ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Distributed Information Systems Laboratory
LSIR

Semester project

Inducing human-interpretable rules for automated document classification

Author:
Alain Milliet

Supervisors:
Prof. Karl Aberer
Rémi Lebret

January 19, 2018

Abstract

Neural networks often give really good results in natural language processing for regression and classification problems, especially for large dataset in large dimension [1]. Yet, we cannot use their results in certain situations such as in the medical or the law domain because they only give prediction and not human-understandable features on how it computes the classification from the input to the output. In this work we give an example of pipeline to obtain simple rules from neural network decisions, in order to get a method that can be generalized to multiple classification problems. In our case the classification problem is about categorizing tweets into groups with opposing polarity, e.g. on a political issue, during the 2016 US Presidential Election.

Contents

1 Introduction

Experts prefer using their experience and intuition to solve their problems instead of using statistical analysis. Lots of examples of this behaviour can be found in different sectors of the society, such as doctors in medicine or judges in law domain. The result of this type of decision has been found to often give inferior outcomes as the one using statistical models [2, 3]. Still, these neural networks' methods are not yet broadly used because of the difficulty of creating and applying them. Here we tried to develop a method to surpass these difficulties and to make these statistical methods easier to understand and to use. We created a method to classify tweets into 2 different groups; at first, we used simple rules, e.g. the use of characteristic hashtags, to derive our initial class labels. The next step was to get some baseline results to know what kind of outcome we could expect for this problem. We then used the labeled corpus to train a neural network using n-grams[1] to detect which patterns defined classes the most effectively. Finally, we went through different techniques to visualize which decisions the neural network took and to have a better understanding of the clusters of words it created. These clusters and features can be seen as rules used for the classification.

1.1 Dataset used for the project

The data used for the project concerned the US 2016 Presidential Election. It contains documents created on different platforms such as Web, Instagram and Twitter. The database contains different tables that were used to build the initial dataset. A table *document* contained all the documents (Instagram posts, tweets and web docs). We decided to keep only the tweets (6'672'171 tweets) as we used hashtags to first classify our data. Knowing the result of the election, we decided to label our dataset in two classes: Trump's supporters vs Trump's enemies. We gave a score (*trumpscore*) to users in function of their utilization of certain hashtags in their respective tweets, creating a *ranking* table (See original lists of hashtags used to classify the users in Appendix A).

[1] n-grams = group of n consecutive words

1.2 Ranking of the tweets in classes

With the data given, we decided to label each tweet using the ranking table. To do so we mapped the *trumpscore* of every user to all the tweets she/he created. From there we decided to label as 1 all the tweets that have a positive *'trumpscore'* and as 0 the ones that have negative *'trumpscore'*. Here, around 3'000'000 tweets are lost as they do not have any *trumpscore*, meaning that the publisher either is not in the ranking table or did not use any of the previously quoted hashtags.

2 Baseline algorithms

Before going through more complex classifiers, we decided to use some simple machine learning algorithms as a baseline. This allows us seeing what accuracy we could achieve and what were the most important words that would be detected for each group.

2.1 Preparation of the data

First of all, we had to do some upsampling on the data to get the same amount of pro-Trump and anti-Trump tweets. Initially, we had 1'183'061 anti-Trump tweets and 2'289'111 pro-Trump tweets. However, as we detected the same number of users for each group (N=45'000), the Trump supporters (as we classed them) seem to tweet more than people not supporting him. For the cleaning of the tweets, we decided to remove all the URLs that were part of the texts as it is not interesting for the result we want to get at the end. We also replaced the mentions and the hashtags to general strings (respectively every mention became *@mention* and every hashtag became *#hashtags*). We then tokenized the text and removed stopwords and symbols.

2.2 Corresponding results

For the baselines, we had to decide how we wanted to create the vocabulary used in the classifiers. In our case we wanted to find interesting patterns that define our classes and not necessarily obtain the best accuracy so we decided to take a minimum of 1'000 occurences of an unigram or a bigram to put it in the vocabulary. It is a good tradeoff between accuracy and interpretability.

As we are not looking for the best accuracy but only trying to find interesting patterns, that's why we kept either unigrams or bigrams but not both at the same time. The vocabulary contains then 2'537 entities for unigrams and 604 for bigrams, while if we took a minimum number of occurences of 1 we would have obtain a vocabulary of 379'446 entities for unigrams and 4'729'111 for bigrams. On Figure 1 is represented the different accuracies obtained using different classifiers and either using unigrams or bigrams.

