Bogazici University

CMPE 492 MIDTERM REPORT

Topic Modeling for Twitter Accounts

Burak Suyunu - 2012400156
Mehmet Akif Çördük - 2012400228

supervised by
Assoc. Prof. Ali Taylan Cemgil

March 27, 2017

1 Introduction

The increase in social media usage has created tremendous number of data. These kind of user generated unstructured data is created a new phenomenon called big data. As a result, the interest in big data and data processing is increased. In order to make use of the big data, statistics and machine learning methods have been applied. Big companies such as Facebook, Twitter, Instagram etc. have opened the access of their data to developers through the API's. These API's are used to create programs for a lot of useful purposes like marketing, advertising and social media analysis.

Social media is the place where everyone posts their interests, specialty and opinions. Twitter is considered as high profile social environment where everybody makes use of other's opinions. But finding the person of interest can be hard even on global and easy-to-find Twitter. Because a Twitter account does contain various kinds of tweets and it is hard to determine who tweets about what. Our project is to categorize the Twitter accounts by their tweets. We considered each twitter account as documents and used topic modeling to determine what the account is about. We experimented on the followers on TRT World which is the global news channel located in Turkey. The followers are diverse which is useful for the start of our research since we can examine different features from different accounts from different countries. From 15.000 followers we pulled 2.000 the accounts used in English. We reached satisfying results on modeling these Twitter accounts' topics.

You can find all the codes at "https://github.com/suyunu/Senior-Project"

2 Related Work

2.1 Topic Modeling

In machine learning and natural language processing, a topic model is a type of statistical model for discovering the topics that occur in a collection of documents. Topic modeling is a widely used text-mining tool for discovery of hidden semantic structures in a text body. We know that a document is about a particular topic, we expect particular words to appear more often than others since some words are more related to the subject. For example, if a document contains words like Trump, election, republican, towers, it is most likely that this topic is related to U.S election and it is less related to economy (Trump towers). So we can say it is 20% about economy and 80% about the U.S presidency. A topic model captures the cluster of similar words based on the statistics of the words in document. Topic models are also referred as probabilistic topic models which is based on statistics algorithms for discovering the latent semantic structures of a text body. Techniques used in modern topic modeling algorithms are SVD (Singular value decomposition), NMF (Non-negative matrix factorization) and some extensions of LDA (Latent Dirichlet Allocation).

2.2 K-Means Clustering

K-means clustering is a method used in vector quantization, cluster analysis and feature learning. It is an NP-hard problem even just for two clusters. However there are efficient heuristic algorithms that quickly converge to a local optimum. The method is mainly used for grouping N number of data points to K clusters by minimizing the sum of euclidean distance from data points to mean of clusters.[wik]

The algorithm runs iteratively to converge into some cluster mean points. First we determine initial K means $m_1, ..., m_K$. Since it is an NP-hard problem, some heuristics may be applied to determine the initial clusters to make it converge to a near-optimal solution. Next step is to assign each data point to a cluster by comparing the euclidean distance. After the assignment of each point, we recalculate the new means of each cluster. These two steps are applied iteratively until the mean points of clusters don't change, in other words converge to some mean points. The assignment and update steps are as follows:[Mac03]

$$S_i^{(t)} = x_p : ||x_p - m_i^{(t)}||^2 \leq ||x_p - m_j^{(t)}||^2 \forall j, 1 \leq j \leq k$$

$$m_i^{(t+1)} = \frac{1}{|S_i^(t)|} \sum_{x_j \in S_i^{(i)}} x_j$$

where t is the number of iteration, S_i is the cluster, m_i is the mean of the cluster, x is the data point.

2.3 Latent Dirichlet Allocation

Think about a paper which is about using the data analysis to determine the number of genes the organism needs to survive. Assume that by hand, we highlight the words about data analysis in blue, evolutionary biology in pink and genetics in yellow. We see that blue,pink and yellow colors are in different proportions. LDA is a statistical model of document classification that tries to capture above mentioned concept. We define a topic to be a distribution over a dictionary. For instance, genetic topic has genetic related words with high probability and data analysis words with low probability.

