Bogazici University

CMPE 492 FINAL REPORT

Topic Modeling for Twitter Accounts

Burak Suyunu - 2012400156
Mehmet Akif Çördük - 2012400228

supervised by
Assoc. Prof. Ali Taylan Cemgil

June 5, 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 created a dataset about makers' and influencers' tweets and experimented our program with this dataset. We applied LDA using SkLearn and Gensim modules of python and applied NMF using SkLearn of python. 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.

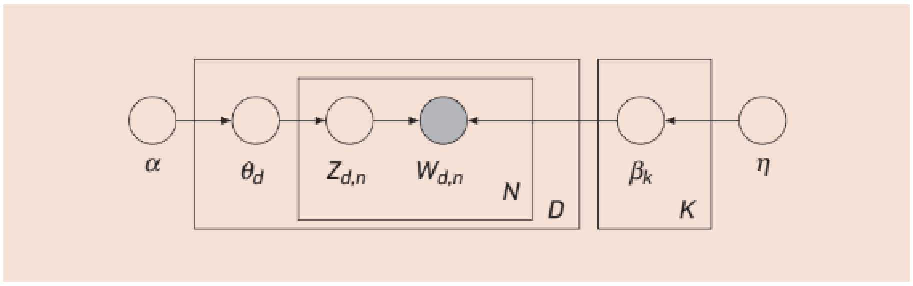
We will reassign the word w to a new topic t, which is step of improvement of the distributions. New assigned topic is chosen according to probability $P(Word\ W|Topic\ T) \times P(Topic\ T|Document\ D)$. We said that this is the backtracking of our generative process, notice that this probability is 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.

2.4 Non-negative Matrix Factorization

NMF is a state of the art feature extraction algorithm which is useful when there are many attributes and the attributes have weak predictability. NMF combines the attributes and produces meaningful patterns and topics. NMF decomposes the data according to user-defined number of features. Each feature is a linear combination of original attribute set. The coefficients of the linear relation are non-negative, this name of the algorithm comes from this concept. NMF decomposes the data matrix V to two lower rank matrices W and H where product of these matrices approximately equals to original data matrix V. NMF uses iterative procedure to modify the initial data of W and H so that the product of these matrices approaches to original matrix. The rank of the the matrices is chosen r(n+m) $<$ nm so that W and H can be regarded as compressed form of V.

The decomposition is done by optimizing the Frobenius norm which is basically Eucledian norm extended to matrices. Other optimization objectives have been suggested in the NMF literature, in particular Kullback-Leibler divergence. However, these objectives are not implemented by sci-kit learn yet.

$$\arg\min_{W,H} \frac{1}{2}||V - WH||^2_{Fro} = \frac{1}{2}\sum_{i,j}(V_{ij} - WH_{ij})^2 \qquad \text{(Objective function)}$$

The directly visible variables are constructed from the hidden variables according to NMF model. Each hidden variable co-activates a subset of visible variables, or "part". Activation of a combination of hidden variables combines these parts additively to generate a whole. There are several update rules for NMF, sci-kit learn uses additive update rule without any subtractions. Such additive methods are efficient for representing images and text. So that, we are constructing the actual data from the decomposed matrices.

For text categorization application, a corpus of documents is represented by a matrix V. The word counts in the corpus can be regarded as the set visible variables modelled as being generated from an underlying set of hidden variables. In the VQ(Vector Quantization) factorization , a single hidden variable is active for each document. If the same hidden variable is active for a group of documents, we say that they are semantically related,because they have similar frequencies of word occurrence. Each column of the W,or semantic feature, consists of the word frequencies of the corresponding hidden variables. VQ allows only one semantic variable to be active, which prevents more than one topic from being attributed to a document. PCA can be a solution for activating multiple semantic variables, however using PCA generates semantic variables that are difficult to interpret visually. So we use NMF,which allows non-negative values for semantic variables. In each semantic feature(columns of W), NMF has grouped semantically related words together. Each document is represented by additively combining several of these features. For example , to represent the document about "Constitution of the United States", the semantic feature containing "supreme" and "court" and the one containing "president" and "congress" are co-activated. The algorithm also uses context to differentiate the different meanings of a word. In a document the same word can occur anywhere in the document but with different meanings or in different concepts. For example "hike" can be used with "mountain" or "interest", NMF determines the meaning according to the concept:

- "Hike" + "Mountain" -> "Outdoor sports"
- "Hike" + "Interest" -> "Interest Rates"

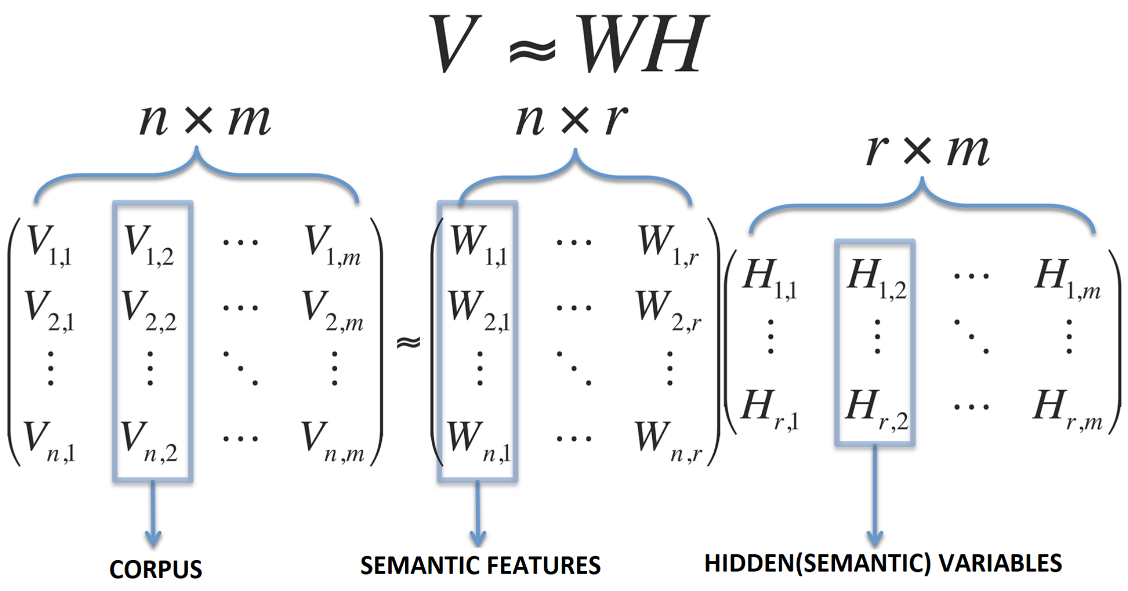


Figure 2: Non-Negative Matrix Factorization

2.5 Word2Vec

Word2Vec is model for creating word embeddings for a collection of words. Word embedding is the language modeling technique used in NLP(Natural Language Processing) where words are mapped to vectors of real numbers. The mapping involves mathematical embedding from a space with one dimension per word to a continuous vector space with much lower dimension. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located in closely to one another in the space. The methods used for generating this mapping neural networks, dimensionality reduction, probabilistic models etc. The most common method is the skip-gram neural network model. It trains a simple one layer neural network to perform a certain task. But it doesn't use the neural network for the task. Instead, the goal is actually just to learn the weights of the hidden layer which are,in fact, the "word vectors" we are trying to learn. Word embeddings have been showed to improve the performance in NLP tasks. The method is especially very powerful in detecting words that are used in similar context.

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 project we used topic modeling to determine what topics influencer and makers in Twitter are tweeting about. We used Python3 for our project and we tried different methodologies to categorize our user dataset.

First of all, we wrote a tool to create a dataset consists of influencer and makers. We used Twitter API clients for Python to collect Twitter Data. 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.

We used LDA of Gensim and both LDA and NMF of scikit-learn tools. Gensim has its own representation of corpus and dictionary. In scikit-learn, we used term frequency–inverse document frequency (tf-idf) statistics for NMF and term frequency (tf) statistics for LDA.