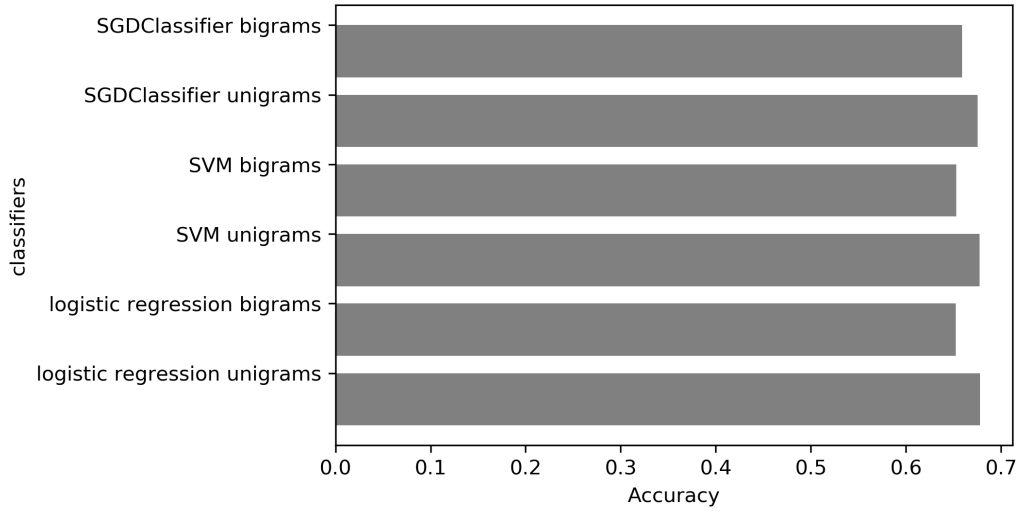

Figure 1: Accuracy of different simple classifiers

In the subsections below you will find what were the top most representative unigrams and bigrams with their corresponding score for their class in their respective classifier.

2.2.1 Logistic Regression

AntiTrump	Score
donny	-3.3375
donnie	-3.1571
trumpcare	-2.4846
microwave	-2.3307
bigly	-1.9508
emolument	-1.8886
orange	-1.7804
authoritarian	-1.7283
cruel	-1.7049
suppression	-1.7024

ProTrump	Score
yourvoice	4.9547
rinos	3.9646
islamization	3.8114
rino	3.6610
geller	3.5339
globalists	3.4285
marxist	2.9940
via	2.9108
globalist	2.8606
antifa	2.8582

Table 1: unigrams (accuracy = 0.6782)

AntiTrump	Score
voter suppression	-2.4769
alternative fact	-2.4467
pathological liar	-2.3468
tax break	-2.2304
release taxis	-2.1270
dem lawmaker	-2.0901
white nationalist	-2.0542
russia scandal	-2.0498
mar lago	-1.9331
blame obama	-1.9319

ProTrump	Score
geller report	5.0002
@mention app	4.6490
muslim migrant	4.6430
pamela geller	3.8547
open border	3.5592
muslim brotherhood	3.3002
breitbart @mention	3.2604
illegal alien	3.0194
via @mention	2.9912
muslim refugee	2.7925

Table 2: bigrams (accuracy = 0.6525)

2.2.2 Support Vector Classifier

AntiTrump	Score
donny	-0.9704
donnie	-0.9235
trumpcare	-0.7896
microwave	-0.7820
bigly	-0.6979
orange	-0.6457
emolument	-0.6363
authoritarian	-0.6241
cruel	-0.6186
yiannopoulos	-0.6162

ProTrump	Score
yourvoice	1.3469
rinos	1.1654
rino	1.1060
islamization	1.0709
globalists	1.0423
geller	1.0134
via	0.9705
marxist	0.9417
gloablist	0.9227
libs	0.9225

Table 3: unigrams (accuracy = 0.6776)

AntiTrump	Score
voter suppression	-0.8417
pathological liar	-0.8175
alternative fact	-0.8150
tax break	-0.7773
dem lawmaker	-0.7518
release taxis	-0.7512
white nationalist	-0.7349
blame obama	-0.7267
mar lago	-0.7216
trump lawyer	-0.7119

ProTrump	Score
@mention app	1.1861
geller report	1.1503
pamela geller	1.0631
withhold germany	1.0352
breitbart @mention	0.9937
muslim migrant	0.9675
open border	0.9505
muslim brotherhood	0.9430
via @mention	0.9209
illegal alien	0.9151

Table 4: bigrams (accuracy = 0.6529)

2.2.3 SGD Classifier

AntiTrump	Score
trumpcare	-1.1458
donnie	-1.1404
donny	-1.1136
orange	-1.0612
bigly	-1.0350
thread	-1.0268
45	-1.0109
democracy	-0.9973
aca	-0.9825
fuck	-0.9803

ProTrump	Score
app	1.7106
leftist	1.5582
msm	1.5358
soros	1.5009
libs	1.4233
liberal	1.3654
maxine	1.1939
globalist	1.1655
geller	1.1600
hillary	1.1589

Table 5: unigrams (accuracy = 0.6756)

AntiTrump	Score
@mention lie	-1.1890
white supremacist	-1.1890
@mention gop	-1.1863
donald trump?s	-1.1846
mar lago	-1.1835
pathological liar	-1.1835
health care	-1.1830
trump lawyer	-1.1830
trump lie	-1.1830
trump attack	-1.1824

ProTrump	Score
@mention app	1.8018
breitbart @mention	0.9377
video @mention	0.9203
illegal alien	0.9061
geller report	0.8946
mt @mention	0.8733
@mention android	0.8525
god bless	0.8504
news channel	0.8454
muslim brotherhood	0.8438

