Progress Report **Kenneth Sanders**
COMP 4800-011 7/7/17
Prof. Rumshisky

Abstract

Twitter has been a boon for social media, allowing for users from all over the world to share short tweets containing a wide range of ideas and opinions. Hashtags allow users to group their tweets in hashtags by placing a pound or hash sign (#) before any non-whitepace string. Tweets themselves theoretically are made up of a sequence of words, just like a sentence or phrase. Thanks to the Word2Vec algorithm proposed by Mikelov et al. [10], we may create word embeddings (multi-dimensional vectors to represent the meaning of words in relation to the other words in the vocabulary) fairly easily. It seems readily apparent that one could model a tweet by averaging the embeddings for the most important words that are used in the tweet. Once tweet embeddings are created, we may then also average the most relevant tweets that are used with a hashtag in order to generate a hashtag embedding. Hashtags and their embeddings may work as indicators of conflict and polarization since their meaning can be derived from the meaning of the words they are used with. In summary, generating a graph of relationships between a set of hashtags may help gain insight into the polarity of conflict on Twitter.

Hypothesis

Hashtag embeddings can be created through a specialized summation of word embeddings generated from the set of tweets in which the hashtags were used. Clustering and analysis of the relationship between these hashtag embeddings will be another useful tool in analyzing polarization and conflict on Twitter.

Theory

This theorized method assumes the collection of a large number of tweets and their associated hashtags. They will be used to create a graph representing the similarity relationship between each hashtag on a scale of -1 to 1 (-1 is high dissimilarity, 1 is high similarity). Given this set of tweets, each tweet will contain text and a set of zero or more hashtags that were used with the tweet. Each tweet is of course made up of a set of words. Additionally, we may model a hashtag as being made up of a set of tweets that users ascribed with the hashtag.

The words can be modeled as vectors called word embeddings by training a shallow neural net with target-context pairs (a pair of words that co-occur in the same tweet). This procedure is well documented as Word2Vec by Mikolov et al. [10], and a slightly modified algorithm produces consistent and accurate word embeddings for our dataset. These embeddings allow us to use cosine similarity (the dot product of two vectors divided by the product of their magnitudes) to estimate the similarity between the two words that

the embeddings represent. The goal of this project is to bring that same functionality to hashtags by creating embeddings to represent them well.

In theory, we may sum the embedding for every word that is used in a tweet to make a vector representing a tweet embedding, and then sum the embeddings of all the tweets that are used with any given hashtag to create and embedidng for that hashtag. Additionally, we may weight each embedding before summing so that tweets and words more relevant to the hashtag will have greater impact than the ones that are not relevant.

More formally, we are given a vocabulary W and set of embeddings $E_W = \{E_w \text{ for } w \in W\}$ For any hashtag h, we can say there exists a set of tweets T_h in which the hashtag was used ($\forall t \in T_h$, h was used in t). We may also say that every tweet $t \in T$, $t = [w_1, w_2...w_n], w_i \in W$. We may attempt to construct a tweet embedding in a similar way to Le and Mikolov's Paragraph2Vec model [6] by summing the word embedding of each word used in a tweet T. In terms of w, h, and t, this is

$$E_t = \sum_{w \in t} F_h(E_w)$$

where F_h is a function that attempts to apply a scalar weight to word embedding E_w based on the importance of the word w in tweet t. In our current model, specificity [15] is used to measure how specific a word is when used with a hashtag than when it is used normally in the data set. For instance, one would expect "defund" to have a relatively high specificity in regards to the hashtag "#defundpp". In other words, since "defund" is very likely to be used with "#defundpp", the word embedding for "defund" should have a greater scalar multiple in order for the "#defundpp" embedding to be skewed in its direction.