For each document we have, we generate the words in two-stage process:[Ble12]

- Randomly choose a distribution over topics
- For each word in the document
  - Randomly choose a topic from the distribution over topics in step #1
  - Randomly choose a word from the corresponding distribution over the vocabulary.

This statistical model reflects that a document can have multiple topics with different proportions. The distribution in step #1 is called a Dirichlet distribution. In the generative process of LDA, the Dirichlet is used to allocate words of the documents to different topics. So that, all of the documents share the same set of topics, but each document exhibits those topics in different proportions. We have a hidden structure here that is the topics, topic distributions per topic, topic distributions per word in documents. The main problem is to infer these hidden structure from observed documents. The inferred hidden structure represents the thematic structure of the collection. LDA is a part of the probabilistic topic modeling which uses generative process that defines a joint probabilistic distribution over both observed and random variables. Data analysis is performed over the conditional distribution which is generated from joint probabilistic distribution. This conditional distribution,which is posterior distribution, is the distribution of hidden variables given the observed variables. If we can compute the conditional distribution, we can find the hidden structure.

We will explain the probabilistic model of LDA first informally(by giving examples) then formally(by giving formulas). As mentioned before a document is a mixture of topics that contains words with certain probabilities. We will examine a document generation from topic distributions, then we will backtrack this logic to obtain topics from the documents. First we decide number of words a N document will have according to some distribution(say Poisson distribution). Then, choose a of topic mixture for a document(according to Dirichlet distribution ever fixed number K of topics). For instance we have two topics that is cute animals and food. We might choose the document to consist of 2/3 food and 1/3 cute animals related words. Now we will generate words for the document. For each word to be generated, we first pick a topic that word might belong to, according to above mentioned distribution. The word will be related to food topic with 2/3 probability and cute animals topic with 1/3 probability. After choosing the topic for word to be generated, now we choose a word from the word distribution of the topic. For example, let say the topic chosen is food, we might pick the word "broccoli" with 30% probability, "bananas" with 15% probability and so on. To summarize until now, we first picked a topic mixture for the document(from Dirichlet distribution), from that topic mixture we chose each word to be put into the document(from Multinomial distribution that is words' distribution per-topic). This process is called generative process, the LDA backtracks this process to obtain hidden structure(topic distribution,word distribution per-topic) from the collection of documents.

Now we want to learn topic representation of each document and the words associated to each topic. We learn the hidden structure by using Gibbs sampling. To start, we first assign each word in a document to one of K topics randomly. This random assignment creates both topic representation and the word distributions of all topics, for now they are not fully correct. To improve this distributions, we iterate over each document D and for each document we iterate over each word W. For each topic T, we compute two probabilities:

- P(Topic T |Document D) is the proportion of words in document that are currently assigned to topic t.
- P(Word W |Topic T) is the proportion of word w assignments to topic T over all documents.

We will reassign the word w to a new topic t, which is step of improvement of the distributions. New assigned topic is chosen according to probability $P(TopicT|DocumentD \times P(WordW|TopicT)$. We said that this is the backtracking of our generative process, notice that this probability is essentially the probability that topic T generated word W. So it makes sense that we assign current words topic with this probability. The key point here is that while examining a word we assume that all other words have the correct distribution, we reassign our word according to this assumption. We process these steps a number of times to reach a good assignments of words to topics. Since we now know the correct assignments of words, we obtained the correct topic distribution per document.

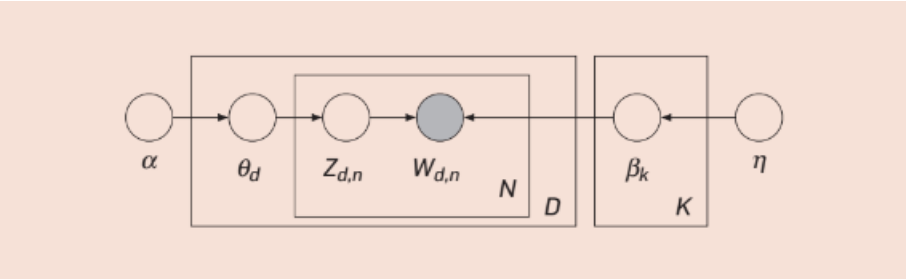
We can describe the generative process of LDA formally by the following joint distribution: [Ble12]