Table 6: bigrams (accuracy = 0.6592)

2.3 Comments on the entities found

The entities found in the results are sometimes noisy and not really representative (for example: *rt @mention*, *w/ @mention*, *j.*, etc..) but we can also find unigrams and bigrams that surely represent the support or the opposition to Donald Trump. For the anti-Trump movement, we can find entities like *white supremacist*, *orange* and *trump lie* that are directly related to the opposition. For the pro-Trump movement, we also have certain entities that make sense, such as *yourvoice radio* which is the radio designed to support Trump, and which appears at the top of the most significant features for this group. It also seems like the data for the supporter of Trump is more noisy as we have a lot of these *@mention* in the top entities (*@mention app*, *@mention video*, *mt @mention*). It can be explained by the popularity of the hashtags we used for the pro-trump group that can lead to more noise.

3 Neural network

Now that we obtained some results thanks to basic machine learning algorithms, we want to see what accuracy we can get using a neural network. In this section, the neural network used for this classification task is presented, as well as the parameters used in the experiment and the results obtained.

3.1 Description of the model

The model is constituted of an embedding layer where the words are transformed into vectors ($\mathbb{R}^{V \times D}$ where V is the number of words and D is the word embedding dimension). It is followed by a convolutional layer with multiple filter width and feature maps ($\mathbb{R}^{D \times Co \times K}$ where D is the word embedding dimension, Co the number of each kind of kernel and K the size of the current kernel). We then used a dropout layer such that during training, it randomly zeroes some of the elements of the input tensor with a certain probability using samples from a Bernoulli distribution to regularize and prevent the co-adaptation of neurons (with probability p). At the end, a linear layer applies a linear transformation (without bias) to the incoming data ($\mathbb{R}^{Co \times C}$ where Co is the number of each kind of kernel and C the number of class).

3.2 Parameters used in the experimental setup

After testing a lot of different set of parameters, the most efficient setup that we kept for the rest of the experiment was:

- Number of embeddings (V): $102'292$ (we only took words that appeared at least 10 times in the corpus)
- Dimension of embeddings (D): 100
- Number of classes (C): 2
- Number of each kind of kernel (Co): 1000
- Kernel sizes (Ks): $3, 4, 5$
- Dropout probability (p): 0.25

3.3 Preparation of the data

Contrary to the data preparation of the tweets for the baseline algorithms, the stopwords and other symbols are important here, as well as the words in their integrity. We just did tokenization using spacy [4] and replaced entities such as user mentions, hashtags and URLs to *<user>*, *<hashtags>* and *<url>* respectively. As the goal is to use the whole tweets to learn the corresponding mapping of n-grams to a certain group or the other, the text is kept as original as possible.

3.4 Corresponding result

With this setup we obtained an accuracy of classifying tweets to the correct label of 79.8%. This accuracy is greater than every other methods we used in the baselines.

4 Anatomising N-grams learnt from neural networks

With the results obtained, we wanted to find which n-grams where the most important for our neural network to do the classification. For each n-gram we took its embedding, passed it through the convolutional layer and applied the rectified linear unit to obtain numbers between 0 and 1. We then passed it through the linear layer and applied the SoftMax function such that our n-dimensional input Tensor is rescaled and the elements of the n-dimensional output Tensor lie in the range (0, 1) and sum to 1. Thus we obtained two scores between 0 and 1 for each n-gram, the first score corresponding to its importance for the pro-Trump group and the second one for the anti-Trump group. We kept all these n-grams and started looking for interesting patterns in the different groups. First of all we decided to look at the proportion of types of n-gram at each score level (Figure 2).

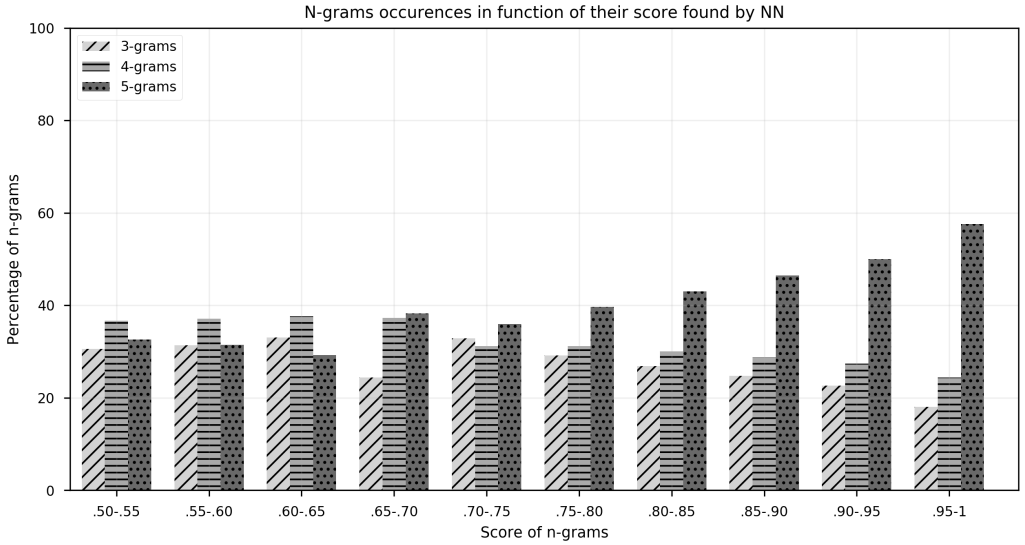

Figure 2: Proportion of each type of n-gram in function of its score level

On Figure 2, there is nearly the same percentage of 3-grams, 4-grams and 5-grams in the n-grams with low importance. But as we go to more important n-grams it appears that the percentage of 5-grams goes up while both the 3-grams and 4-grams are going down. It can be understood as the more

complete the n-grams are the most efficient they are for the classification.