As a result, we obtained the distribution of topics over users and also distribution of words over topics. And finally we visualized both the word distribution over topics and similarity between users via beautiful graphs.

Also, before all this effort, we had implemented a k-means clustering algorithm on a small dataset as an entrance to the project.

3.1 Collecting Dataset

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. Our ultimate dataset is makers and influencers including machine learning, arduino, 3dprinting, robotics, data science experts. We manually collected a number of twitter users which are influencer in some topic. Then, we wanted to expand our dataset to a large number. As a result, we wrote a python script that finds similar twitter users to a given user base.

3.1.1 Similar-Twitter

With Similar-Twitter tool we are trying to find similar users to a given user base on Twitter. The objective is to create a database of similar users around a topic. We are mainly focusing on Twitter lists to find similar users. A list is a curated group of Twitter accounts. One can create its own lists or subscribe to lists created by others. Viewing a list timeline will show a stream of Tweets from only the accounts on that list. Lists seems nice however, because lists are created by Twitter users, it is hard to find a good list to subscribe. For example; Andrew NG is member of XXXXX lists. So to maximize the usage of lists, Similar-Twitter focuses on common lists of base users to obtain similar users.

While coding Similar-Twitter we used **birdy** (cite https://github.com/inueni/birdy) which is a super awesome Twitter API client for Python in just a little under 400 lines of code.

1. **Determine base users.** These users will underlie our similar user database. So it is important to choose users that are related to a common topic.

2. **Get base users' lists which they are a member of.** Here we are using "GET lists/memberships" call to obtain lists.

3. **Extract important specifications of the lists.**

    - name
    - slug

- id
- full_name
- subscriber_count
- member_count

4. **Find common lists which all the base users are a member of.**
5. **Eliminate some lists according to the subscriber and member count of the lists.**
6. **Get members of the common lists.** Here we are using "GET lists/members" call to obtain users of each lists.
7. **Delete duplicate users and extract important information of users.**
    - id_str : ID of the user
    - screen_name : Screen name of the user (@screen_name)
    - followers_count : # Followers
    - friends_count : # Following
    - favorites_count : # Likes
    - listed_count : Total number of list membership
    - statuses_count : # Tweets
    - verified : True or False
    - protected : True or False / if true can't crawl the account
    - created_at : Creation time of the account / (2009-10-30 12:11:39)
8. **Eliminate some users according to predetermined followers and tweets count.**

Objective of all those elimination processes are first discarding useless lists, second discarding non-influencer users.

We have applied Similar-Twitter to lots of different user bases to obtain users from various topics. Finally, in our dataset we have **1118 users** distributed on these topics:

- Artificial Intelligence
- Machine Learning
- Data Science
- Robotics
- 3D Printing
- Arduino
- Computer Vision

As a last step, we collected last 3200 tweets of all the users in our dataset with "GET statuses/user_timeline" request. 3200 is the limit determined by the Twitter API.

Similar-Twitter is our original idea that we study out to create a dataset.

3.2 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.

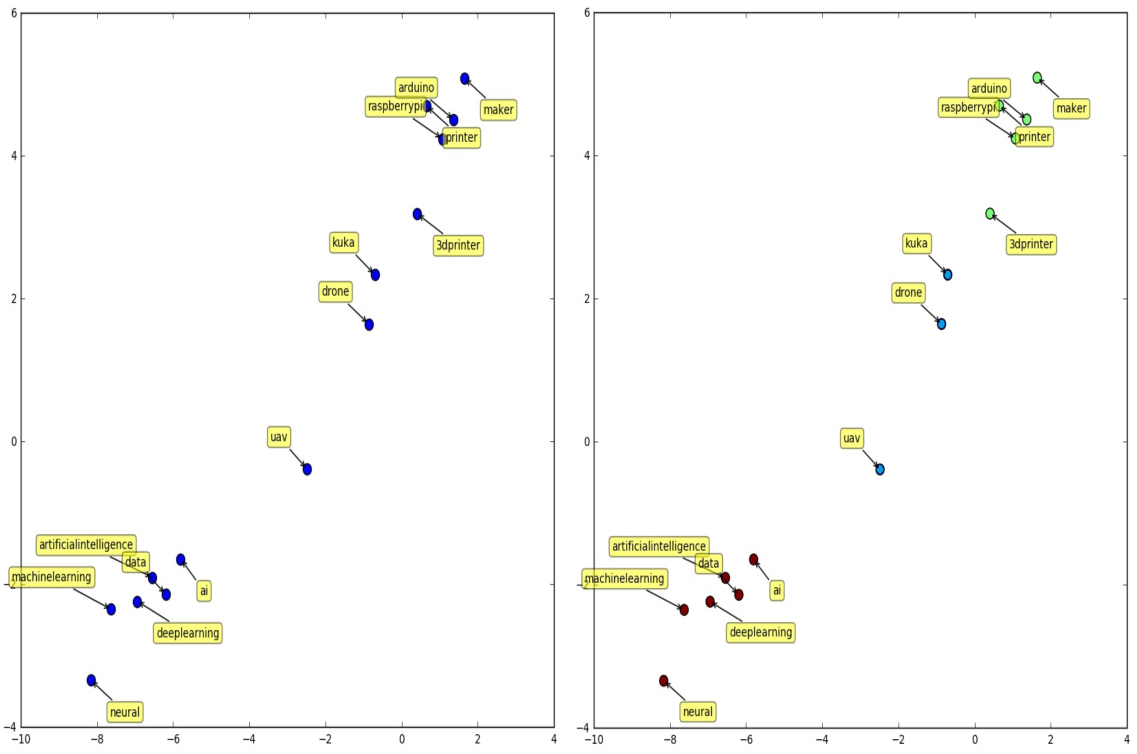
- **Remove Twitter Accounts that has less than 2000 words in their tweets.** We are excluding accounts that have relatively small word count. (1117 users and 328.154.028 words)

- **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.(1117 users and 260.152.628 words)
- **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. (1117 users and 38.119.510 words)
- **Stop words.** Stop-words usually refer to the most common words in a language. So that, being common makes stop-words 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 to be added to stop-words.(1117 users and 22.240.226 words)
- **Remove non-English accounts.** Non-English accounts have a disturbing impact on results. We removed accounts from our corpus whose twitter accounts are not classified as English-"en".(998 users and 19.530.500 words)
- **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, are more effected 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.(998 users and 19.530.500 words)
- **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 at most ten times 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.(998 users and 19.530.500 words)

3.3 Word2Vec

Until now we have applied lots of natural language processes to tweet texts to make them more viable inputs to our topic modeling algorithms. However, all those steps are dealing with words separately. With Word2Vec we tried to approach words from a wholistic perspective. Even we apply stemming to words, this method can only reduce words from same base to a same simpler form. Even for simple similar words stemming may fail to reduce those words to a same form. On the other hand, not only same base words are generally means similar thing, but also some similar words like 'deep learning' and 'neural network' points out very similar topics. So that, with Word2Vec approach we aimed to cluster those very similar words to increase the effectiveness of words on specific topics.

Word2Vec basically turns words into vectors. We used Gensim's word2vec model in our project. It produces word vectors with deep learning via its skip-gram and CBOW models. After we collected and processed all the twitter data, we trained them with Gensim word2vec with vector size 30. This training step converted each of our words into 30 dimensional vectors. We used relatively small vector size because in the next step we applied k-means clustering to the word vectors to find similar words. After lots of trials, training k-means with 2000 clusters gave the best results. With k-means we achieved our finding similar words goal. However, the hardest part was representing those clusters in the corpus. Because now, we have nameless cluster of words instead of one word. So to overcame this problem we decided to gave the name of the word to the cluster that is the nearest to the cluster center. After that, we replaced every occurrence of a cluster member with the cluster name. However, with this approach, clusters including more



words reached so high amounts in the corpus. To handle this problem we normalized the values of the word occurrences with the number of words in the clusters. For example; if the word 'learning' is in the cluster 'AI' and this cluster has 150 words in it. All the words which are in the cluster 'AI' are replaced with the word 'AI'. Then the total occurrences of cluster 'AI' in a document is divided by 150, the number of words in cluster 'AI'.