$$p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D}) = \prod_{i=1}^{K} p(\beta_i) \prod_{d=1}^{D} p(\theta_d)(\prod_{n=1}^{N} p(z_{d,n}|\theta_d)p(w_{d,n}|\beta_{1:K}, z_{d,n}))$$

where $\beta_{1:K}$ are the topics, where each β_k is distribution over vocabulary, θ_d is the topic proportions for document d, where $\theta_{d,k}$ is the topic proportion for topic k in document d, z_d is the

topic assignment for document d where $z_{d,n}$ is the topic assignment for the nth word in document d, finally the observed words for document d are w_d, where $w_{d,n}$ is the nth word in document d which is an element over a fixed dictionary. We can see that distribution is composed of dependent random variables which define the LDA.

Figure 1: The graphical model for latent Dirichlet allocation. Each node is a random variable and is labeled according to its role in the generative process. The hidden nodes, the topic proportions, assignments, and topics are unshaded. The observed nodes, the words of the documents are shaded. The rectangles are "plate" notation, which denotes replication. The N plate denotes the collection words within documents; the D plate denotes the collection of documents within the collection.[Ble12]



Now we will explain the computation of conditional distribution of topic structure given the observed documents. This is the backtracking part of the generative process where we find topic distributions from the documents. [Ble12]

$$p(\beta_{1:K}, \theta_{1:D}, z_{1:D}|w_{1:D}) = \frac{p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D})}{p(w_{1:D})}$$

We use the Gibbs sampling algorithm to create a Markov chain. The Markov chain is defined on the hidden topic variables for a particular corpus, and the algorithm runs the chain for a long time collecting samples from the limiting distribution and then it approximates the distribution with collected samples.

3 Methods

In Twitter, on a daily basis people and organizations express their feelings generally in words but also with pictures and videos. So we can think Twitter as a very big open collection of personal thoughts. However, like in our daily life thoughts, the data on Twitter is also unstructured and unlabeled. So understanding and making deductions from the data of a Twitter user is a very crucial task.

In this project, we are focusing on grouping Twitter users under some common topics. This process is called Topic Modeling in the literature. Topic modeling has variety of application field. In this paper we used topic modeling to predict what topics followers of TRT World's Twitter account are tweeting about. In short, we tried to determine a Twitter account's follower profile by investigating their recent tweets. We used Python3 for our project and we tried different methodologies to categorize the follower profiles and while doing this we also took advantages of some nice Python tools.

First of all, we used python wrappers of Twitter API to collect tweets and some useful information of Twitter users. However, we couldn't start to apply our topic modeling algorithms directly on those data right after we gathered them because tweets are unfortunately highly degenerated texts. So we applied lots of natural language processing methods to tweets to make them more viable for our topic modeling algorithms. After that we used a bag-of-words representation and created a dictionary and a corpus based on this preprocessed tweets of bag-of-words.

Before using this preprocessed data, we implemented a k-means clustering algorithm on a small dataset to start our project.

We used our preprocessed text corpus and dictionary to train a Latent Dirichlet Allocation model. As a result of training LDA, we gained the distribution of topic over users and also words distribution over topics. And lastly we represented both the word distribution over topics and similarity between users via graphs.

3.1 Collecting Twitter Data

Collecting Twitter data is an easy but a timely process. It is easy because Twitter API is very user friendly and very well documented. It is a timely process because like every other API it has its own rate limits and also there are lots of data to gather. We used followers of TRTWorld Twitter account as our database. So we collect the information of those users. From now on we will call our users for the followers of TRTWorld Twitter account. To deduce meaningful patterns from a Twitter user we used its:

- User ID
- Screen name
- Language
- Status (Tweet) count
- Protected status (If protected, we are unable to get its tweets)
- Recent 200 tweets.

3.1.1 Twitter API

To collect Twitter data of users, we used Python wrappers of official Twitter API, python-twitter and tweepy. To collect all the information about users that we have mentioned above, we followed those steps (We put original Twitter API request names rather than python wrapper equivalent):

1. We collect the IDs of our users with "GET followers/ids" request. This request collects the specified account's (TRTWorld) followers' IDs in groups of 5000. Thanks to its paging system, we could easily collected all of our users' IDs.

