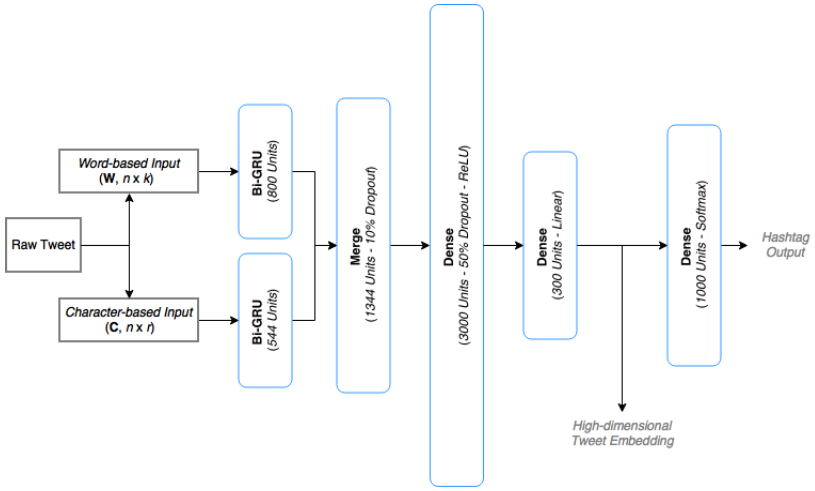


Figure 1: A diagram of the neural network architecture for our model

with the input tweet. See Figure 1 for an overview of the architecture. We detail the training data fully in Section 8 (Appendix).

4.3 Model Training

4.3.1 Training and Test Data

We trained our architecture on a dataset collected from June 1, 2015, a few weeks prior to Donald Trump's announcement of his candidacy for President of the United States, to November 12, 2016, just after the November 8th election day. We confined the context of our tweets to the political arena by requiring that tweets contain at least one word from a set of keywords including {trump, hillary, clinton, debate, president, white house, makeamericagreat, americafirst, imwithher, prolife, drumpf, wikileaks, etc.}. See the Section 8.2 for a more detailed description of the dataset including training and test splits.

In order to get the word embeddings for the W input matrix, we trained a Gensim implementation of a word2vec model [12] using all words in the cleaned dataset as the input vocabulary.

Our neural network model was implemented in Keras [13] with Tensorflow as a backend and was trained using Stochastic Gradient Descent (SGD) to minimize the categorical cross-entropy of the model, during which we saw a consistent decline in both training loss and validation loss (see Figure 2). We trained the model in batches of 1000 tweets on $E = 133$ epochs of 1 million tweets each. We used an initial learning rate $\lambda_0 = 0.3$ with a decay $\delta = 0.01$, where the learning rate at epoch $t \in [1, 2, ..., E]$ is defined:

$$\lambda_t = \frac{\lambda_0}{1 + \delta t} \tag{1}$$

Training the model itself presented a technical challenge. With such a large dataset, we couldn't guarantee that the training data (both in its raw form and "input matrix" form) and model parameters could fit into memory. We took advantage of Keras' `fit_generator` method, which expects a function that yields input data rather than one giant matrix/tensor with labels.

We developed a system that iterates through the lines of the raw input file and does the preprocessing (cleaning, tokenizing, turning into matrices) in a streaming fashion – that is, no more than one batch is held in memory at any given time. Furthermore, because the actual model training was done on the GPU, we had ample cores available to run the preprocessing in parallel, which increased our training efficiency.

The entire training process took approximately 48 hours using a machine with eight 3.3 GHz cores, 32GB RAM, and a GeForce GTX 750 Ti GPU. Note that the validation loss in Figure 2 is consistently less than that the training loss. This result is due in part to the training loss being calculated with