Once we have a set of tweet embeddings $E_T = \{E_t \text{ for } t \in T\}$, we may construct a hashtag embedding

$$E_h = \sum_{t \in T_h} G_h(E_t)$$

where $T_h \subset T$. In this case, weight function G_h applies a scalar weight to the tweet embedding E_t based on its relevance to the hashtag h. One form of this weight could be the consistency of the tweet creator's vocabulary in relation to the current tweet, which may be a primitive measurement of a user's authority on the topic. In the projects current form however, this weight is not used, instead opting for a function that reduces each vector to a unit vector (vector magnitude = 1) in the same direction as the original vector.

Methodology

The creation of word embeddings and subsequent hashtag embeddings is a multistep process, which can be divided into functions and run by a main. Each of these steps is described below.

Database

Since a relatively large amount of data (2,114,926 tweets) is being processed in $O(n^2)$ time, there is a need to store the product of each step of the process in order to make

improvements to the algorithm and allow for analysis of intermediate data, such as the word embeddings and the vocabulary dictionary. Therefore, intermediate data is stored in PSQL tables by its producer, and accessed by its consumer accordingly. A default configuration of each table needed in the database is also provided, and schemas may be used to create multiple intermediate and final datasets.

Dataset

The data set used is a collection of over 2,100,000 tweets and 23,000 tweets that were gathered by the Text Machine Lab at Umass Lowell under the supervision of Prof. Anna Rumshisky. The data is rather raw, and was gathered through the use of (ideally) neutral keywords on the topic of abortion through the Twitter API [13]. However, the neutrality of the data is not guaranteed, since there may be underlying functionality in Twitter's keyword search that is not publicly disclosed, amongst other possible biases. The supplied data set also includes topics of taxes and politics, however these topics were not used yet with the procedure.

Splitting and Stemming

Initially, a tweet is a string of up to 130 characters and a unique long integer ID. To be able to create word embeddings, the tweet text must be split into individual words. The NLTK (Natural Language Tool Kit)[7] Python library offers a fairly comprehensive tweet tokenizer, which simply returns an array of strings representing words and symbols. Next, stop words and certain symbols are removed, and a snowball stemming algorithm from NLTK is applied to each word in a tweet in order to reduce each word to its root. Once the text of each tweet is tokenized, filtered and stemmed, the array of strings is inserted into a formatted-tweets table in the database using the tweet ID as an index.

Constructing the Dictionary

Once each tweet is reduced to an array of stemmed words, we can iterate over all the words of all the tweets in the data set and count the number of occurrences of each word. The most common words are mapped to unique integer IDs and inserted into a dictionary table in the database. The number of occurrences of each word is also included in the dictionary table, as it will be useful in generating hashtag embeddings. Any words that are not a part of those added to the database are considered to be unknown and are assigned the ID 0 for the next step.

Integerize Tweets

Once the dictionary table is created, the process of converting the tweets from an array of strings format to an array of integers begins. This is done by searching for each word and its corresponding word ID in the dictionary. If the word is not found, it is replaced with 0 instead of a positive integer ID. The tweet IDs and integer arrays are then inserted into an integer-tweets table in the database.

Creating Word Embeddings

Generating word embeddings once the tweets are integerized and the dictionary is created is fairly straightforward. This is outlined by Mikolov et al. [10] and is easily implemented using the Tensorflow library [1]. We use the skip-gram model of Word2Vec since it scales well with the varying sizes of data sets and does not treat the entire sentence as the context. In our model, each tweet is modeled as a sentence in the training for the neural net, meaning that each target-context pair is tweet specific, which helps maintain the sentiment of each individual tweet. Figure 1 shows the embeddings of the top 100 most common words in the data set. In theory the distance should represent the similarity of the words to each other. However, since the embeddings are reduced from 128 dimensions to 2, this may not always be accurate.