4.1 Clustering algorithms used

As we want to find insights that would explain the decision of the neural network, we decided to look for topics in the important n-grams. We explored two different methods to see which of them fits best our task, namely K-means and Latent Dirichlet Allocation (LDA). For these two methods we took only the most significant n-grams (with score between 0.95 and 1).

4.1.1 K-means

First method we used was K-means, it permits to decide of the number of centroids we want to find in the data and then compute to which centroid each n-gram belongs. We used a library, developed by Facebook AI Research, called Faiss [5] that permits to compute efficient similarity search and clustering of dense vectors. With this method we could not create a good representative graph where we could visualize the centroids as the method does not give the position of the centroids but only the correspondence between n-grams and centroids. Visualization of the results of this method could be part of future works.

4.1.2 Latent Dirichlet Allocation (LDA)

The second method we used to find topics in our data is LDA. Given a number of topic n, this method permits to find words that describe each of the n topics. We used this method with t-Distributed Stochastic Neighbor Embedding (t-SNE) [6] to have a clear visualization of the result. LDA works directly on the textual form, so in this case we used the n-grams directly and not their embedding representation. We thus combined LDA with 100 topics with t-SNE on the most significant n-grams to obtain Figure 3.

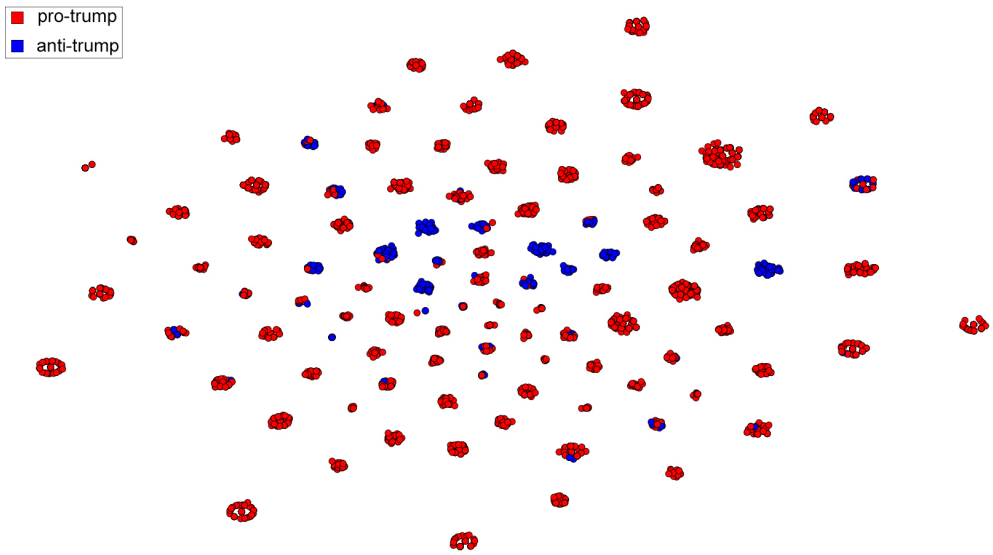

Figure 3: t-SNE visualization of 100 topics for most important n-grams (pro-Trump n-grams represented in red, anti-Trump n-grams represented in blue)

We can clearly see some clusters on Figure 3 and we can already see some of them being mostly pro- or anti-Trump. This kind of clusters can tell us what subject the classes are talking about and can be used as rules found thanks to our neural network. More than just seeing what subjects the different classes are talking about, we wanted to see how differently people of different class talks about the same subject. This leads us to our next section on In-topic exploration 4.2.

4.2 In-topic exploration

From these topics we decided to take a topic containing both pro- and anti-Trump n-grams and we found the topic you can see selected on Figure 4 that is represented by the word "sater felix muslims". Felix Sater is the name of a Moscow-born businessman that was a former Trump associate. We decided to look for this and find differences between the expression of pro-trump and anti-trump people about this subject. This topic is defined by 62 n-grams with 32 of them being labelized as "pro-trump".