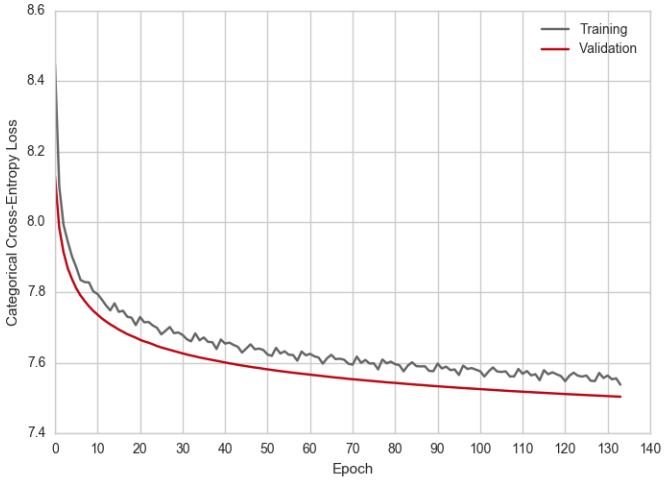


Figure 2: The neural network architecture was trained over 133 epochs of 1M tweets each to minimize the categorical cross-entropy on hashtag labels.

regularization methods such as dropout being turned on, while validation error is calculated with these methods turned off.

5 Low-Dimensional Embedding Via Active Learning

5.1 Overview

Straightforward sentiment analysis of tweets relies on labeled data and would involve a laborious process of prescribing positive or negative sentiment labels to a training set of tweets. We propose an alternative approach where we generate a low-dimensional embedding of tweets in which the distance between tweets is a function of how similar they are in meaning and sentiment.

We efficiently obtain this low-dimensional embedding on a small subset of our tweets using adaptive sampling and active learning; we ask a small number of triplet queries of the form "is tweet i is more similar in sentiment to tweet j than tweet k?". This method of generating embeddings using of triplet comparisons is a well studied problem [7, 8, 11].

With enough responses to these queries, we can generate an embedding that has the sentiment-similarity properties we seek. We can then find a mapping from our high-dimensional tweet-space (where we have the embeddings for all of our tweets) to the low-dimensional tweet-space, so we can project samples of tweets into the low-dimensional space and analyze the relative densities.

5.2 Stochastic Triplet Embedding

We implement the Stochastic Triplet Embedding (STE) algorithm to select triplets to be queried and also generate the embedding[8]. STE is able to generate an accurate embedding with $\mathcal{O}(dn \log n)$ responses, which is far fewer than the total number of $\binom{n}{3}$ possible triplet labels.

STE selects all possible triplets (k, i, j) and finds the probabilities that item k is more similar to item i or j as $p_{k,i,j}$. Further detail is available on how this is used in triplet embeddings [7, 8]. Modifications to crowd kernel include feature vector modifications [11] and changes to the triplet probability $p_{k,i,j}$ [8]. In [8] this probability can be represented as

$$p_{k,i,j} = \frac{s_{i,k}}{s_{i,k} + s_{j,k}} \tag{2}$$

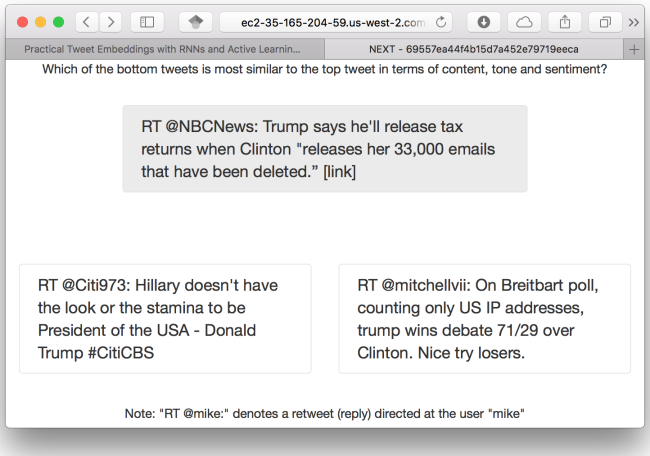


Figure 3: An example NEXT query shown to Mechanical Turk participants

where $s_{i,k} = (1 + \|x_i - x_k\|)$ is a similarity measure between the current embedding coordinates x_i and x_k for examples i and k, respectively. STE uses this probability to choose to query the triplet (k, i, j) with the highest probability $p_{k,i,j}$.

5.3 Choosing a Representative Subset

While STE significantly reduces the number of responses needed to generate the embedding, $\mathcal{O}(dn \log n)$ is still very large for datasets of our scale. We cannot feasibly generate an embedding for our whole dataset given its size. Instead, we generate the embedding for a smaller representative set of tweets.