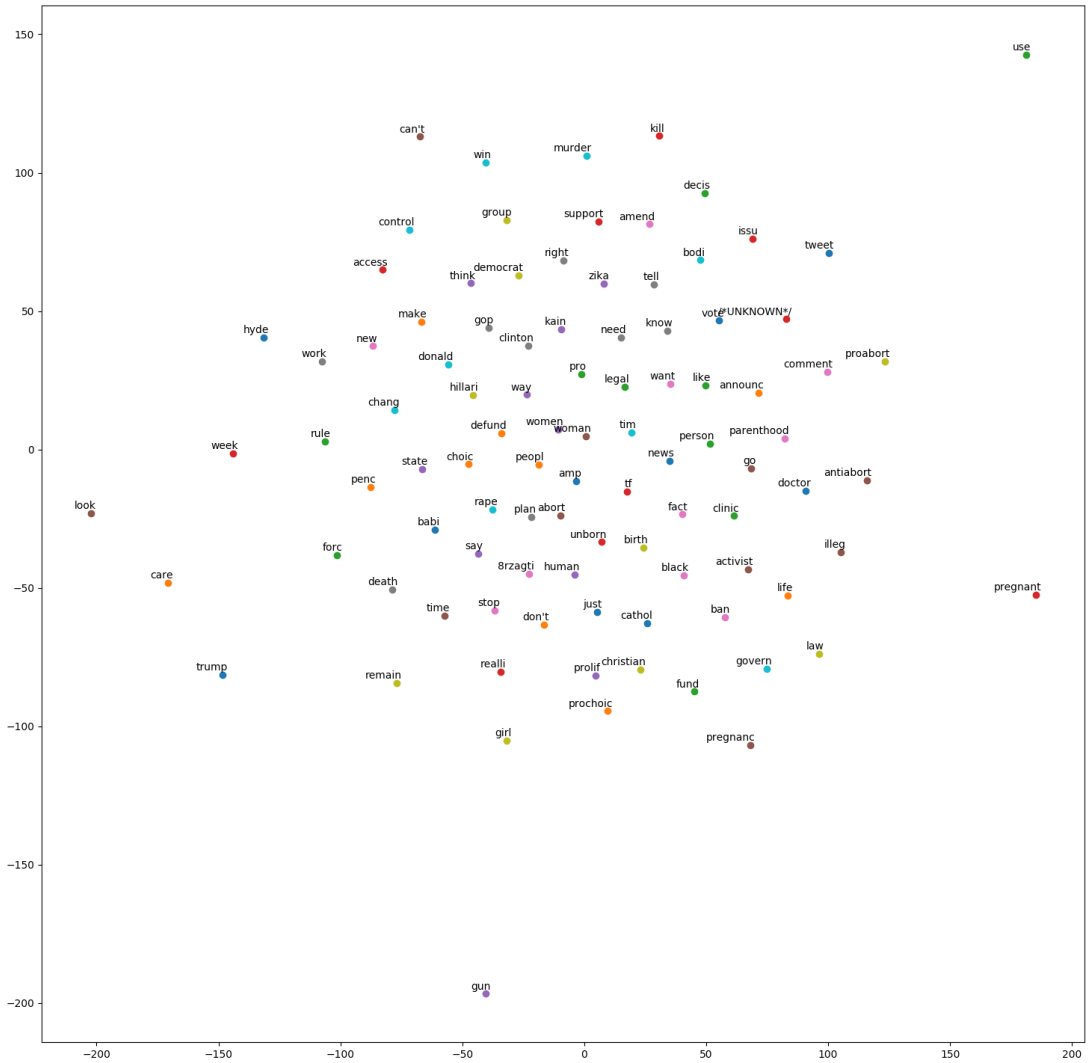


Figure 1: The 100 most common word roots graphed by their dimension reduced embeddings.