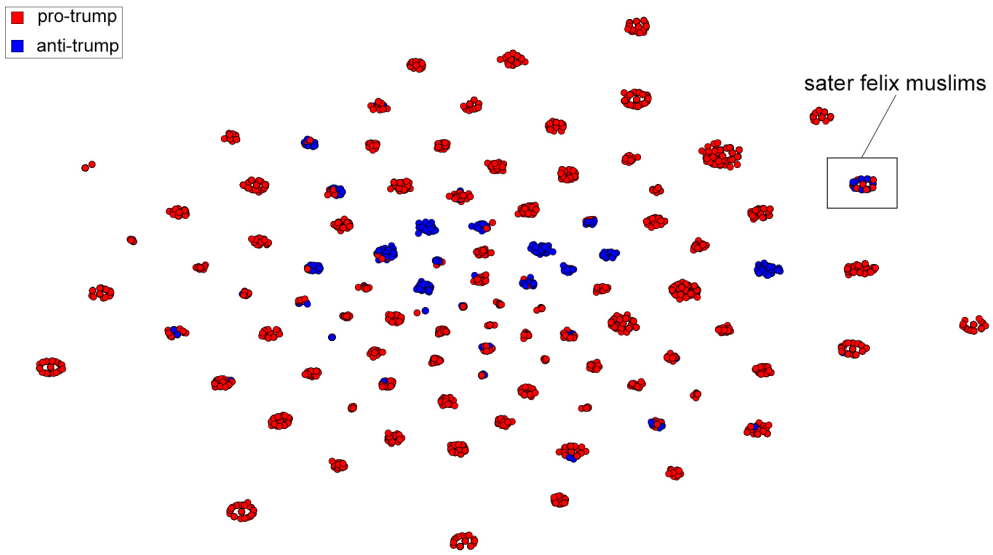

Figure 4: t-SNE visualization with position of topic concerning "sater felix muslims" (pro-Trump n-grams represented in red, anti-Trump n-grams represented in blue)

4.2.1 Word Mover Distance

To observe more specifically what is happening in the inside of a topic we decided to use *Word Mover Distance* [7] as it seems to be the best distance metrics for our research. The first idea with this metric was to create a new graph containing the n-grams of one topic and looking what clusters would be created with the distances obtained. We could not create a graph using directly the distances and the n-grams because for N n-grams the problem becomes N-dimensional and it does not necessarily have any solution in 2-D, which means the graph would not keep the distances as expected. We still managed to create a graph using Gephi [8] and its Force Atlas 2 algorithm such that we can visualize the different n-grams and the distance between each others. The computation of the word mover distance is pretty expensive but as we have only 62 n-grams we could do it on the whole topic. We denoted two different clusters in the resulting visualization (see Appendix B for full graph).

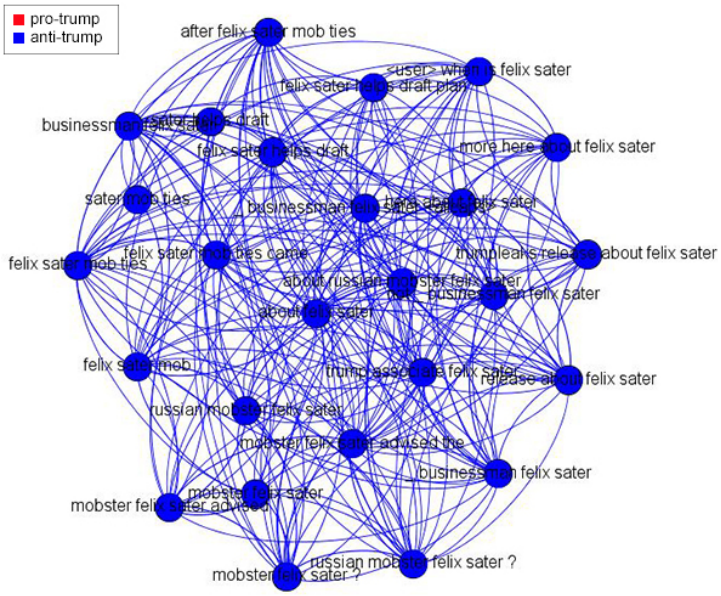

Figure 5: Word mover distance graph: Cluster 1 (pro-Trump n-grams represented in red, anti-Trump n-grams represented in blue)

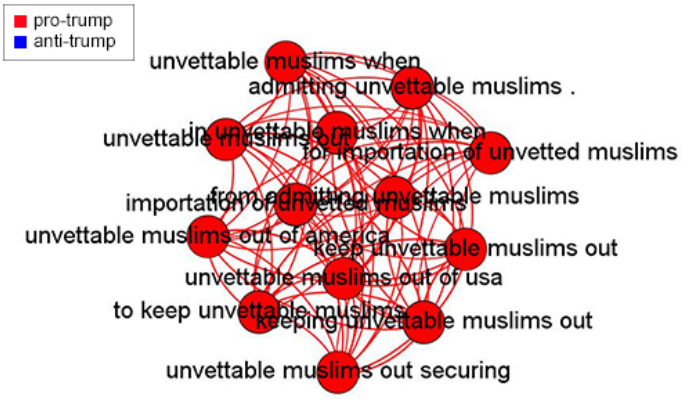

Figure 6: Word mover distance graph: Cluster 2 (pro-Trump n-grams represented in red, anti-Trump n-grams represented in blue)

Cluster 1 on Figure 5 contains n-grams with the word "felix sater". There are only anti-trump n-grams which could lead to a hypothesis that n-grams talking about "felix sater" are more likely to be anti-trump.