To choose this subset, we cluster the high-dimensional embeddings using k-means clustering with $k = 100$ clusters. We choose the 100 tweets that are closest to the cluster centers as our representative sample. Our motivation behind this was to choose a contextually diverse set that would more accurately represent the full tweet-space than a random sample of 100 tweets.

We did do some cleaning and preprocessing on the tweets (e.g. replaced links with '[link]'). After preprocessing, some tweets in were found to be either lacking sufficient information (e.g., "RT @RobPulseNews: @realDonaldTrump [link]") or were in Spanish, so we removed these from the final active learning set. We used a total of $n = 94$ tweets to generate our $d = 2$ dimensional embedding.

5.4 Implementation: Mechanical Turk and NEXT

We use NEXT [14] to implement our adaptive sampling system and Mechanical Turk to collect human responses. The system NEXT makes it simple to collect human responses while implementing an adaptive algorithm like STE. An example query from our experiment is shown in Figure 3. The NEXT implementation of STE chooses the next query by finding probabilities for a random subset of triplets instead of all possible triplets. This is to reduce computation time and provide a better experience for the user.

We implemented this version of STE without modification, though in Section 5.5 we simulate a modification that utilizes the high-dimensional embeddings associated with each tweet for selecting new queries. We show that including these modifications could lead to more efficient implementations for future experiments.

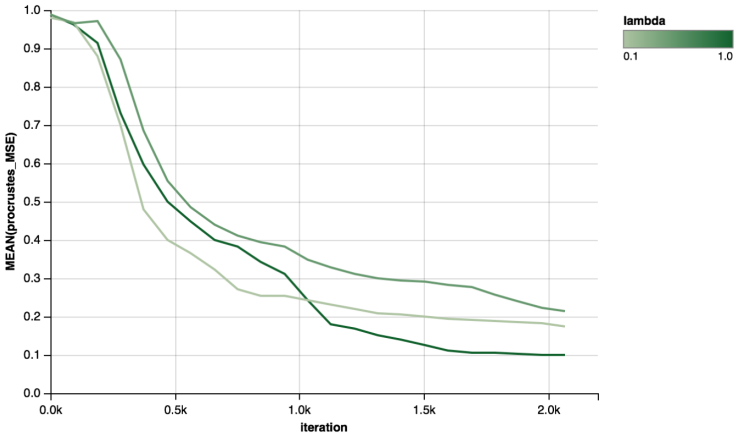


Figure 4: Here we simulate the adaptive modification method for various values of λ, which alters how dependent the adaptive sampling is on the current low-dimensional embedding state. The y-axis show s the embedding mean squared error (MSE) after accounting for embedding rotation. We use simulated data of similar dimension or our specific implementation with $n = 94$ examples and features in $x_i \in \mathbb{R}^{300}$.

5.5 A Proposed Modification to STE

It is intuitive that using our high-dimensional representations of each tweet could benefit us by reducing the number of queries needed for the same embedding accuracy as a method that does not use this information. Fewer required responses directly translated to a decrease in the required time and money.

We simply need to modify the probability metric $p_{k,i,j}$, defined in Section 5.2, to include feature information. We accomplish this by redefining our similarity measure between examples i and k to be

$$s_{i,k} = \lambda \left(1 + \|x_i - x_k\|\right) + (1 - \lambda) \left(1 + \|a_i - a_k\|\right) \tag{3}$$

where λ is the amount of confidence in the original STE embedding, and for example i, x_i is the current low-dimensional embedding coordinates, and a_i is the high-dimensional vector returned by our neural network.

Figure 4 shows this formulation for $\lambda \in \{0.1, 0.5, 1\}$. Note that $\lambda = 1$ is the STE algorithm that is currently used by NEXT. STE can only recover the original embedding up to a Procrustes transformation, which accounts for rotation of the embedding, and hence we do a Procrustes transformation between the ground truth embedding and estimated embedding before plotting the MSE.

We see that in early iterations using primarily high-dimensional feature information ($\lambda \approx 0$) is advantageous but using object information ($\lambda \approx 1$) is more useful later. It seems worthwhile to investigate by changing λ_k to tune the amount of information used from the features instead of a fixed λ.