Generating Hashtag Embeddings

After the word embeddings are created, they can be combined into tweet embeddings, and then hashtag embeddings. The tweet embeddings created are temporary, allowing

the specificity of the word to the hashtag in relation to the whole dataset to be used as a weight when building the hashtag embedding. Therefore the tweet embeddings are not stored in the PSQL database, as they are hashtag specific, and they would be a multiple of the size of the tweet database. Once the tweet embeddings for a hashtag are created, they are unitized (reduced to magnitude = 1 in the same direction) in order to ensure that each tweet begins with the same weight. In the current model, the unitized tweet embedding for each tweet in a hashtag set is simply summed together with no weight to create the hashtag embedding. Therefor, these weights have the potential to improve results easily, assuming an accurate and consistent weighting method can be found.

Creating a Hashtag Graph

Once hashtag embeddings are created, we may calculate the cosine similarity between each possible pair of embeddings in order to get the relationship between each pair of hashtags. These relationships can be expressed as a square matrix or a list of n dimensional points, where n = the number of hashtags. This hashtag graph allows each hashtag to act as a dimension, and the relationship it has to all the other tweets will be the value of that dimension in all the other hashtags in the graph.

Clustering and Graph Analysis

After a hashtag graph is created, we may attempt to analyze the accuracy of the embeddings for use in polarization analysis. The graph is clustered using a k-means ($k = 2$) algorithm [5] provided by pyclustering [11]. K-means is a simple centroid-based clustering algorithm that aims to minimize the within-cluster variance. This algorithm suffices for now, although it is certainly an area to be improved upon. Once the clusters are generated, the within-cluster standard deviation is calculated and the hashtags that belong to each cluster are recorded, along with the size of cluster, standard deviations, and distance between centroids.

Results

So far, I have run my program on the set of abortion tweets split by week and with all 10 weeks combined. The produced hashtag graph is clustered and analyzed, producing in hashtag clusters. These clusters can be interpreted using the reference hashtags manually picked to be representative of the issue being focused on. The reference hashtags and the clusters they are grouped in can be noted in Figure 2. The expected and ideal results would be for absolutely all of the hashtags in the pro-life group to belong to the opposite cluster from the pro-choice hashtags. This never occurs in the reference hashtags, but this is not surprising given how volatile the meanings of hashtags are, especially when dependent only upon a week-long interval. The main notable feature in Figure 1 is that there is a distinct (but not absolute) polarization between pro-life and pro-choice hashtags. Specifically, in weeks 2, 5 ,6, 7, 9, and 10 the majority of pro-life hashtags are in the opposite cluster from the majority of pro-choice hashtags. While these polarized majorities hardly indicate success, they are promising considering that a random distribution should contain very few of these occurrences. Additionally, none of

Week		1	2	3	4	5	6	7	8	9	10	All
Prolife Hashtags	defundpp	1	0	1	0	1	1	0	1	1	0	1
	praytoendabortion	1	0	1	0	1	1	0	1	1	0	1
	prolife	1	0	1	0	1	1	0	1	1	0	1
	ppsellsbabyparts	0	0	1	0	1	1	0	1	1	0	1
	stopabortion	0	1	0	1	1	1	1	1	1	0	0
Neutral Hashtags	abortion	1	0	1	0	1	1	0	1	1	0	1
	pregnant	0	1	0	1	0	0	1	0	0	1	0
	healthcare	0	1	1	0	0	0	1	1	0	1	0
	medical	0	1	0	1	X	0	1	0	0	1	0
	health	1	0	0	1	0	1	0	1	1	0	0
Prochoice Hashtags	prochoice	1	0	1	0	1	1	0	1	1	0	1
	antichoice	0	1	1	0	0	0	1	1	0	1	0
	womensrights	1	1	0	1	0	1	1	0	0	1	0
	mybodymychoice	0	1	0	1	0	0	1	0	0	1	0
	fundpp	1	0	1	X	X	0	X	1	X	0	1

Figure 2: Reference hashtags and their resulting clusters for each week of data. X symbolizes that the hashtag was not found that week.

the neutral hashtags have a ratio higher than 3:1 (aggregate run excluded), indicating that the validity of the clusters remains.