2. Thanks to the tip in the documentation of Twitter API, we used "GET users/lookup" request in conjunction with collections of user IDs returned from "GET followers/ids" and collected all the required information (screen name, language, status count, protected status) of our users.

(a) Here while collecting the data, we only kept users whose language is English, whose account is not protected and who has at least 200 tweets.

3. In the last step we collect all the remaining users' last 200 tweets with "GET statuses/user_timeline" request.

TRTWorld Twitter account had 75000 followers when we started this project. We collected 15000 of it in the first step. After elimination process in the second step 2000 accounts left. In the third step we collected users' last 200 tweets for evaluation purposes.

3.1.2 Natural language processing

The language of twitter is generally close to daily language. People share their ideas and emotions at any time of the day. Other than normal texts, tweets can include hashtags, emoticons, pictures, videos, gifs, urls etc. Even normal text part of the tweets may consist of misspelled words. Apart from these, one user may tweet in lots of language. For example, one tweet may be in Turkish, and another one in English. So we need to make a cleanup before using those tweets. We will explain each natural language process that we applied item by item.

- **Remove Twitter Accounts that has less than 2000 words in their tweets.** We are excluding accounts that have relatively small word count.
- **Remove URLs.** We have removed all urls which are starting with "http://" or "https://. So we excluded all pictures, videos, gifs etc. from the text.
- **Tokenization.** Tokenization is basically process of splitting text into words, phrases or other meaningful elements called tokens. We words as our tokens. To better process the text and to create a dictionary and a corpus we tokenized and converted to lower case all the tweets. We used nltk library with regexp to tokenize.
- **Stop words.** Stop words usually refer to the most common words in a language. So being common makes stopwords less effective and sometimes misleading while making decisions. Thus generally stop words are words which are filtered out. We used nltk library to obtain general English stop words, also we determined some words ourselves and also added one and two character words from tweets to stop words.
- **Remove non-English words from tweets.** We used nltk corpus to remove non-English words form tweets.
- **Remove non-English accounts.** It is an extension process to removing non-English words. After removing non-English words from tweets, we removed accounts from our corpus whose tweets are majorly not in English.
- **Delete accounts whose number of left tokens are less than 200.** After all those preprocessing on tweets, we have removed lots of words from original tweets. Some of the accounts, which are possibly not majorly in English but still includes English words, effected more but still existed in the corpus. So to eliminate those misleading accounts from the corpus we deleted accounts whose number of left tokens are less than 200.
- **Stemming.** For grammatical reasons, documents are going to use different forms of a word, such as organize, organizes, and organizing. Additionally, there are families of derivationally related words with similar meanings, such as democracy, democratic, and democratization. The goal of stemming is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form. [steb] nltk library has mainly 3 kinds of stemming tools for English: lancaster, porter and snowball. We chose Snowball stemmer because it uses a more developed algorithm then Porter Stemmer (Snowball is also called as Porter2) and less aggressive than Lancaster. [stea]
- **Remove words that appears only once in the whole corpus.** This process removes some kind of outlier words (like non-English, meaningless or heavily degenerated words) from the corpus which are passed undetected from the former natural language processes.

3.1.3 Dictionary and Corpus

To properly use the Twitter data that we have preprocessed, we need to put into a shape that will be understandable by Topic Modeling algorithms. Bag-of-words representation is perfect fit for those kind of algorithms. In bag-of-words we first created a dictionary which consists of all the words from our preprocessed twitter data as values and their ids as keys. Then we created our corpus. Each element of the corpus corresponds to one Twitter account. Each element consists tuples which includes dictionary id of words and the number of that words' occurrences in that account. We used a very useful python library called Gensim to create our dictionary and corpus.

3.2 K-means clustering

Grouping Twitter users with K-means clustering was the first method that we have tried. It was a very basic and easy approach to the problem and also it is K-means algorithm is not a direct topic modeling algorithm. But here we wanted to observe that, even without much text processing and bigger data, could we classify Twitter users with only their tweets. We got small but promising results which winded us up.