5.6 Mapping New Tweets into the Low-Dimensional Tweet-Space

In order to apply our approach to a large set of tweets, we need a mapping from our high-dimensional embedding to the low-dimensional embedding that minimizes the error the set of $n = 94$ tweets where we have both embeddings. We can then apply this mapping to any general set of tweets to yield what their distribution looks like in the low-dimensional space. We find this linear mapping by minimizing the following loss function over w:

$$f(w) = \frac{1}{n} \sum_{i=1}^{n} (y_i - w x_i)^2 + \lambda \|w\|_F^2$$

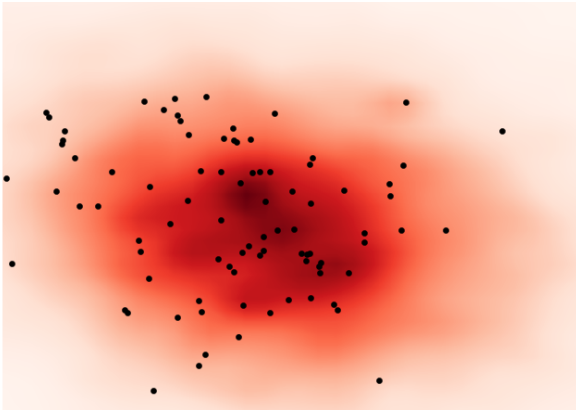


Figure 5: The resulting low-dimension embedding of our 94 representative tweets (black dots) overlaid with the density of tweets after mapping tweet's vectors in $\mathbb{R}^{300}$ into this embedding in $\mathbb{R}^2$ for this debate sample.

where $w \in \mathbb{R}^{2\times 300}$ represents our linear map from $\mathbb{R}^{300}$ to $\mathbb{R}^2$. In this standard machine learning problem we choose Tikhonov regularization; we have no reason to believe that w should be unbounded or that only a sparse subset is important in determining the low-dimensional embedding. Another reason to choose this mapping is because it is stable [15] if our features are bounded which implies low generalization error [16]. Generaization error is important because we're training on a small subset of the available tweets (94 out of 800,000), even though we carefully selected representative tweets as described in Section 5.2.

In Figure 5 we plot the results of our embedding with the relative densities of all Twitter activity in our debate sample projected into the same subspace. In Section 6.4 we examine this embedding in more depth.

6 Evaluation

6.1 Predicted Hashtags on Holdout Set

As seen in Figure 2, the trained model has an average categorical cross-entropy loss of about 7.5 on the holdout set. This number is abstract and hard to interpret. Also, because there is overlap and reduncancy between hashtags (e.g. #hillary4prison and #hillaryforprison), it is not practical to ever expect near-perfect performance. However, our intention was never to predict hashtags with precise accuracy, but to capture the meaning and intention behind the the tweet. In Table 1, we present a random sample of tweets from the holdout test set and the top 5 predicted hashtags (that is, with the highest probability). We conclude that the model clearly demonstrates an ability to distill the topic of the tweet.

6.2 Binary Classification Using High-Dimensional Embeddings

This analysis is designed to demonstrate how informative the high-dimensional vectors generated by our model are relative to a few baselines. For the sake of efficiency and simplicity, we developed a series of binary classification tasks on which we can compare the performance using classification accuracy and area-under-curve. These binary classification tasks consist of selecting two sets of hashtags and calling tweets with any hashtags from the first set *class 1* and tweets with any hashtag from the second set *class 2* (tweets with overlapping hashtags are disregarded).

We performed this evaluation on the holdout set – we did a 75/25 split on the holdout set and divided it into a training set consisting of 750,000 tweets and a testing set of the 250,000 tweets. Then, for

Table 1: A random sample of tweets from the holdout set and their predicted hashtags