In addition to the reference hashtags, much can be said of the original clusters themselves. Figure 3 shows the sizes and standard deviation of the two clusters formed from the hashtag graph, as well as the distance between them. By taking the squared sum of error for the clusters, we can approximate a distance between the centroids of the clusters relative to the normal probability density of them. Mathematically, we may write N_1 and N_2 to represent the sizes of each cluster and $\sigma(C_i)$ to represent the standard deviation of cluster i. Then the weighted average standard deviation (WASD) will be

$$s(C_1, C_2) = \frac{(N_1\sigma(C_1)+N_2\sigma(C_2))}{(N_1+N_2))}$$

and the relative distance will be quotient of the distance (D) over the WASD, or

$$r(C_1, C_2, D) = \frac{D}{s(C_1,C_2)}.$$

We may note that the relative distance is fairly consistent in the range of 9 to 20, and we may even note a peak in debate during week 6 (7/29/16 - 8/5/16) which followed controversial remarks from U.S. Vice Presidential Candidate Tim Kaine [3][8]

Lastly, we may reduce the hashtag relationship dimensions so that they may be visualized. This is achieved using T-distributed Stochastic Neighbor Embedding (TSNE) [14]. This allows our hashtag graph to be expressed as a set of two dimensional points for each hashtag, which subsequently allows for the hashtag relationships to be plotted on paper. These dimension reduced hashtag graphs can be noted in Figure 4. It should be noted that much of the information is lost in the TSNE reduction, since the data

is reduced from about 5000 dimensions (depending on number of hashtags in dataset) down to 2. However, there are still clearly defined and consistent clusters despite this loss, which is very promising for the validity of the hashtag relationships. Notable features of these graphs are the consistent "weaker" arm on the right side, as well as far outliers at the top. It also appears as if week 6 was an especially abnormal week considering the extensive growth of the left cluster, which agrees with the data in Figure 3. It is important to note that the swapping of colors between the clusters by week is irrelevant, as this is only based on the number assigned to the cluster, and the hashtags remain in the same clusters together for the most part. Additionally, this is independent data from separate weeks (subfigure K excluded), and the consistency between weeks is a very promising indicator of accurate hashtag embeddings.

Week	# of Tweets	C_1 size	$\sigma(C_1)$	C_2 size	$\sigma(C_2)$	D	$s(C_1, C_2)$	$r(C_1, C_2, D)$
1	161,537	6,665	2.61	3,335	2.53	23.36	2.58	9.04
2	150,758	1,902	2	4,290	0.77	11.83	1.15	10.31
3	182,649	4,180	0.77	1,939	2.12	11.87	1.20	9.91
4	170,366	1,752	1.75	3,873	0.69	11.27	1.02	11.05
5	106,860	3,926	0.67	1,678	1.66	11.67	0.97	12.08
6	150,406	4,853	0.16	497	0.86	4.31	0.23	19.15
7	478,441	893	0.93	3,141	0.55	9.5	0.63	14.98
8	141,943	3,734	0.72	1,582	1.76	11.71	1.03	11.37
9	180,676	3,777	0.99	1,820	2.02	14.26	1.32	10.76
10	165,554	1,706	1.86	3,808	0.66	10.85	1.03	10.52
All	2,114,926	5,934	1.66	4,066	2.47	20.89	4.12	5.08

Figure 3: Results of clustering the generated hashtag graph for data on the topic of abortion (weeks of 6/24/16 - 9/2/16 and their aggregate)

Future Tasks

Unfortunately, the clusters created are not as consistent in size as one would expect. This could be attributed to the volatility of Twitter or the data collection methods. However, not having a comparison methodology to compare to certainly prevents any conclusions from being made. Therefore, one of the future focuses of this project should be finding a comparable methodology.