We first created two data group which consist of 8 sport accounts and 12 political accounts and collected their last 200 tweets. We made very few preprocess on the data. First, we tokenized the tweets and then removed words that were appeared less than 3 times in all the corpus. After that we created a dictionary and a corpus with bag-of-words method. After creating dictionary we added 8 more accounts to the corpus. As we have mentioned before each element of corpus was consist of accounts. And representation of those element were word id and word occurrences count tuples. But this time we didn't count the occurrences of the word, we put 1 if the word was exist in the tweets of that user, 0 otherwise.

To apply K-mean clustering we used Python library called sklearn and tried to cluster data into 2 clusters. It worked like a charm and clustered data nearly perfect. Obviously data was very easy to cluster because of the tendencies of the given Twitter accounts. But as we mentioned before, this was our first try and observing an output like this was very motivational. We created a basic metric to plot those high dimensional representation of the accounts in the corpus to a 2D plot. For each account, first dimension was the sum of the multiplication of each dimensions of the account and the first cluster point. Second dimension is the same process with the second cluster point.

3.3 Latent Dirichlet Allocation(LDA)

We used the Python library called Gensim to train our corpus using LDA model. LDA has 3 main parameters need to be optimized. Finding the right parameters for LDA can be considered as an art:

1. K, the number of topics
2. Alpha, which dictates how many topics a document potentially has. The lower alpha, the lower the number of topics per documents
3. Beta, which dictates the number of word per document. Similarly to Alpha, the lower Beta is, the lower the number for words per topic.

Since we are dealing with tweets, we assumed that each follower would have a limited number of topics to tweet about and therefore set alpha to a low value 0.001. (default value is 1.0/num_topics). We left beta to its default setting. We tried several different values for the number of topics. Too few topics result in heterogeneous set of words while too many diffuse the information with the same words shared across many topics. Training LDA on near 800 accounts with parameters "number of topics = 30 and passes = 20" took 7 minutes on our Dell Inspiron i7559 laptop which has Intel i7 processor and 8GB RAM.

The output of the LDA model gives us lots of useful information as expected, word distributions over topics and topic distribution over users. However those information are all hard to read and interpret by looking. Fortunately, we found a library called LDAvis to explore and interpret the results of LDA. LDAvis maps topic similarity by calculating a semantic distance between topics (via Jensen Shannon Divergence) [ale]

Apart from topic visualization we also calculated the similarities between each user based on the LDA results using a function of gensim library and created a distance matrix between users. Then we plotted that distance matrix on a 2D plot which shows similar users closer to each other while showing users that tweets about different topics farther.[con] Then we applied K-means algorithm on that 2D data to observe similar users visually better. You can find more detailed analysis and graphs on the results section.

4 Results

4.1 Results of K-Means

K-means clustering without improving data with NLP was our first try of topic modeling. We used a very basic and separable dataset and got very good results. Here in Figure 2 we can observe that k-means succeeded in grouping the data into right clusters. However, K-means was just a simple trial to apply topic modeling without knowing anything about topic modeling.