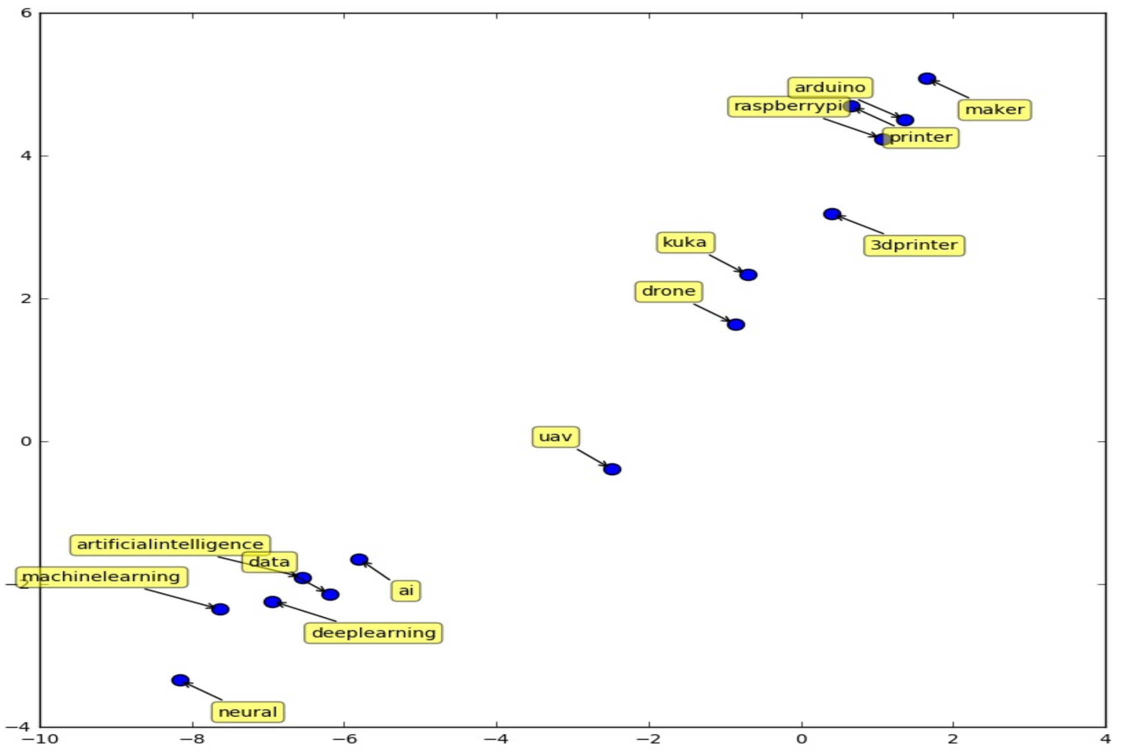
In theory, with this approach we increased the power of some important terms (generally appears in smaller clusters but has high occurrences in corpus), decreased some irrelevant terms (generally appears in bigger clusters but has low occurrences in corpus) and decreased the size of corpus which enables much faster training times. But the naming problem of the clusters is still a big issue.

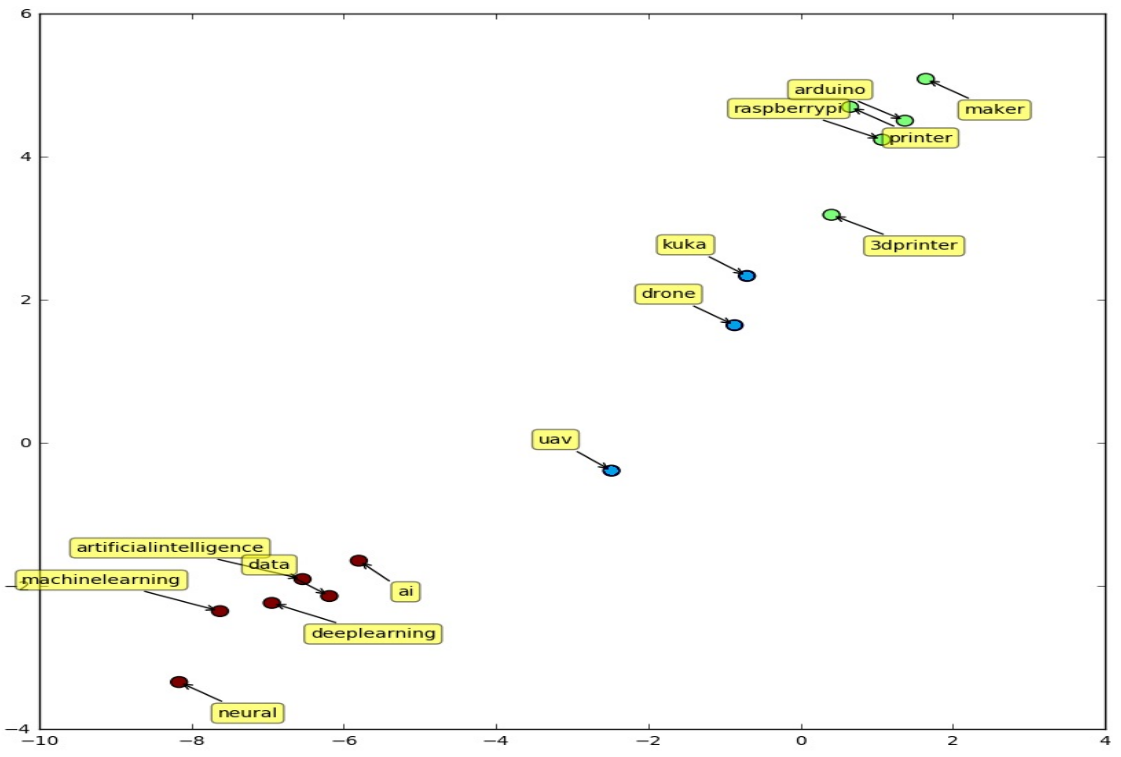
3.4 K-means clustering

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.

3.5 Latent Dirichlet Allocation(LDA)

3.5.1 Gensim

To properly use the Twitter data that we have preprocessed, we need to put into a shape that will be understandable by Gensim LDA algorithm. Bag-of-words representation was perfect fit for tit. 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 again used Gensim's funtions to create our dictionary and corpus.

Gensim-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.

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.

3.5.2 Sci-kit Learn

We first create TF vectorizer according to maximum number of features using count vectorizer. Then, we use this TF vectorizer to obtain the term frequency of the corpus. We create a LDA object by giving necessary parameters:

1. n_topics, number of topics.
2. max_iter, maximum number of iterations.
3. learning method, can be "batch" or "online".
   - Batch: batch variational bayes method. Uses all training data in each EM update(expactation maximization).
   - Online: Online variational Bayes method. In each EM update, use mini-batch of training data to update the variables incrementally. If the data size is large, online update works faster than batch update.
4. A (positive) parameter that downweights early iterations in online learning. It should be greater than 1.0. In the literature, this is called tau_0.
5. Pseudo-random number generator seed control.

After creating the LDA object, we use fit function by giving TF features as parameter. Now that we have our LDA model, we can print top words used in a topic or we can visualize it via LDAvis.

3.6 Non-Negative Matrix Factorization

We used sci-kit learn NMF module to train our NMF model. We first extract the tf-idf features for NMF by using tf-idf vectorizer. TF-IDF is the ratio between two components: 1) the term frequency of a term t in a document d, tf(t,d) and 2) inverse document frequency, idf(t,D) which captures the inverse frequency of a term in a corpus. So if a term like "the" appears many times in a document, but it also appears many times in the corpus, its tf-idf value will be low. A term that is frequent in a document, but not very frequent in the corpus will have a higher score, which means that it is statistically significant. We give the number of features(words in the dictionary) to the constructor then we fit a tf-idf feature via the fit-transform function of the vectorizer. Scikit optimizes a regularized objective function which is the extended form of objective function mentioned in related works. Regularized objective function is Frobenius norm plus prior terms. The necessary parameters for computing prior terms in NMF regularized objective function:

1. K, the number of topics
2. Alpha, constant that multiplies the regularization terms. If set to zero, there is no regularization.
3. L1 ratio, the regularization mixing parameter for the penalty calculation, with $0 \leq L1\_Ratio \leq 1$. For L1 ratio equals 0 the penalty is an elementwise L2 penalty (aka Frobenius Norm). For L1 ratio equals 1 it is an elementwise L1 penalty. For $0 <$ L1 ratio < 1, the penalty is a combination of L1 and L2.