Tweet	Top 5 Predicted Hashtags
No Gun Voter Left Behind - Why all California Gun Owners MUST VOTE https://t.co/Mb8sCXO8b7 #nra #tcot	#pjnet, #2a, #tcot, #nra, #ccot
Cameron shares disturbing outlook for UK defence if #Brexit campaign severs us from Europe. https://t.co/Md51omT8aN	#news, #iran, #brexit, #india, #syria
Learn how to identify and cope with stress. #Military #TalkRadio Live Now: https://t.co/tcv8qWPGQt .	#talkradio, #military, #healthcare, #besttalkradio, #usa
RT @TrumpMyPres: #KellyFile #Hannity #oreillyfactor RETWEET THIS #glennbeck #TheBlazeTV @theblaze #socialmedia #Instagram #asksean https...	#makeamericagreatagain, #trump2016, #maga, #trumptrain, #imwithher
RT @DMAC193: "@billboard: Canada's immigration site crashed as momentum built for a Donald Trump victory on #ElectionDay: https://t.co/xO14...	#trump2016, #trump, #trumptrain, #maga, #makeamericagreatagain
'Making A Murderer' Meets #DonaldTrump https://t.co/1a1X7qFRcJ https://t.co/qA5ZAhKOEK	#iran, #news, #trump, #jobs, #usa
RT @realDonaldTrump: Brought to you by @HillaryClinton & her campaign- in Chicago, Illinois. #BigLeagueTruth #DrainTheSwamp https://t.co/1...	#trump2016, #maga, #makeamericagreatagain, #trumptrain, #draintheswamp
RT @Always_Trump: Mainstream media REFUSES to publish this image! #TrumpPence16 https://t.co/VI13bSm3ZU	#trump, #maga, #trump2016, #hillary, #crookedhillary
Pediatric Dentist - Part Time - 20817 - Kool Smiles #jobs https://t.co/VBU9YpHtta	#jobs, #hiring, #job, #healthcare, #careerarc
RT @Maryam_Rajavi: Maryam Rajavi: Democratic Muslims are the Force to Defeat Islamic Fundamentalism #Iran #ParisAttacks https://t.co/6ipN...	#iran, #syria, #freeiran, #no2rouhani, #iraq

each of the tasks, we trained and tuned (via 5-fold cross-validation) a support vector machine (SVM) with linear kernel on the training set and evaluated the performance on the test set.

The first baseline we compared to was Bag-Of-Words (BOW) with term frequency–inverse document frequency (TF-IDF) weights clipped to the 5000 most frequent words (we tried 1000 and 2000 as well, but 5000 consistently outperformed these two choices). The second baseline is what we refer to as "Sent2Vec" which comes from [17]. This "simple but tough-to-beat baseline" is basically a TF-IDF-weighted averaging of word embeddings with the first PCA direction removed, and has been shown to outperform complicated RNN-based models like ours when the training dataset is small. We used the same word2vec model to get the word embeddings for this model, so this will be a good assessment of how much our model improved upon the information contained in the word embeddings alone through its character-based input and more complex neural network architecture. In other words, because both "Sent2Vec" and our Tweet2Vec model utilize the same well-trained word2vec models, we expect any improvement in classification accuracy to be primarily due to each model's ability to interpret the contextual connection between each word embedding.

Table 2: Binary classification performance using features computed from our model versus some baselines

Class1	Class1 n	Class2	Class2 n	Feature	Accuracy	AUC
#clinton #clintonkaine #clintons #hillary #imwithher	train: 6277 test: 2127	#trump	train: 6879 test: 2417	**Tweet2Vec**	**0.907**	**0.966**
				Sent2Vec	0.532	0.589
				BOW	0.476	0.5
#hillary4prison #hillaryindict-ment #hillarysemails ...	train: 1433 test: 532	#maga	train: 4696 test: 1549	**Tweet2Vec**	**0.93**	**0.97**
				Sent2Vec	0.744	0.642
				BOW	0.269	0.508
#msnbc #cbs #abc #ap #nbc	train: 554 test: 191	#foxnews #fox	train: 703 test: 247	**Tweet2Vec**	**0.753**	**0.86**
				Sent2Vec	0.564	0.549
				BOW	0.438	0.481

We found that the features from our Tweet2Vec model consistently outperform (often by a very large margin) these baselines. We performed this test on a set of progressively more difficult tasks. They go from distinguishing Clinton vs. Trump tweets, anti-Clinton vs. pro-Trump tweets, to Fox news vs. other news networks. The results are found in Table 2.

6.3 Tweet Proximity Using High-Dimensional Embeddings

Since the model was trained to be predictive of hashtags, the previous performance is not surprising. The real test is assessing if these tweet embeddings are suitable for more nuanced tasks like semantic analysis. Since we have no proper labels for this kind of task, we present a qualitative assessment here.