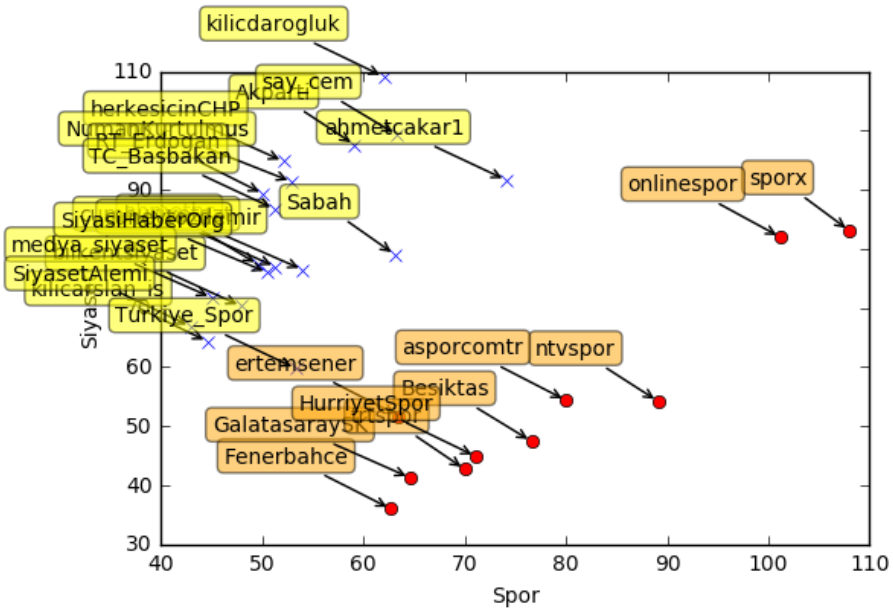
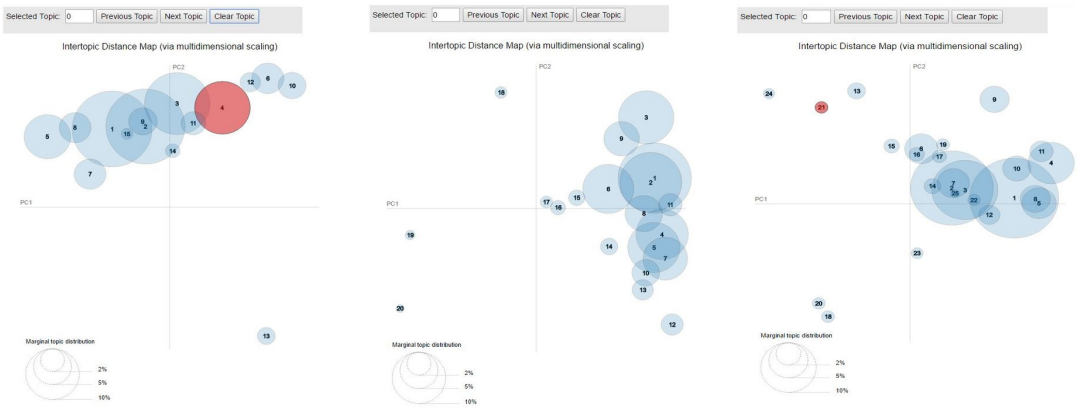


Figure 2: K-means clustering on a simple data

4.2 Results of LDA

We have run the LDA topic modeling program with different parameters and observed satisfying results. When we trained LDA model, it gives us word distributions over topics and topic distribution over users as lists and some extra information. However topic and word distributions are high dimensional vectors and hard to interpret. So, to understand and visualize the output we used a wonderful tool called LDAvis. LDAvis extracts information from a fitted LDA topic model and plots it onto 2D. On this plot we can observe topics as bubbles, the bigger the bubble it covers more of the documents.



(a) 15 Topics (b) 20 Topics (c) 25 Topics

Figure 3: LDAvis - Inter-topic Distance Map with different topic numbers K

Figure 4: LDAvis - Word distribution over topic

Also we observe the distribution of the words over topics in an interactive way. The tool shows the topic's 30 most relevant words and also you can change the relevance metric λ which sets words appearance in selected topic to all topics ratio.

As we improved our NLP, we run the program with different parameters. First we observed that some words are only used once or twice in the whole corpus which disturbs the topic distribution of words. We removed the words that are used only once or twice in whole corpus and we observed improvements on the results but there were still some words which are in another language, irrel-

evant or non-sense. So we decided on removing the words that are used only three times in whole corpus. Then we trained the program with 20 topics and realized that some accounts have low number of tokenized words(have small number of tweets or it melted after NLP) which disturbs the distribution. So that, we removed the accounts that have less than 200 tokenized words. After that we added Snowball stemmer to improve our results.

We started our process with 836 unique accounts and after all those natural language processes we ended up with 641 unique accounts with 8311 unique tokens and trained our LDA model with this corpus and dictionary.

After several runs we decided that training the LDA model with 25 topics gives us better results with respect to dividing into more meaningful topics for this dataset.