After giving this parameters to the constructor of NMF and call the fit function with the tf-idf parameter.Now that we have our NMF model and tf-idf features, we can print top words used in a topic or we can visualize it via LDAvis.

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.

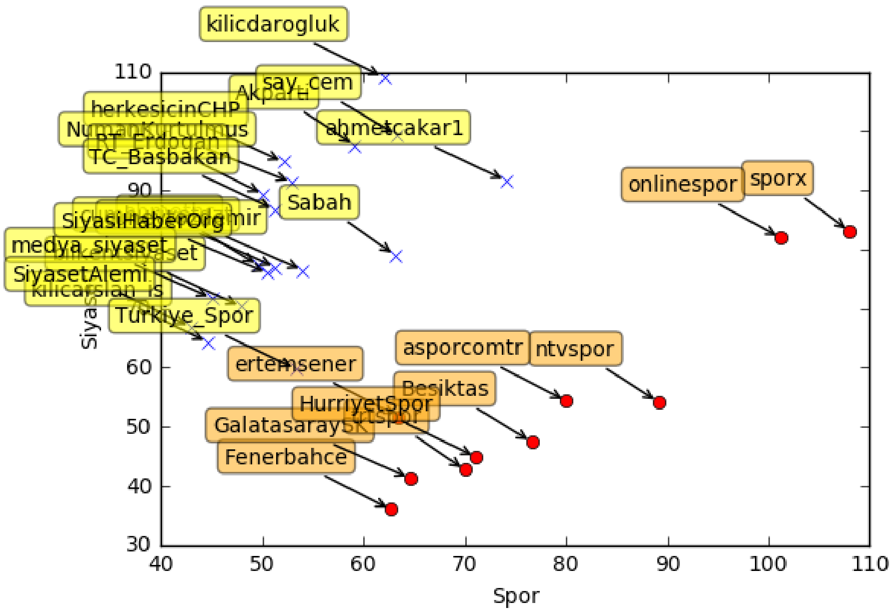


Figure 4: K-means clustering on a simple data

4.2 Results of NLP

We obtained approximately 3.000.000 tweets from 1117 accounts of our dataset all of which are related to computer science. The tweets contained url's, hashtags, non-sense words etc. We applied NLP techniques to obtain a more clear dataset. The results of every step we applied:

- **Remove Twitter Accounts that has less than 2000 words in their tweets.** Since our data set includes the most active twitter users, all of them tweeted more than 2000 words.So that, we have still 1117 accounts after this step.
- **Remove URLs.** We had .... words before removing the URLs. After this step, we have 236.316.529 words in all accounts.
- **Tokenization.** This step tokenizes every single word and converts them into lower case. After tokenization, we have 38.119.510 tokens in all accounts.
- **Stop words.** Stop words from NLTK and our list are removed from the tweets. After this step, we have 22.240.226 tokens in all accounts.
- **Remove non-English accounts.** Remove accounts whose language is not marked as 'en'. After removal of non-English accounts, we have 998 accounts and 19.530.500 tokens.
- **Delete accounts whose number of left tokens are less than 200.** This step doesn't change anything since all of the accounts in the dataset are active users.
- **Stemming.** Stemming finds the roots of the words in a rule based fashion rather than a semantic base fashion. This step doesn't change the number of tokens but it is useful to calculate the number of unique words. After this step, we have 375.149 unique tokens in 998 accounts.

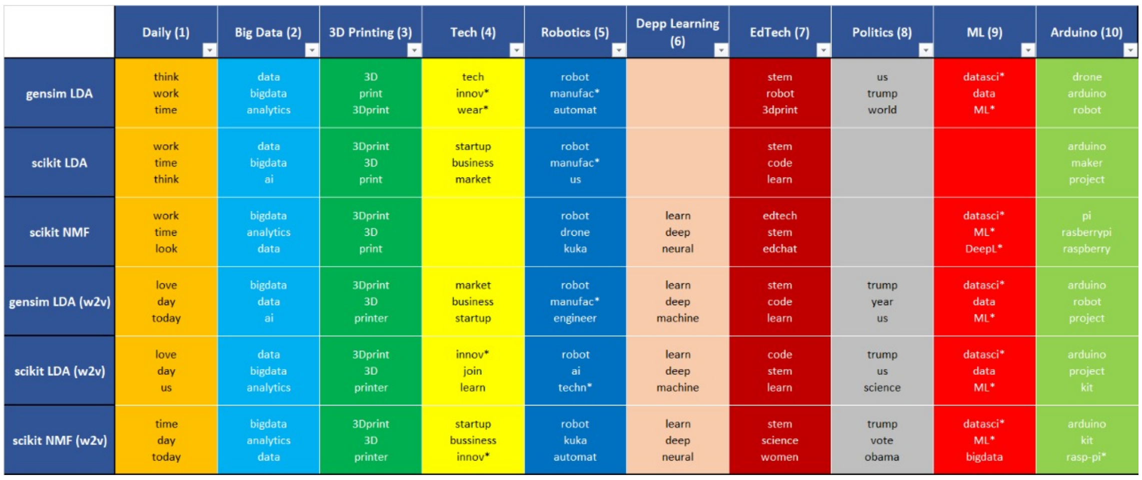
- **Remove words that appears at most ten times in the whole corpus.** We removed the words that occurs less frequently(less than ten times). After this step, the number of unique tokens reduced from 375.149 to 45.262.

4.3 Results of LDA and NMF

The outputs of the LDA and NMF models gives us lots of useful information as expected, word distributions over topics and topic distributions over users. However topic and word distributions are high dimensional vectors and hard to interpret. 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] 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.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 used less than ten times in the whole corpus.In order to train faster and see the results more clear. 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, irrelevant or non-sense. So we decided on removing the words that are used less than ten 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.

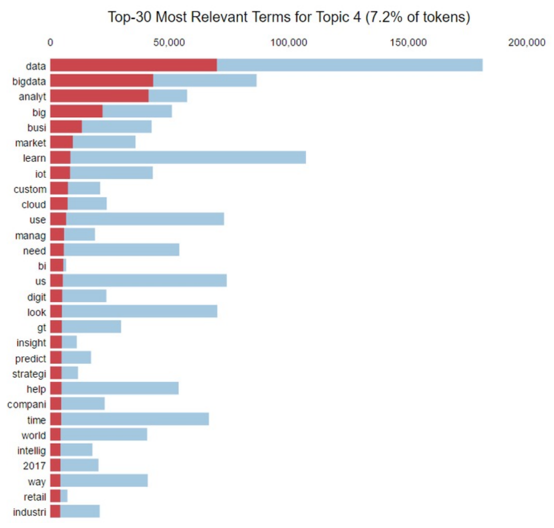
After several runs we decided that training LDA-NMF with this data set with 17 topics gives us better results with respect to dividing into more meaningful topics for this dataset. We have visualized topic distributions of documents and word distribution of topics with LDAvis. To interpret the results better we have created a color coded table of topics and the top 3 words used in the topic. In order to compare users tweeting about different areas, we created charts displaying top 3 topics of users for 6 of the methods we implemented. In the following sections, we compared users and word distributions of topics for different topic modeling implementations.