Cluster 2 on Figure 6 seems like a smaller cluster and it contains words like "unvettable muslims". This cluster only contains pro-trump n-grams.

Here, we demonstrated that by selecting precise words or precise topics, such as the "felix sater muslims" topic, we can cluster more or less precisely the tweets. This technique is really interesting and permits to see more in details how users talk about a specific topic. Again, this kind of clusters can be transferred to classification rules.

4.2.2 t-SNE

As seen before t-SNE is a machine learning algorithm for dimensionality reduction such that we can visualize vectors of N dimensions in 2-D for example. In this section we just applied t-SNE directly on the embedding of the n-grams contained in our topic.

Some small clusters can be found but this time they are less obvious as we do not see the links between the n-grams but only their position in function of their embedding. We can still find some different clusters. For example, at the bottom of Figure 7 we show nearly the same cluster as in Figure 6.
The advantage of this method is that we do not need to compute the distance between each n-grams but we just have to plot the output of the t-SNE in 2-D.

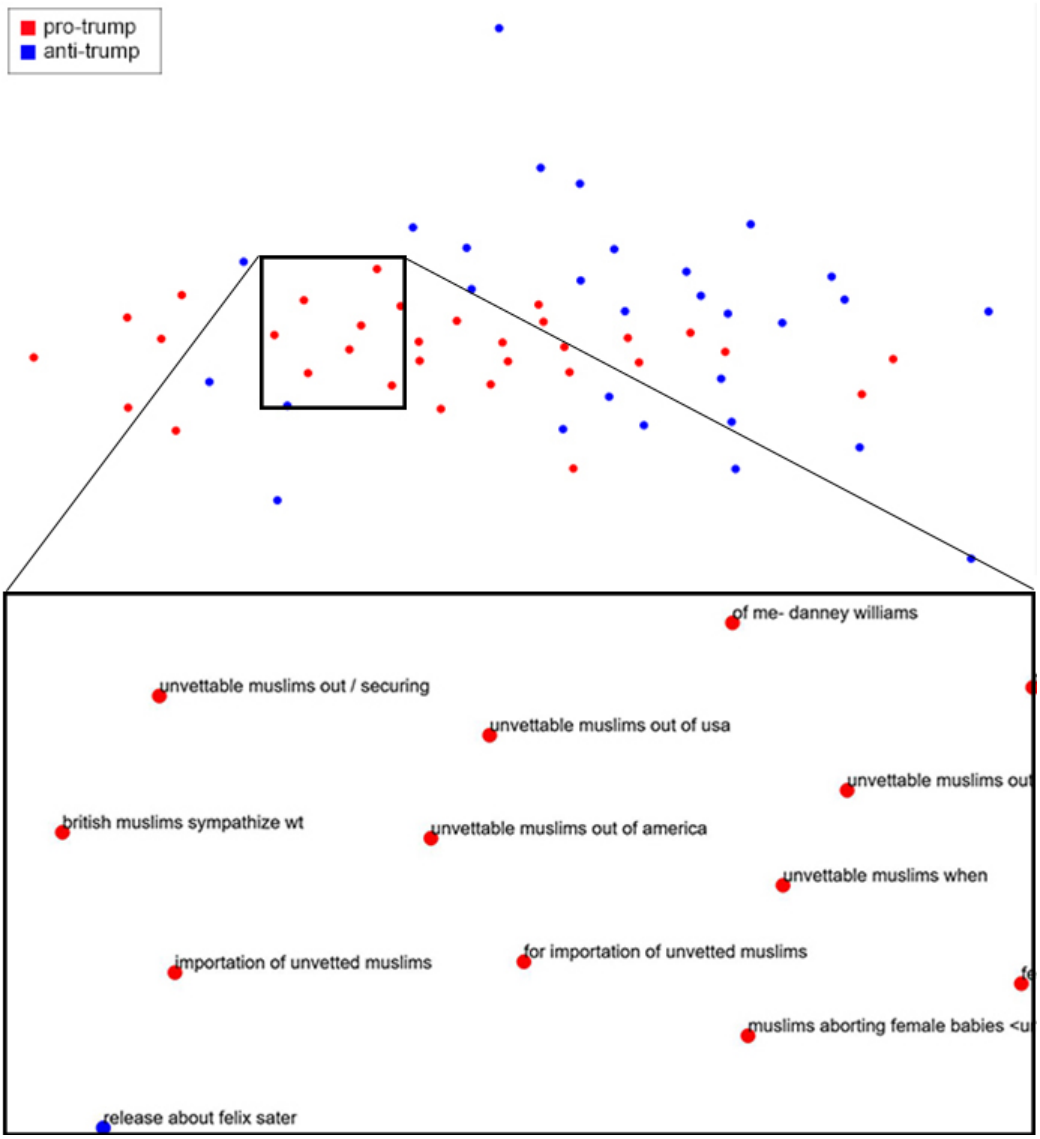

Figure 7: Top: t-SNE visualization in the topic "sater felix muslims" with 62 n-grams. Bottom: zoom on a cluster of pro-Trump n-grams containing words "unvettable muslims"