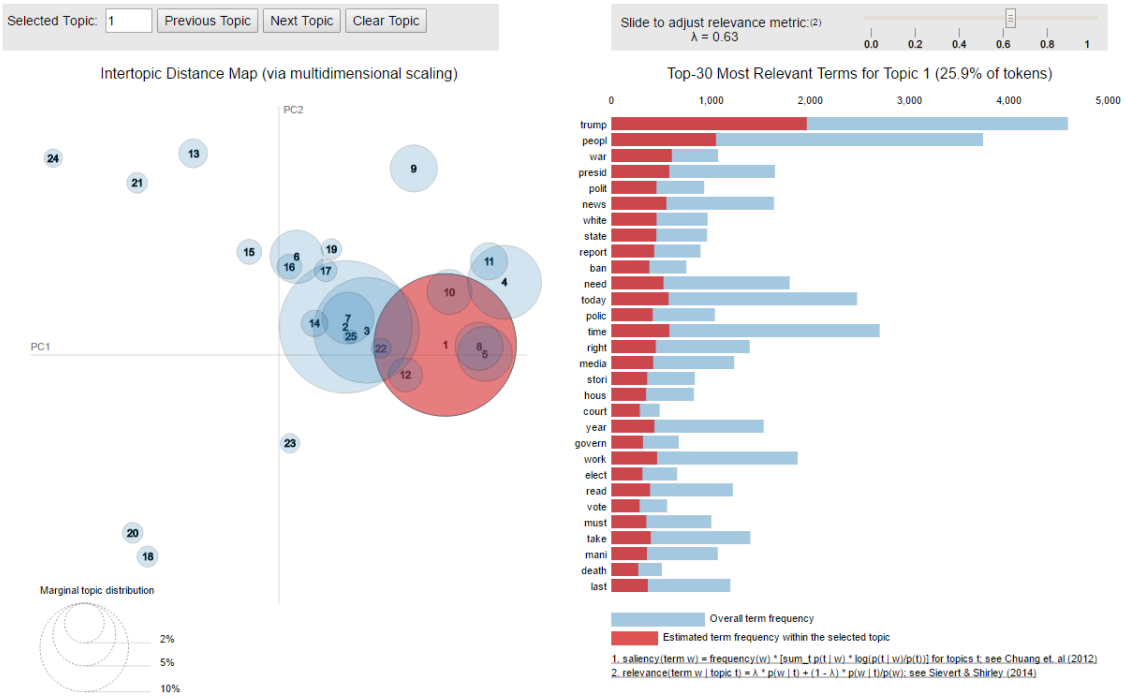


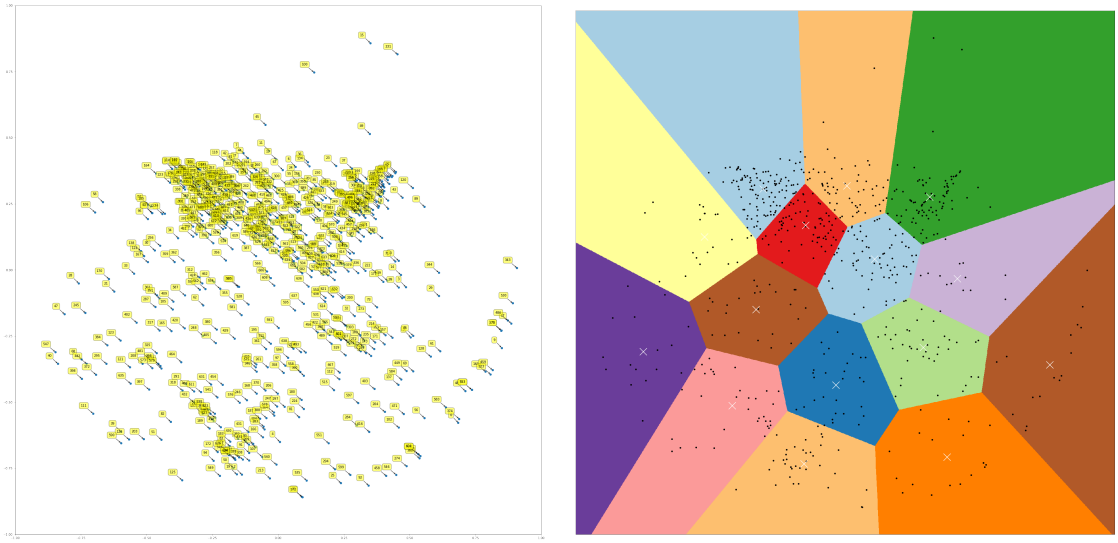
Figure 5: Observed best clustering over 25 topics

In Figure 4 and Figure 5 you can observe the outputs of LDAVis from our last trained LDA Model with 25 topics. You can see that TRT World's followers mostly tweeted about US Presidential election. Words related to US Presidential election constitutes a big portion of the whole corpus. But we face with the real power of the LDA when we dig deeper to understand the names of the topics. In Figure 4, you can see the word distributions of topics trained with the same LDA as Figure 5.