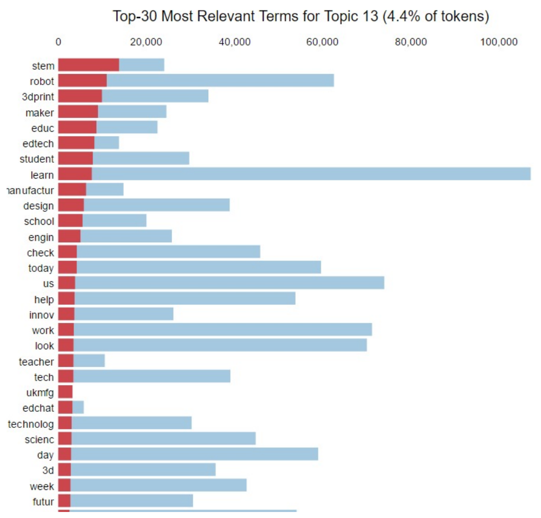
	Daily (1)	Big Data (2)	3D Printing (3)	Tech (4)	Robotics (5)	Depp Learning (6)	EdTech (7)	Politics (8)	ML (9)	Arduino (10)
gensim LDA	think work time	data bigdata analytics	3D print 3Dprint	tech innov* wear*	robot manufac* automat		stem robot 3dprint	us trump world	datasci* data ML*	drone arduino robot
scikit LDA	work time think	data bigdata ai	3Dprint 3D print	startup business market	robot manufac* us		stem code learn			arduino maker project
scikit NMF	work time look	bigdata analytics data	3Dprint 3D print		robot drone kuka	learn deep neural	edtech stem edchat		datasci* ML* DeepL*	pi rasberrypi raspberry
gensim LDA (w2v)	love day today	bigdata data ai	3Dprint 3D printer	market business startup	robot manufac* engineer	learn deep machine	stem code learn	trump year us	datasci* data ML*	arduino robot project
scikit LDA (w2v)	love day us	data bigdata analytics	3Dprint 3D printer	innov* join learn	robot ai techn*	learn deep machine	code stem learn	trump us science	datasci* data ML*	arduino project kit
scikit NMF (w2v)	time day today	bigdata analytics data	3Dprint 3D printer	startup bussiness innov*	robot kuka automat	learn deep neural	stem science women	trump vote obama	datasci* ML* bigdata	arduino kit rasp-pi*

Figure 5: Comparison of top 3 words of different topics and methods

4.3.1 Gensim-LDA



(a) Word distribution of big data topic



(b) Word distribution of tech in education topic

Figure 6: Visualization of word distributions over topics

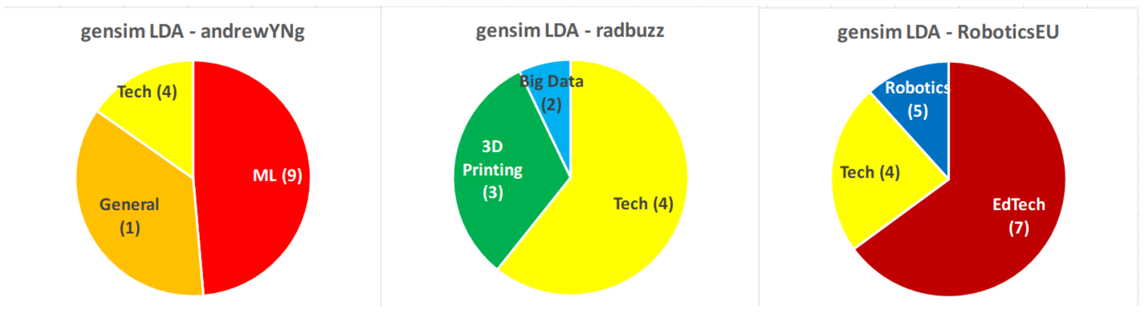
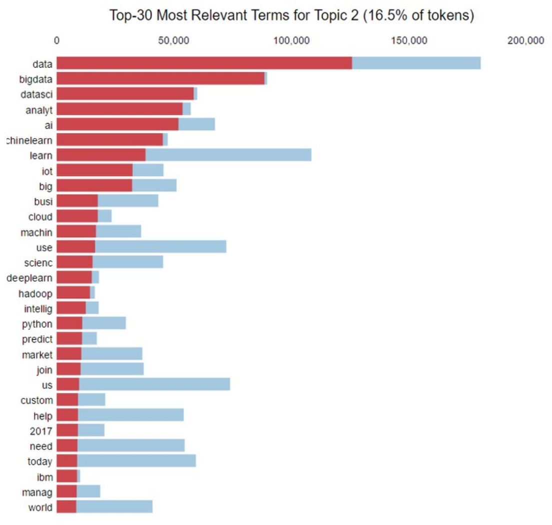


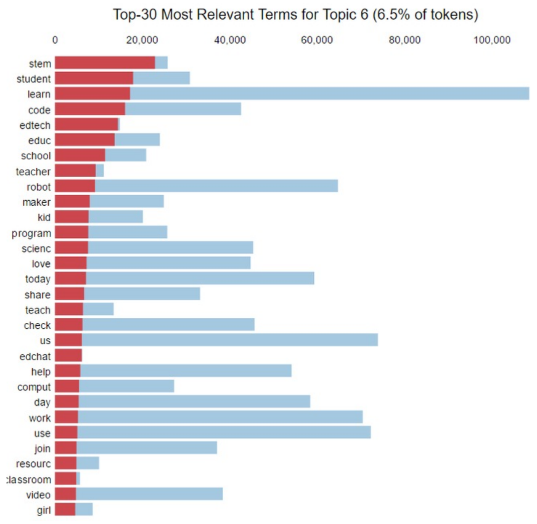
Figure 7: Comparison of top 3 topics of users in different areas

- **Training time:** It took 2688 seconds to train our model on Gensim-LDA with 20 passes and asymmetric alpha as regularization term, it is the slowest of all methods we have implemented.
- **Word distribution of topics:** It successfully finds words that are used in the same topic. There are some unrelated words which couldn't be detected by Gensim-LDA. Even though LDA is successful at determining groups, it is not as much successful at detecting the context of the words.
- **Topic distribution of users:** We compared 3 different users in different areas.
  - **AndrewYNg** tweets about machine learning and is an influencer person in that area.The algorithm successfully finds the topic of this user.
  - **Radbuzz** tweets about 3D printing and machine learning in medical technology. We see that it successfully finds the corresponding topic that is "tech", "printing" and "big data".
  - **RoboticsEU** tweets about robotics and robotics in education. We see that big portion of the graph is tech in education. Even though it contains robotics as third topic, we expected robotics to be the main topic.
- **Topic modeling performance:** We conclude that Gensim-LDA is successful, however not as accurate as NMF. We will see that in the following chapters, using Word2Vector improves the results significantly.

4.3.2 SkLearn-LDA



(a) Word distribution of big data topic



(b) Word distribution of tech in education topic

Figure 8: Visualization of word distributions over topics

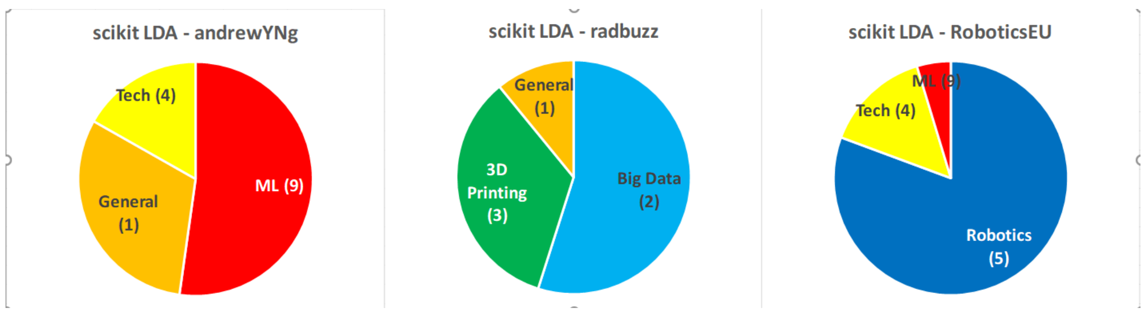
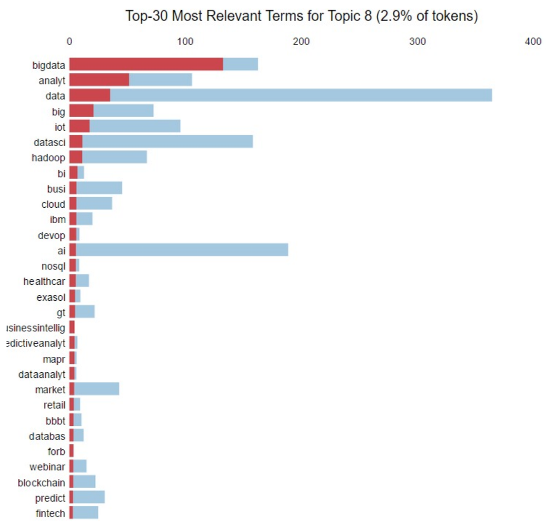