For the results analysis section, I also plan to improve both my clustering and graphing algorithm in order to account for the highly variable density of the hashtag relationships, since this data property seems to be a major issue for the k-means algorithm. Instead, I will be looking into density based clustering algorithms such as DBSCAN [4] or OPTICS [9], which will hopefully lead to better defined clusters and the exclusion of outlier relationships. Additionally, a better idea for a polarization measure may be to use the cosine similarity between the cluster centroids instead of the euclidean distance. Lastly, when graphing embeddings and relationships, the TSNE algorithm used for embedding reduction is known not to be optimal for high dimension embeddings. This can be improved

using other dimension reduction methods such as Principle Component Analysis (PCA) [2] and Truncated Singular Value Decomposition [12]

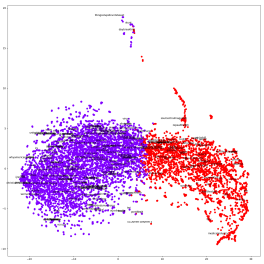
Lastly, another focus of this project will be further documentation, optimization, overall improvement of code. Additionally, making this library portable to other database structures would be a worthwhile endeavor.

Schedule

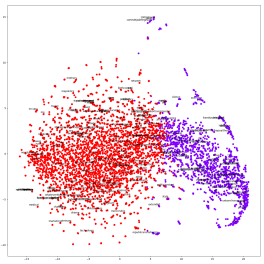
Week	Task	Completed
5/14/17	Literature analysis	✓
5/21/17	Plan procedure and outline code	✓
5/28/17	Build code utilities	✓
6/4/17	Build main code functions	✓
6/11/17	Apply code to topics and analyze accuracy	✓
6/18/17	Revisit program and procedure for improvements	✓
6/25/17	Generate results	✓
7/2/17	Progress Report	✓
7/9/17	Work on noted improvements	
7/16/17	Optimization, documentation, and cleanup	
7/23/17	Finalize code and regenerate results	
7/30/17	Analyze improvements and make conclusion about hypothesis	
8/6/17	Work on paper (update progress report)	
8/13/17	Finish paper	

Conclusion

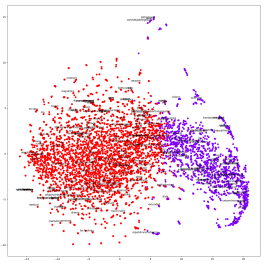
In conclusion, the project has made notable progress with promising results. The following semester will include many optimizations and improvements for the methodology and results analysis, and hopefully a way of better measuring effectiveness of the procedure.