- **Topic 1:** Trump and US Presidential election
- **Topic 4:** Recent events in Turkey
- **Topic 6:** Sports
- **Topic 9:** Turkish words that escaped from NLP

The results are very promising. LDA splits the the data into meaningful topics successfully. But of course, there are a lot of room for improvements. First of all we need more data to get better results. Also as we can see in Topic 9, we need to make better NLP on data. It is also nice to see that LDA can detect missed Turkish words for us.

To observe the relations between each user we calculated the similarities between each user based on the LDA results using a function of gensim library and created a distance matrix between users. To visualize this distance matrix we converted this distance it into 2D plot which shows similar users closer to each other while showing users that tweets about different topics farther.



(a) 2D plot of distance matrix of users

(b) k-means clustering with 15 clusters [sci]

Figure 6: Visualization of user similarities using data from LDA with 25 topics

Then to understand this plot better we applied K-means algorithm on that 2D data to observe similar users.

In Figure 6a we can see the how relevant the users are according to our distance matrix. We can observe that data is grouped around one big and a number of small centers. By investigating each point one by one we can understand which users are related to each other and what topics they are interested in. For example, when we print the topic distribution of users with $id = 366$ and $id = 277$ we get what we are looking for:

- **366** = Topic 8 with probability 0.03 and Topic 9 with probability 0.97
- **277** = Topic 8 with probability 0.09 and Topic 9 with probability 0.91

In Figure 6b you can see the k-means applied version of the left dataset. Even though we trained LDA with 25 topics, here we trained k-means with 15 clusters. We got reasonable clusters, which may help us to understand data when we have more data which includes more separable topics.

5 Further Work

5.1 More Data

We have worked on the TRTWorld Twitter account, although it has a diverse follower range, the tweets are generally unrelated to each other. So that an account will have too many topics, so it is harder to assign specific topics to an account that tweets in so many topics. We need to find a good data set that has tweets about specific subjects. A good dataset example might be twitter accounts of maker communities, since these accounts have more specific topics. For now, we have implemented the program only in English, we will retrieve tweets in Turkish language and model the topics in Turkish.

5.2 More NLP

As we mentioned before, we are taking accounts that tweets in English. However, we encountered tweets that contain non-English words, special nouns. Some of the accounts that are marked 'en', tweets in multiple languages, we will try to improve our algorithm against these issues. Stemming is used to detect words that have derived from the same word. We will try different stemming algorithms as well as lemmatization. We will add NLP methods to process Turkish language, since we will also work on Turkish. Finally, hashtags should have great importance on the subject of a tweet as well as the account. To improve the algorithm, hashtags will have more importance than other words in the corpus.

5.3 Other Methods

Although the implemented LDA model is widely accepted and successful algorithm, we need to try other algorithms to understand topic modeling better and experiment with other methods if they give better results for our interest which is topic modeling of Twitter accounts. Most of the topic modeling algorithms are under probabilistic topic modeling subject. We will try to implement NMF (non-negative matrix factorization) and LSA(Latent semantic analysis) using k-means clustering.

References

[ale] Topic modeling of twitter followers. http://alexperrier.github.io/jekyll/update/2015/09/04/topic-modeling-of-twitter-followers.html.

[Ble12] David M. Blei. Probabilistic topic models. *Commun. ACM*, 55(4):77–84, April 2012.

[con] Convert matrix to 2d plotting. http://baoilleach.blogspot.com.tr/2014/01/convert-distance-matrix-to-2d.html.

[Mac03] David J. C. MacKay. Information theory, inference and learning algorithms. *Cambridge University Press*, 2003.

[sci] Plot k-means with sci-kit. http://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_digits.html.

[stea] Stemmers. http://stackoverflow.com/a/11210358/6747127.

[steb] Stemming and lemmatization. https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html.

[wik] Topic modeling. https://en.wikipedia.org/wiki/Topic_model.