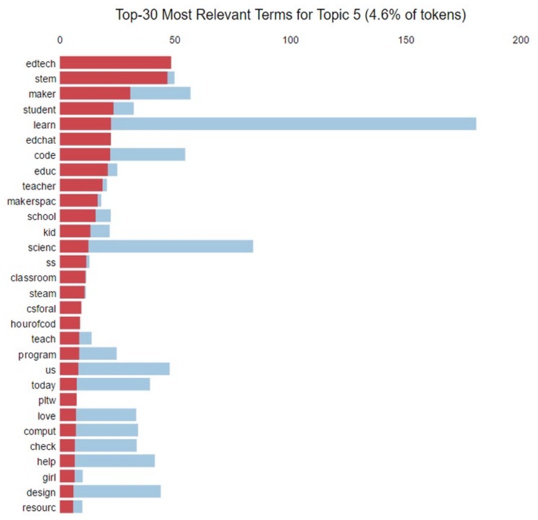
Figure 9: Comparison of top 3 topics of users in different areas

- **Training time:** It took 210 seconds to train our model on SciKit-LDA with 20 passes and online learning, it is 12 times faster compared to Gensim-LDA.
- **Word distribution of topics:** It successfully finds words that are used in the same topic. Most of the words are related to the topic.It is more successful than Gensim LDA in detecting related words.
- **Topic distribution of users:** We compared 3 different users in different areas.
  - **AndrewYNg** tweets about machine learning and is an influencer person in that area.The algorithm successfully finds the topic of this user.
  - **Radbuzz** tweets about 3D printing and machine learning in medical technology. We see that it successfully finds the corresponding topic that is "big data" and "printing" which are better topics compared to Gensim-LDA.
  - **RoboticsEU** tweets about robotics and robotics in education. We see that big portion of the graph is "robotics" which is expected result that we couldn't observe from Gensim-LDA.
- **Topic modeling performance:** We conclude that SciKit-LDA is more successful than Gensim-LDA, however still not as accurate as NMF. .We will see that using Word2Vector improves the results of LDA significantly.

4.3.3 NMF



(a) Word distribution of big data topic



(b) Word distribution of tech in education topic

Figure 10: Visualization of word distributions over topics

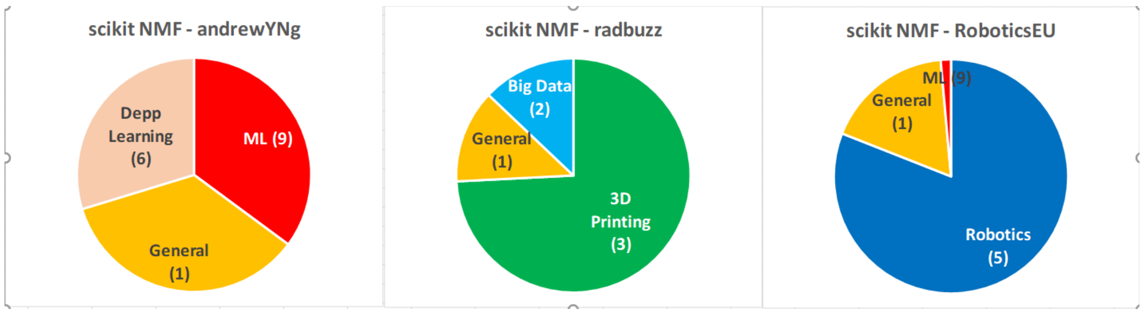
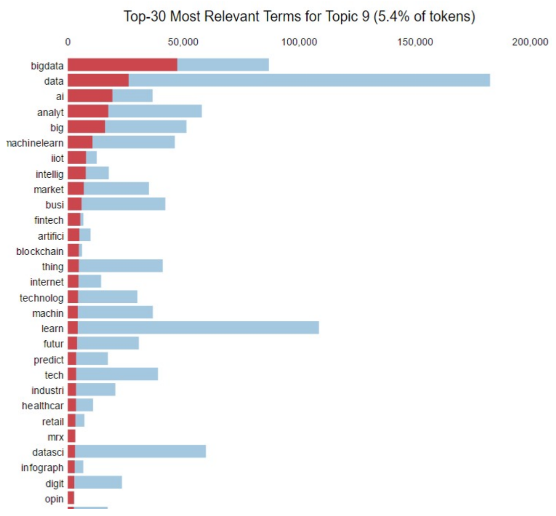


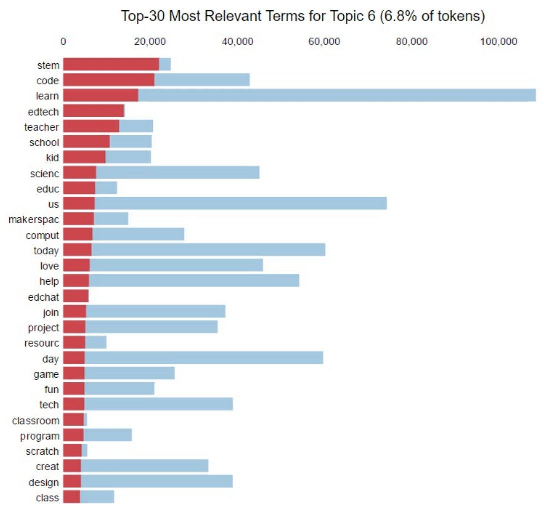
Figure 11: Comparison of top 3 topics of users in different areas

- **Training time:** It took 40 seconds to train our model on Scikit NMF ,it is much more faster compared to LDA implementations.
- **Word distribution of topics:** It successfully finds words that are used in the same topic. Almost all of the words are related to the topic. It also detects the context of the word which improves the accuracy. It is more successful than both LDA models in detecting related words.
- **Topic distribution of users:** We compared 3 different users in different areas.
  - **AndrewYNg** tweets about machine learning and is an influencer person in that area.The algorithm detects the context and differentiates machine learning from deep learning.
  - **Radbuzz** tweets about 3D printing and machine learning in medical technology. We see that it successfully finds the corresponding main topic that is "3D printing" .
  - **RoboticsEU** tweets about robotics and robotics in education. We see that big portion of the graph is "robotics" which is a better result compared to LDA in terms of percentage of the topic.
- **Topic modeling performance:** We conclude that NMF is the most successful method among our implementations. It is successful at detecting words used in different contexts.