For each sample we find its nearest neighbor (using Euclidean distance as a metric) among all 250k tweets in our neural network holdout set. We provide a sample of closest tweets in Table 3. Note that although the topic is often different for each sample, the nearest tweets are mostly similar in tone ("anti-Clinton", "anti-Trump", or the two "joking" tweets). Also note that off-topic tweets about voting for your favorite celebrity (e.g., for the Video Music Awards (VMA)) got grouped together. This will prove valuable in filtering out irrelevant tweets.

6.4 Event-Based Sentiment Analysis Using Low-Dimensional Embeddings

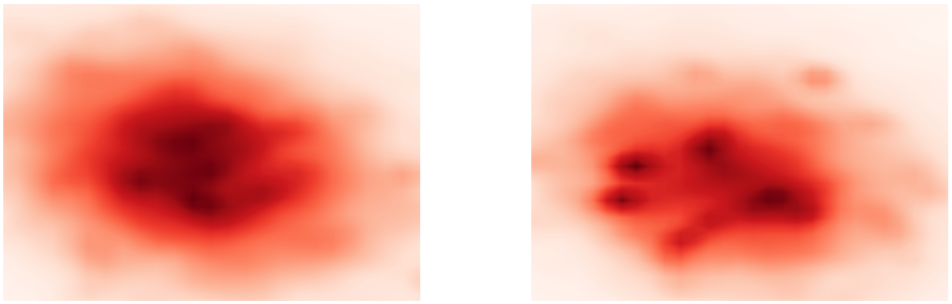
Finally, we present a qualitative analysis to assess the relevance of our low-dimensional embeddings. For this analysis, we take samples of tweets from two days before the first presidential debate on September 26th and two days after. We compute their high-dimensional vectors using our Tweet2Vec model and then project them onto the low-dimensional space using the linear map described in Section 5.6. The resulting plot for the two time periods is in Figure 6.

Before the debate, the distribution appears Gaussian: no one topic or group is dominating the activity or conversation. After the debate, we see clusters form. We've identified some of these clusters; the big one in the lower left and the small one in the upper right seem to be mostly neutral news tweets that simply report on the debate.

Next we explore the polarization aspect – do some clusters represent distinct "sides" or "topics" of the debate? In Figure 7, we've selected ten tweets from the general debate-related pool (five anti-Trump/pro-Hillary and five pro-Trump/anti-Hillary) and plotted them. There is clearly a trend for Hillary-positive tweets to appear in the lower and lower right portion of the plot whereas Trump-positive tweets appear in the upper half. We expect this distinction to improve as we refine the low-dimensional embedding by collecting more Mechanical Turk responses.

Table 3: A random selection of tweets from the neural network holdout set with its Euclidean distance nearest neighbor in high-dimensional tweet-space

Original Tweet	Closest Tweet
RT @eddiecarl4468: Omg!!! They're discussing how much Algeria needs to pay to get off terror list!!! Treason!! #PodestaEmails13...	RT @RealJamesWoods: So when you consider to what astounding lengths @HillaryClinton has gone to hide her #emails, they must be career-endin...
RT @Always_Trump: Mainstream media REFUSES to publish this image! #TrumpPence16 https://t.co/VI13bSm3ZU	RT @Always_Trump: #CrookedHillary is spending $1 Million to have this image deleted from the Internet! #Trump #Trump2016 https://t.co/ZT4g1...
RT @meanpIastic: "barack please don't leave me with them" "joe you're leaving when I leave" "oh right lmao love u" https://t.co/bZ92at6xt0	RT @torinxa: what if god is real and trump laughs after this election is over and says "alright this is a joke you've all been punk'd"
Comment: The #hackers who hate Donald Trump https://t.co/Q14HBg2tNi	Andy Samberg Blasts Bill Cosby and Donald Trump in Emmys Monologue http://t.co/1OlpVgYw2b #music #chart #info
RT @backseatslrh: LUKE if you're still online lead us in voting you got this don't let ur fingers slip #ShesKindaHotVMA	GUYS IF YOURE VOTING FOR JENN MAKE SURE TO USE THE RIGHT HASHTAG!! (#ChoiceFemaleWebStar)



(a) Before debate

(b) After debate

Figure 6: A comparison before and after the first presidential debate (September 26, 2016)