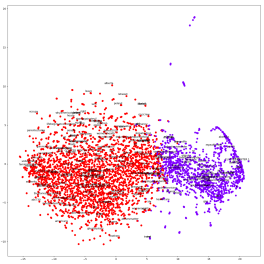
(a) Week 1



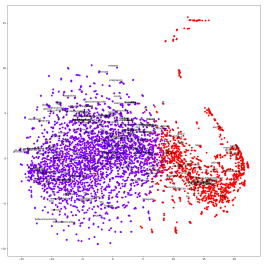
(b) Week 2



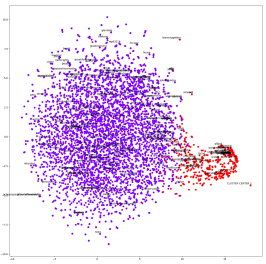
(c) Week 3



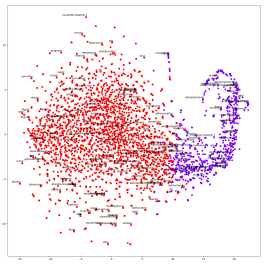
(d) Week 4



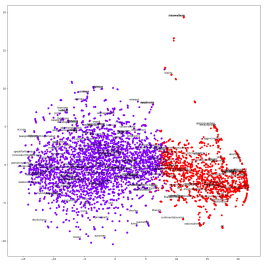
(e) Week 5



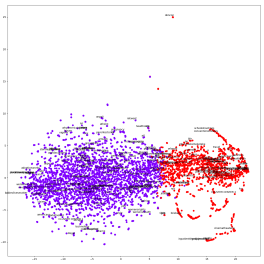
(f) Week 6



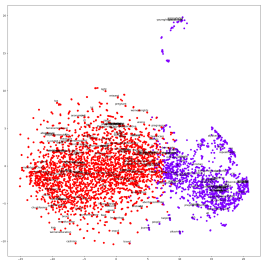
(g) Week 7



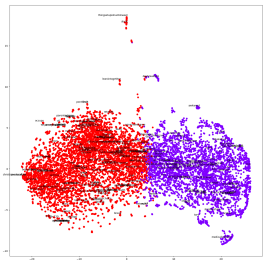
(h) Week 8



(i) Week 9



(j) Week 10



(k) All data

Figure 4: Dimension reduced graphs of abortion hashtag embeddings by week (and all data combined)

References

[1] Abadi, Martn, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Man, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Vigas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, (Version 1.2) 2015. Web. 7 June 2017.

[2] Abdi. H., & Williams, L.J. (2010). "Principal component analysis". Wiley Interdisciplinary Reviews: Computational Statistics. 2 (4): 433459. Web. 6 June 2017.

[3] Davis, Susan. "The Policy Split Between Clinton And Kaine On Abortion, Explained." NPR. NPR, 01 Aug. 2016. Web. 07 July 2017.

[4] Ester, Martin, Hans-peter Kriegel, Jrg Sander, and Xiaowei Xu. "A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". N.p.: AAAI, 1996. 226-31. Web.

[5] Hartigan, J. A., and M. A. Wong. Algorithm AS 136: A K-Means Clustering Algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, 1979, pp. 100108. Web. 5 June 2017.

[6] Le, Quoc V., and Tomas Mikolov. "Distributed Representations of Sentences and Documents." CoRR 4053rd ser. Abs.1405 (2014): n. pag. Web. 7 June 2017.

[7] Loper, Edward and Steven Bird. 2002. "NLTK: the Natural Language Toolkit". In Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics - Volume 1 (ETMTNLP '02), Vol. 1. Association for Computational Linguistics, Stroudsburg, PA, USA, 63-70. Web. 6 June 2017.

[8] Mellen, Ruby. "Kaine breaks with Clinton on abortion provision." CNN. Cable News Network, 01 Aug. 2016. Web. 07 July 2017.

[9] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jrg Sander. 1999. "OPTICS: ordering points to identify the clustering structure". In Proceedings of the 1999 ACM SIGMOD international conference on Management of data (SIGMOD '99). ACM, New York, NY, USA, 49-60. Web. 7 June 2017.

[10] Mikolov, Tomas and Sutskever, Ilya and Chen, Kai and Corrado, Greg S and Dean, Jeff. Distributed Representations of Words and Phrases and their Compositionality, *Advances in Neural Information Processing Systems 26*. Eds. C. J. C. Burges and L. Bottou and M. Welling and Z. Ghahramani and K. Q. Weinberger Curran Associates, Inc., 2013, 3111–3119. Web. 7 June 2017.

[11] Novikov, Andrei. Pyclustering Library (Version 0.6), 2017 Web. 7 June 2017.

[12] Per Christian Hansen. 1987. "The truncated SVD as a method for regularization". BIT 27, 4 (October 1987), 534-553. Web. 5 June 2017.

[13] "Rust API." Twitter. Twitter, n.d. Web. 07 July 2017.

[14] van der Maaten, L.J.P.; Hinton, G.E. (Nov 2008). "Visualizing High-Dimensional Data Using t-SNE". Journal of Machine Learning Research. 9: 25792605. Web. 06 July 2017.

[15] Zhang, J. , W. Hamilton, C. Danescu-Niculescu-Mizil, D. Jurafsky, and J. Lescovec. "Community Identity and User Engagement in a Multi-Community Landscape." AAAI International Conference on Weblogs and Social Media (ICWSM) (2017). Web. 07 July 2017.