4.3.4 Word2Vec-Gensim-LDA



(a) Word distribution of big data topic



(b) Word distribution of tech in education topic

Figure 12: Visualization of word distributions over topics

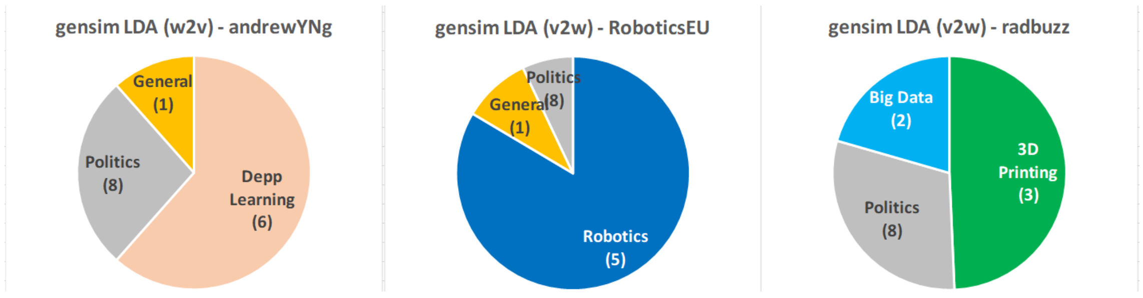
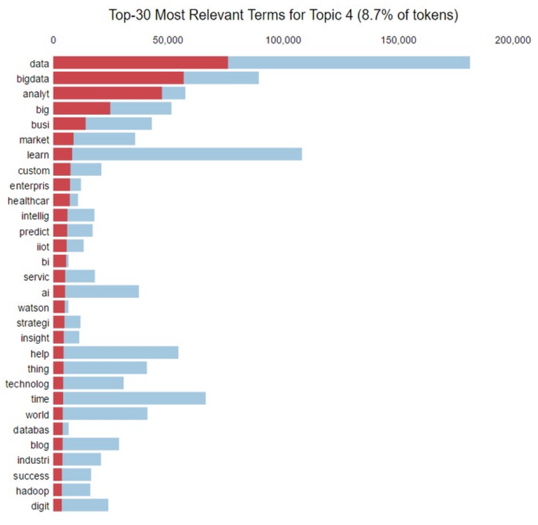


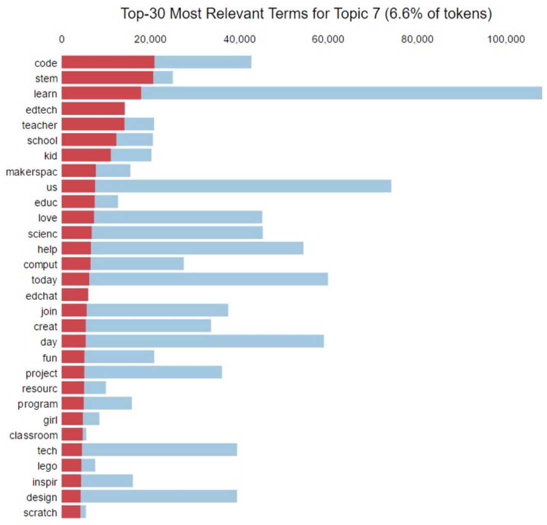
Figure 13: Comparison of top 3 topics of users in different areas

- **Training time:** It took 210 seconds to train our model on Gensim-LDA with 20 passes and online learning. We used Word2Vector to cluster the relative words, then we trained the model which resulted in shorter train time.
- **Word distribution of topics:** It successfully finds words that are used in the same topic. Using word2vector improves the word distribution results significantly. The irrelevant words are vanished.
- **Topic distribution of users:** We compared 3 different users in different areas.
  - **AndrewYNg** tweets about machine learning and is an influencer person in that area.The algorithm extracts deep learning which is the users specialty thanks to word2vector. It assigns "politics" topic which is different from regular LDA.
  - **Radbuzz** tweets about 3D printing and machine learning in medical technology. We see that it successfully finds the corresponding topic that is "3D printing" and "big data" which are way better compared to regular Gensim-LDA.
  - **RoboticsEU** tweets about robotics and robotics in education. We see that big portion of the graph is "robotics" which is expected result that we couldn't observe from regular Gensim-LDA.
- **Topic modeling performance:** We conclude that using word2vector improves the results drastically.The topics are very different and better from regular Gensim-LDA

4.3.5 Word2Vec-SkLearn-LDA



(a) Word distribution of big data topic



(b) Word distribution of tech in education topic

Figure 14: Visualization of word distributions over topics

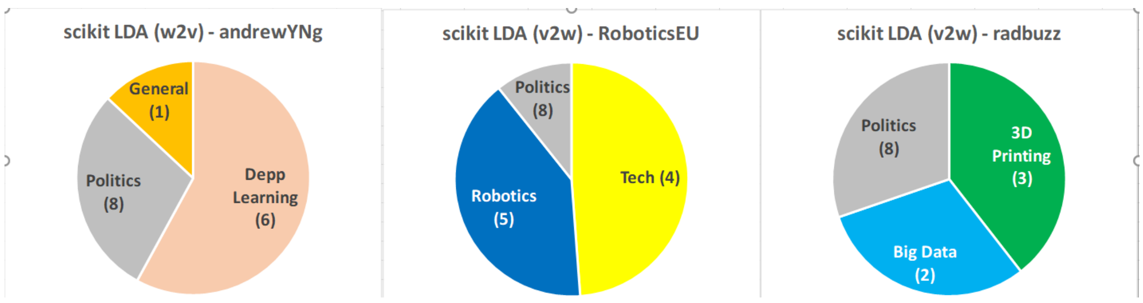
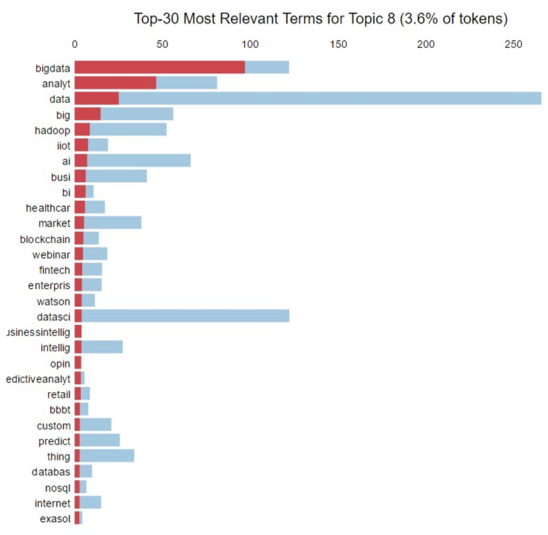


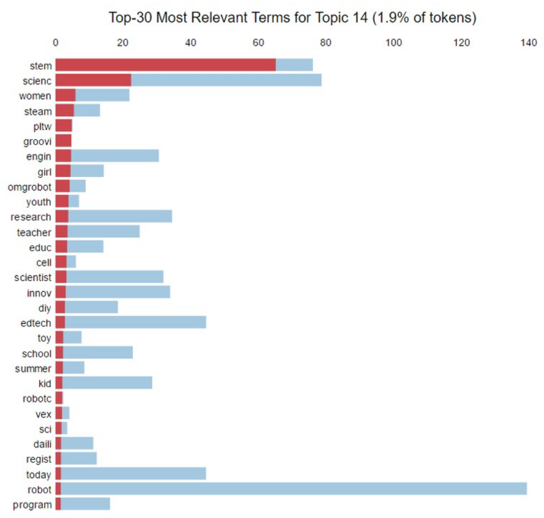
Figure 15: Comparison of top 3 topics of users in different areas

- **Training time:** It took 220 seconds to train our model on Scikit-LDA with 50 passes and online learning. We used Word2Vector to cluster the relative words, then we trained the model.The training time is almost the same even though we increased the iteration number to 50 thanks to word2vector.
- **Word distribution of topics:** It successfully finds words that are used in the same topic. Using word2vector improves the word distribution results significantly. The irrelevant words are vanished.
- **Topic distribution of users:** We compared 3 different users in different areas.
  - **AndrewYNg** tweets about machine learning and is an influencer person in that area.The algorithm extracts deep learning which is the users specialty thanks to word2vector. It assigns "politics" topic which is different from regular LDA.
  - **Radbuzz** tweets about 3D printing and machine learning in medical technology. We see that it successfully finds the corresponding topic that is "3D printing" and "big data" which are way better compared to regular Scikit-LDA.
  - **RoboticsEU** tweets about robotics and robotics in education. We see that big portion of the graph is "tech" but not "robotics". Here using word2vector improved the percentage of "tech" related words, leading us to slightly misleading results
- **Topic modeling performance:** We conclude that using word2vector improves the results drastically. Most of the topics are better than regular Scikit-LDA algorithm.