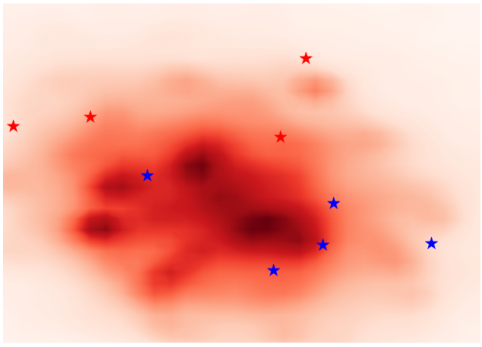


Figure 7: Five pro-Trump/anti-Hillary tweets (red) and five anti-Trump/pro-Hillary tweets (blue)

7 Conclusion and Future Work

In this project, we have developed a neural network model which turns raw tweets into high-dimensional vectors. Using active learning and crowd sourcing, we have also generated a map that projects these high-dimensional vectors into a two-dimensional subspace that is easy to understand. Through various quantitatve and qualitative assessments, we have demonstrated that our high-dimensional vectors are informative, and that our two-dimensional representation offers a human-interpretable and intuitive visualization of Twitter activity.

For future work, we would like to extend the approach to sentiment analysis from Section 6.4 to larger time periods. We could easily generate the same visualization for any day (or collection of days) in the election cycle. We could, for example, assess what impact events like the releasing of FBI director's James Comey's letter to Congress had on Twitter activity. We could generate an animation that shows the flow of Twitter activity over time, highlighting key points.

We also want to investigate controlling the amount of information used from the feature vectors per iteration. We suspect that it provides adaptive gains to use mostly feature information in early queries and mostly object information in later queries.

8 Appendix

8.1 Twitter Database

The Twitter data is collected through Twitter's streaming API and includes a random 10% sample of all (global) Twitter activity from February 2015–June 2015 and a random 1% sample thereafter. For every tweet in the database, in addition to the tweet text itself, we have extensive metadata such as user bios and language. The database is a maintained as a Hive table, and we estimate that it contains approximately 2 billion tweets. This database is large, on the order of 100 terabytes, and (in most cases) takes more than an hour to retrieve any reasonably-sized set of tweets.

8.2 Model Training Dataset Details

Date Range(s): 6/01/2015 – 11/12/2016

Keywords

- trump, hillary, clinton, debate, president, white house, conservative, liberal, libertarian, republican, democrat, gop, makeamericagreat, libtard, constitution, obama, #america, #merica, politic, #capitalism, mikepence, timkaine, teaparty, military, americafirst, #freedom, #liberty, #defenddonald, government, election, secondamendment, 2ndamendment, defendthesecond, supportthetroops, #guncontrol, supportveterans, honorvets, thankvets, usflag, rightwing, prolife, progod, progun, berniebros, campaign, drumpf, imwithher, imnotwithher,

fbi, wikileaks, supporters, voting, poll, dnc, rnc, candidate, potus, supreme court, rally, pence, tax, racist, #tcot, demsinphilly, draintheswamp, #pjnet, #ccot, dncleaks, #usa, basketofdeplorables, #lockherup, #benghazi, #imwithyou, #wakeupamerica, #uniteblue, #demconvention, #feelthebern, #demexit, strongertogether, #cnn, #podestaemails, bernieorbust, #jillnothill, #riggedsystem, berniesanders, garyjohnson, jillstein, alsmithdinner, #bigleague, #corruption, #tntweeters, #altrightmeans, wethepeople, blacklivesmatter, #jobs, #nra, #iran, dapl, #hrc, clint, hilla, hila, #maga, #guilty, #blm, #nbc4dc, voterfraud, voteblue, bluelivesmatter, alllivesmatter, ourrevolution, media, #msm, billary, #dems

Quantity Details

- Total Tweet Count: 26,133,547 (2.9 GB)
- Training Set Tweet Count: 6,331,717
- Test Set Tweet Count: 251,991

Data cleaning and notes:

- Any hyperlink is changed to "url".
- Any hashtags are removed.
- Any references to Twitter usernames are changed to "@user".
- Only tweets with hashtags in the top 1000 most often used hashtags are used for training.

8.3 Sentiment Analysis Dataset Details

Date Range(s): 9/24/2016 – 9/28/2016 and 10/17/2016 – 10/21/2016

Keywords

- trump, hillary, clinton, president, debate

Quantity Details

- Total Tweet Count: 822878 (100 MB)
- NEXT Implementation Tweet Count: 94

Data cleaning and notes:

- Any hyperlink is changed to "url" for High-Dimensional Embedding.
- Any hashtags are removed for High-Dimensional Embedding.
- Any references to Twitter usernames are changed to "@user" for High-Dimensional Embedding.