4.3 Feature extraction

In this section we will explain how to extract features directly from our neural network. At the last layer of our neural network we have a linear layer. This linear layer contains C vectors of size Co (See section 3.2 for more informations on these constants). These vectors contains weights that leads to the classification output of our neural network. From this layer, we looked for the biggest values for each of them. As we want to know what sentences activate them the most, we decided to look at the representation of all sentences of our dataset at this layer and look for the ones that have the greatest value at the indices found before. Once we found them we look at the corresponding n-gram that gave its weight to the sentence at this index and we keep this n-gram as an important one for the class (See Appendix C for a visual explanation of the feature extraction mechanism). Thanks to this techniques we obtained different patterns for both classes. (see Table 7)

Pro-Trump patterns
#jaylive listener
#bluntforcetruth
islamization via settlementjihad
lwc #dailycrowder

Anti-Trump Patterns
dworkingreport #trumpleaks
saveaca #hellervoteno
saveaca #protectourcare
chris matthews obliterates paranoid

Table 7: pro- and anti- Trump patterns

5 Conclusion and future work

In conclusion, this project leads to a pipeline that can be used to find insights on neural network classification based on tweets. This can be extended to the classification of any type of textual elements using the embedding of the words contained in these documents. We could find some particular simple rules and could test them against the neural network accuracy. This exploration of different techniques helped us derives some ideas for future work. An idea would be to continue working with the centroids of topics we found with LDA. From the affiliation of the n-grams to them we can derive some rules and if we found topics containing the two classes then we can use the word mover distance algorithm to again separate them. The current problem with this idea is that it is not automated and you have to look

manually for topics and rules. The other technique would be to derive rules from the linear layer of our neural network as we did in the last section. If we can obtain enough rules with all these techniques, it might be possible to have a certain accuracy in the classification problem or at least have a better idea of what happened in the neural network.

References

[1] Yoav Goldberg. A primer on neural network models for natural language processing. *arXiv:1510.00726*, 2015.

[2] Jon Kleinberg, Himabindu Lakkaraju, Jure Leskovec, Jens Ludwig, and Sendhil Mullainathan. Human decisions and machine predictions. 2017. http://www.nber.org/papers/w23180 - Working paper.

[3] Jongbin Jung, Connor Concannon, Ravi Shroff, Sharad Goel, and Daniel G. Goldstein. Simple rules for complex decisions. 2017. Stanford University.

[4] Matthew Honnibal and Ines Montani. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear*, 2018.

[5] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*, 2017.

[6] Laurens Van Der Maaten and Geoffrey Hinton. Visualizing data using t-sne. 2008. Journal of Machine Learning Research 9.

[7] Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. From word embeddings to document distances. 2015. Washington University in St. Louis, 1 Brookings Dr., St. Louis, MO 63130.

[8] Gephi, the open graph viz platform, 2018.

Appendices

A List of hashtags

List of hashtags pro-Trump:
maga, masa, americafirst, buildthewall, draintheswamp, lockherup, maga3x, makeamericagreatagain, march4trump, presidenttrump, trumppence16, trumptrain, trumpwins4usa, wakeupamerica, wethepeople

List of hashtags anti-Trump:
notmypresident, donthecon, dumptrump, fakepresident, fucktrump, impeachtrump, impeachtrumpnow, liarinchief, notmypotus, notmypresident, putinspuppet, resisttrump, socalledpresident, stoppresidentbannon, theresistance, trumplies, trumpregrets, crookeddonald, crookedtrump, lyingtrump, notmypresident-trump, orangehitler, stoptrump