4.3.6 Word2Vec-NMF



(a) Word distribution of big data topic



(b) Word distribution of tech in education topic

Figure 16: Visualization of word distributions over topics

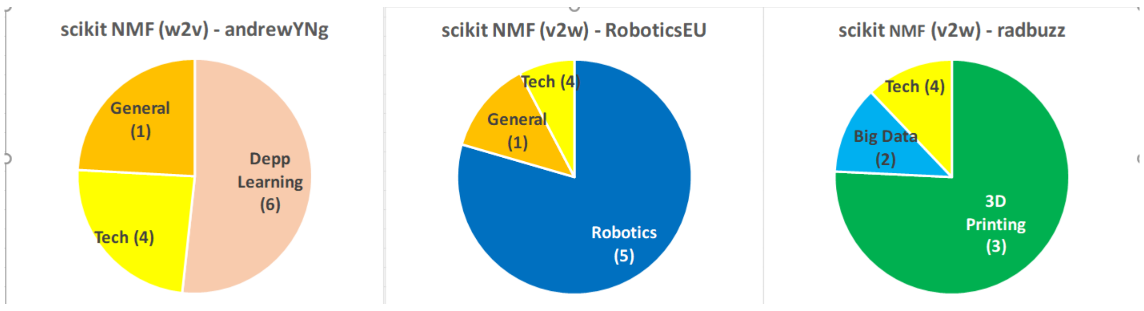


Figure 17: Comparison of top 3 topics of users in different areas

- **Training time:** It took 8 seconds to train our model on Scikit-NMF which is the fastest among all implementations(not including the clustering times after word2vec). We used Word2Vector to cluster the relative words, then we trained the model which resulted in shorter train time.
- **Word distribution of topics:** It successfully finds words that are used in the same topic. Word2Vector causes some differences from regular NMF, some words replaced with some cluster name.
- **Topic distribution of users:** We compared 3 different users in different areas.
  - **AndrewYNg** tweets about machine learning and is an influencer person in that area.The algorithm extracts "deep learning" which is the users specialty thanks to NMF's good context recognition. It assigns "tech" topic which is different from regular NMF.
  - **Radbuzz** tweets about 3D printing and machine learning in medical technology. We see that it is the most successful in finding corresponding topics that is "3D printing", "big data" and "tech".
  - **RoboticsEU** tweets about robotics and robotics in education. We see that big portion of the graph is "robotics" which is expected result that we couldn't observe from regular Gensim-LDA.
- **Topic modeling performance:** We conclude that using word2vector doesn't improve the results too much. Instead it finds more general topics than regular NMF.

The results are promising. All of the methods split the the data into meaningful topics successfully. To observe the relations between each user, we calculated the similarities between each user based on the results using a function of Gensim library and created a distance matrix between users. To visualize this distance matrix we converted it into 2D plot which places similar users closer to each other while showing users that tweets about different topics further. Then to understand this plot better we applied K-means algorithm on that 2D data to observe similar users.

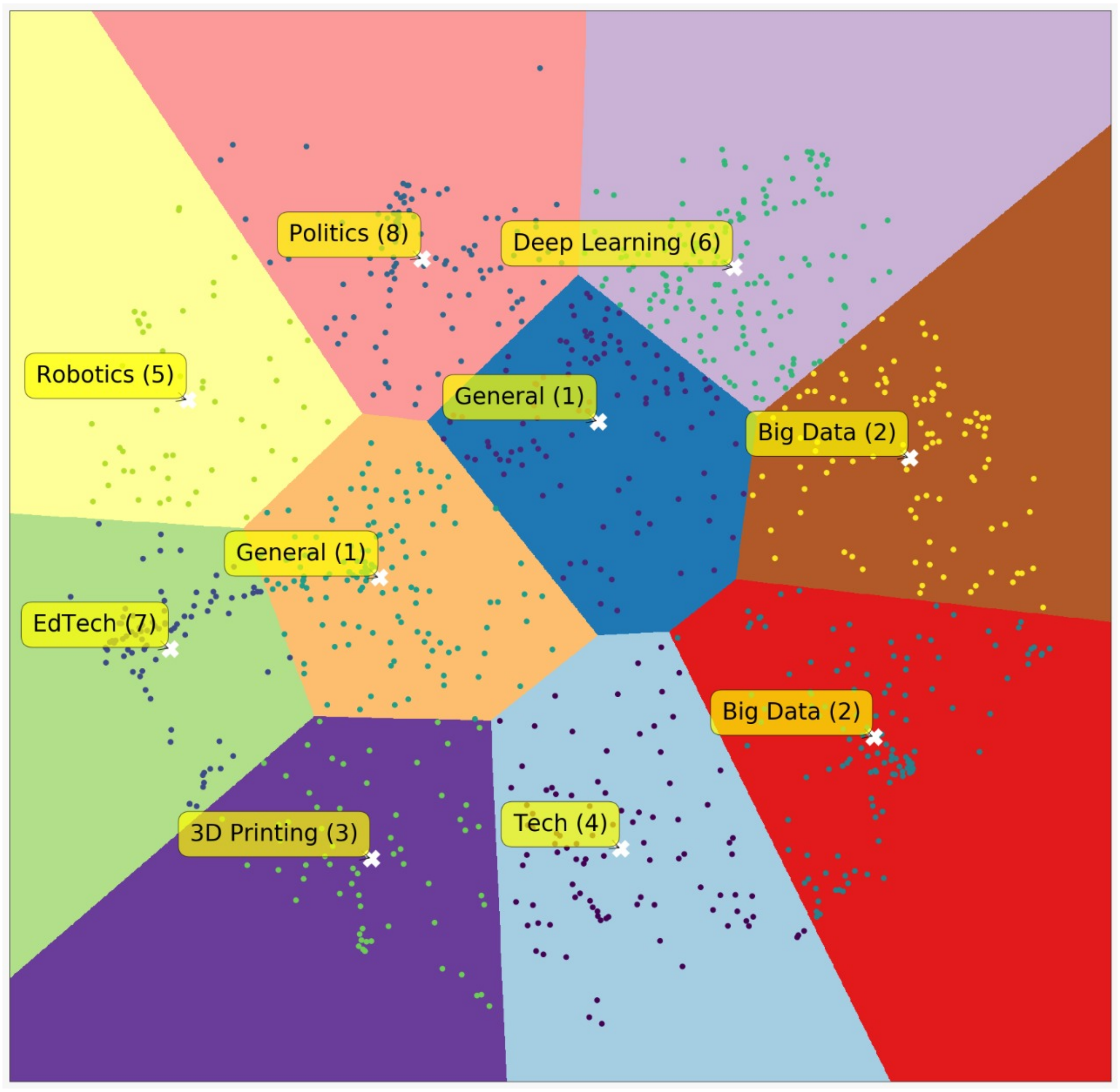


Figure 18: Visualization of users distributed among topics

In Figure 18 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 two close points from the graph shares almost the same topic distribution.

- **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

5 Conclusion

We have experimented on many methods with different parameters. On the way of improving project, we added various NLP filtering to obtain more clear results. After experimenting on small dataset, we created a big useful dataset which consists of computer scientists, makers and technology enthusiasts. While creating the dataset, we produced a tool that finds similar users on Twitter. We implemented various libraries for NMF and LDA to reach the most successful topic modeling program by comparing results. We improved the models by implementing a different approach word to vector. Which significantly improved the results of the LDA.

The hardest part of our project was the evaluation of results. Because all the results we got from topic modeling algorithms needs human interpretation. Thus, to make those interpretation clear and understandable we came up with the idea of color codded charts. Even it is hard to interpret, we got very promising and comparable results. While NMF generally gives better results than LDA; Word2Vec affected both results significantly in capturing the general idea.

All in all, we are able to successfully determine users who tweet about same topic. One can find different datasets with Similar-Twitter and analyze them with our Topic Modeling approaches to create communities in Twitter.

6 Further Work

Tweets may have a wide range of topics which is hard to detect among all of the users. Hashtag's are really important in detecting a tweets topic. Because, the hashtag essentially means the topic of the tweet.

The data set can be extended more to make our model capable of categorizing wider range of tweets.

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.

[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.