Date Range(s): 2016-9-24 - 2016-9-28 (Debate 1), 2016-17-10 - 2016-10-21 (Debate 3) Keywords:

8.4 Example Tweets used in NEXT

- RT @LVBurke: "Trump doesn't have anything to offer but anger, and grievance, and blame." #Obama
- RT @tpel78: Be like a terrorist vote for #HillaryClinton [link]
- RT @ish10040: VIDEO : Trump Asks Hillary to Return the Millions of Dollars She's Taken from Countries that Abuse Women and Gays [link]

References

[1] L. Bode, A. Hanna, J. Yang, and D. V. Shah, "Candidate networks, citizen clusters, and political expression: Strategic hashtag use in the 2010 midterms," *The ANNALS of the American Academy of Political and Social Science*, vol. 659, no. 1, pp. 149–165, 2015.

[2] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *CoRR*, vol. abs/1310.4546, 2013.

[3] G. Z. Tomas Mikolov, Scott Wen-tau Yih, "Linguistic regularities in continuous space word representations," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT-2013)*, Association for Computational Linguistics, May 2013.

[4] C. D. Boom, S. V. Canneyt, T. Demeester, and B. Dhoedt, "Representation learning for very short texts using weighted word embedding aggregation," *CoRR*, vol. abs/1607.00570, 2016.

[5] S. Vosoughi, P. Vijayaraghavan, and D. Roy, "Tweet2vec: Learning tweet embeddings using character-level CNN-LSTM encoder-decoder," *CoRR*, vol. abs/1607.07514, 2016.

[6] B. Dhingra, Z. Zhou, D. Fitzpatrick, M. Muehl, and W. W. Cohen, "Tweet2vec: Character-based distributed representations for social media," *CoRR*, vol. abs/1605.03481, 2016.

[7] O. Tamuz, C. Liu, S. J. Belongie, O. Shamir, and A. T. Kalai, "Adaptively learning the crowd kernel," *CoRR*, vol. abs/1105.1033, 2011.

[8] L. van der Maaten and K. Weinberger, "Stochastic triplet embedding," in *2012 IEEE International Workshop on Machine Learning for Signal Processing*, pp. 1–6, Sept 2012.

[9] J. Zhu, H. Wang, T. Yao, and B. K. Tsou, "Active learning with sampling by uncertainty and density for word sense disambiguation and text classification," in *Proceedings of the 22Nd International Conference on Computational Linguistics - Volume 1*, COLING '08, (Stroudsburg, PA, USA), pp. 1137–1144, Association for Computational Linguistics, 2008.

[10] J. Attenberg, P. Melville, and F. Provost, *A Unified Approach to Active Dual Supervision for Labeling Features and Examples*, pp. 40–55. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[11] E. Heim, M. Berger, L. Seversky, and M. Hauskrecht, "Active perceptual similarity modeling with auxiliary information," *arXiv preprint arXiv:1511.02254*, 2015.

[12] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, (Valletta, Malta), pp. 45–50, ELRA, May 2010. http://is.muni.cz/publication/884893/en.

[13] F. Chollet, "Keras." https://github.com/fchollet/keras, 2015.

[14] K. G. Jamieson, L. Jain, C. Fernandez, N. J. Glattard, and R. Nowak, "Next: A system for real-world development, evaluation, and application of active learning," in *Advances in Neural Information Processing Systems 28* (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), pp. 2656–2664, Curran Associates, Inc., 2015.

[15] A. Wibisono, L. Rosasco, and T. Poggio, "Sufficient conditions for uniform stability of regularization algorithms," *Computer Science and Artificial Intelligence Laboratory Technical Report, MIT-CSAIL-TR-2009-060*, 2009.

[16] O. Bousquet and A. Elisseeff, "Stability and generalization," *Journal of Machine Learning Research*, vol. 2, no. Mar, pp. 499–526, 2002.

[17] S. Arora, Y. Liang, and T. Ma, "A simple but tough-to-beat baseline for sentence embeddings," *ICLR (under review)*, 2016.