B Word Mover Distance graph

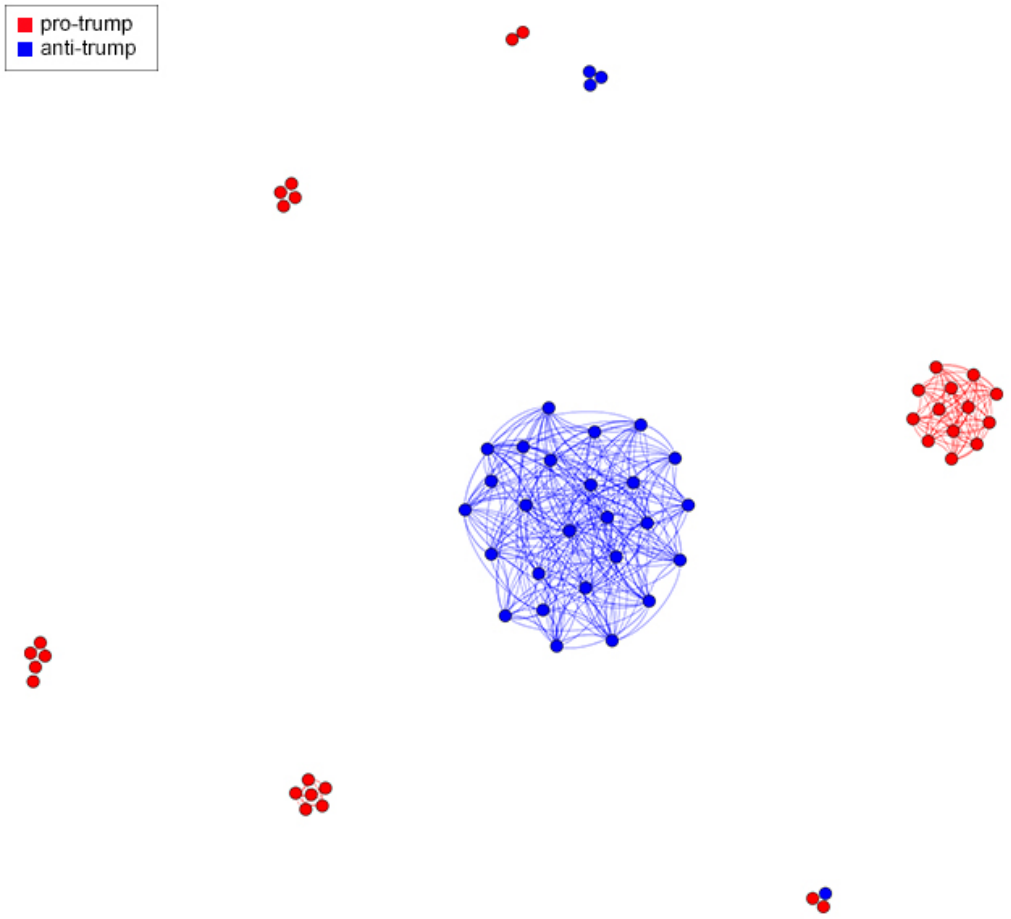

Figure 8: Word mover distance graph on 62 n-grams in "sater felix muslims" topic (pro-Trump n-grams represented in orange, anti-Trump n-grams represented in light blue)

C Feature extraction

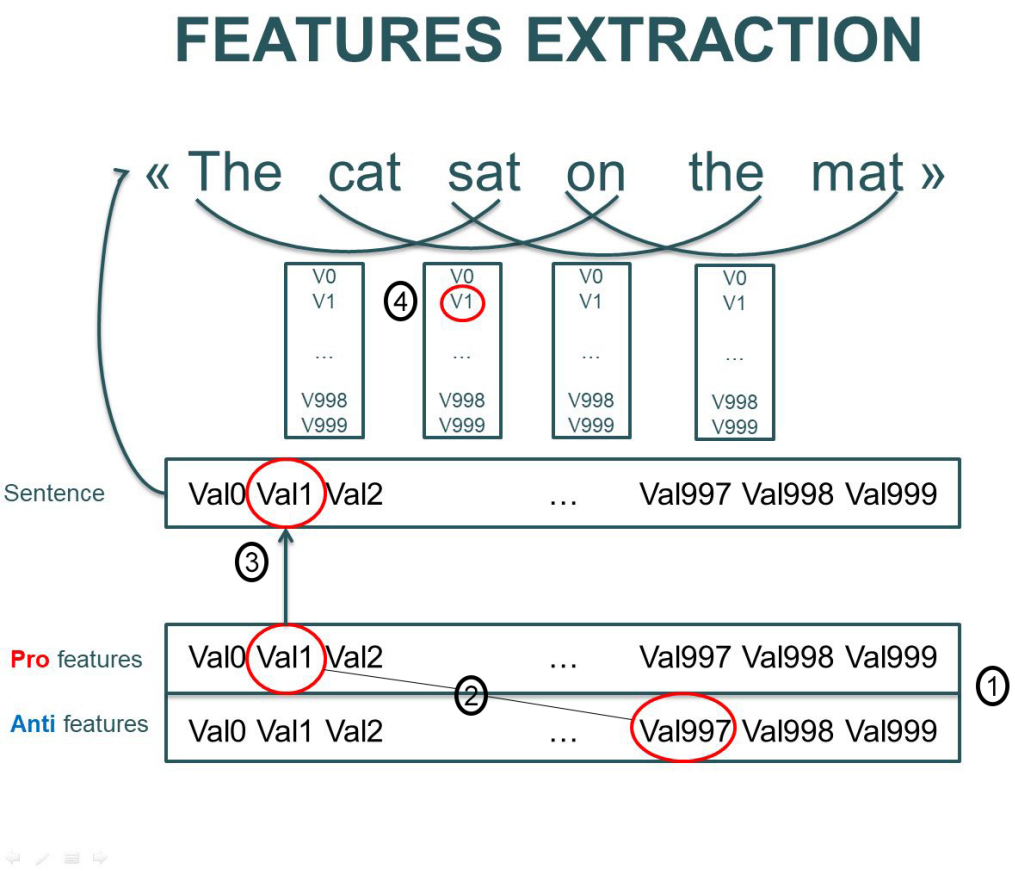

Figure 9: Feature extraction pipeline: 1) Find vectors for each class in Linear Layer. 2) Take greatest element for each vector. 3) Find sentence with greatest value at the index found in 2). 4) Find n-grams that activate the most this feature.