ANALYSIS OF TWITTER TO IDENTIFY TRENDS AND INFLUENTIALS WITH A CASE STUDY ON TURKISH TWITTER USERS

BY

GÖKHAN GÖKTÜRK

SUBMITTED TO THE GRADUATE SCHOOL OF ENGINEERING AND NATURAL SCIENCES

IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

ANALYSIS OF TWITTER TO IDENTIFY TRENDS AND INFLUENTIALS WITH A CASE STUDY ON TURKISH TWITTER USERS

APPROVED BY:

Assoc. Prof. Dr. Yücel Saygın (Thesis Supervisor)	[illegible]
Assoc. Prof. Dr. Berrin Yanıkoğlu	[illegible]
Assist. Prof. Dr. Emre Hatipoğlu	[illegible]

DATE OF APPROVAL: 07/08/2014

ANALYSIS OF TWITTER TO IDENTIFY TRENDS AND INFLUENTIALS WITH A CASE STUDY ON TURKISH TWITTER USERS

Gökhan Göktürk

Computer Science and Engineering, Master Thesis, 2014

Thesis Advisor: Assoc. Prof. Dr. Yücel Saygın

Keywords: Social Network Analysis, Sentiment Analysis, Text Classification

Abstract

Social media is one of the largest information flow medium today. Nevertheless, despite its centrality, conventional public opinion research doesn't take social media into account but instead focuses on surveys, polls and interviews. These research methods have their limitations. By nature, even the most meticulously designed survey, for example, is limited by time and seldom bias free.

If properly utilized social media, can address limitations of these shortcomings; Social Media allows us to continuously observe how information flows both temporally and spatially since its users communicate with each other rather than answering survey questions; the data is without experimenter bias and sample size is much larger than of conventional methods. We aimed to show an interdisciplinary work that provides empirical quantifiable answers for social science problems using network analysis and machine learning.

With this aim in mind, this work combines network analysis and sentiment analysis to analyze Istanbul 2014 local elections as a proof of concept. Furthermore, it illustrates the performance of our sentiment analysis system and structural differences between two parties in the event.

TURK TWITTER KULLANACILARINI INCELEYEREK, TWITTER'IN ANALIZI ILE TRENT VE FIKIR LIDERLERININ BULUNMASI

Gökhan Göktürk

Bilgisayar Bilimi ve Mühendisliği, Yüksek Lisans Tezi, 2014

Tez Danışmanı: Assoc. Prof. Dr. Yücel Saygın

Anahtar Kelimeler: Sosyal Ağ Analizi, Duygu Analizi, Yazı Sınıflandırma

Özet

Sosyal medya günümüzde en büyük bilgi akış ortamlarından biri olmasına karşın geleneksel kamuoyu araştırması kendi merkezi önemine rağmen sosyal medyayı gözardı etmektedir. Geleneksel kamu araştırması onun yerine anketlere yönelmektedir. Ama doğası gereği en titiz hazırlanmış anket bile zaman sınırlı ve meyilli olabilmektedir.

Eğer düzgün kullanılabilirse sosyal medya bu sorunları aşmakta yardımcı olabilir; sosyal medya bizim haberleşmeyi ve bilgi akışını zaman ve yer olarak kesintisiz olarak izlememize imkan vererek araştırmacının anket ile meyilli olmayan, hem de daha büyük verisetine ulaşmasını sağlamaktadır.

Bu tezde disiplinler arası bir çalışma göstererek, ağ analizi ve makine öğrenmesi kullanarak,sosyal bilim sorularına empirik ölçülebilir cevaplar vermeye çalıştık. Bu çalışma sosyal ağ analizi ve fikir madenlemesinin birleştirerek, kavram kanıtlamak için 2014 İstanbul yerel seçimlerini analiz etmektedir. Ve, sonuçlarda fikir madenlemesi sistemizin performasını ve iki politik grup arasındaki yapısal farkları sunmaktadır.

to my beloved family…

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF EQUATIONS

CHAPTER 1

Introduction

Today, social media represents of the largest information flow platforms. Contemporary events including Occupy Wall Street, Arab Spring, and Gezi Protests, have shown how much social media can be effective medium for both personal and mass communications.

Nevertheless, despite its centrality, conventional public opinion research does not take social media into account but instead focuses on surveys, polls and interviews. These research methods have their limitations. By nature, even the most meticulously designed survey, for example, is limited by time and seldom bias free. Conventional methods not only show mere snapshots of public opinion but also fail to illustrate how opinion can change along the information flow and events. Even though polling overcomes a few of these problems, polls can be only applied after a hypothesis is formed, and after significant delays are an intrinsic part of polling process. In addition, both methods can only reach very limited sample of population.

If properly utilized, social media can address limitations of these shortcomings; social media allows us to continuously observe how information flows both temporally and spatially since its users communicate with each other rather than answering survey questions; the data is without experimenter bias and sample size is much larger than of conventional methods.

Even though, public opinion research on social media can solve many problems of traditional opinion research, it brings new problems; Due to size of the data, the analysis requires scale-able algorithms and software that can handle Big Data. Also, the obtained communications data is unstructured and not quantifiable without the use of Sentiment Analysis/Opinion Mining.

In this work, we aimed to create algorithms and a software system to process big social media data that is scale-able and with as little supervision as possible. So that, the concept system can works on real world events.

The resulting system is tested on the Istanbul 2014 local elections. The Istanbul local elections are chosen due to its spatial closeness to this researcher. Also, The Istanbul local election was worthy of analysis, because Istanbul local elections are central for Turkish politics, it is the first election in Turkey that most prominent candidates focused on social media, and also media censorship and self-censorship made social media an important alternative for conventional media.

This thesis organized as follows: in chapter 2 we provide a background to social networks, influentials, and sentiment analysis. The work done to generate, manage, and analyze social networks and its structure is described in chapter 3. Also, detailed description of centrality measures is given in this chapter. The sentiment analysis of the content retrieved from the social network, and detailed description of the techniques is described in chapter 4. Then, chapter 5 gives details of the system implementation. Results from the analysis are available under chapter 6. Finally, chapter 7 gives a conclusion about the work.

CHAPTER 2

Background Information

2.1. Social Networks

2.1.1. Social Media

Social media is an internet-based information sharing and consumption platform; where the content is authored, shared, and exchanged by its users. It differs from traditional media in many ways including, content, authorship, reach, frequency, delay, and perpetuity. Content on social media, generated in a fashion much more rapid, reaches much larger audience with immediacy. Social Media platforms can be listed as internet forums, blogs, micro-blogs, collaborative encyclopedias, social networking sites, virtual social games, social content sharing, and social bookmarking.

According to We Are Social, a global social agency, research in 2014, 2.5 billion internet users active in world, and 1.8 billion of these users are active on Social Media.

On average, 4.8 hours spent daily online on personal computers and with 2.1 hours spent daily online via mobile devices. (We Are Social, 2014)

2.1.2. Social Networks

Social Networks are graph structures that consist of social actors/agents and relationships between these actors. Social networks are often self-organizing, complex and emergent. (Newman, Barabási, & Watts, 2006) We will represent a graph as G = (V, E) where V is the set of vertices and E is the edges that are associated with these vertex set. n = |V| denotes the number of vertices and m=|E| denotes number of edges. The distance between all pairs of two directly connected vertices is assumed to be constant. Edges will be represented as E_{ij} if there is an edge from vertex i to vertex j, value of E_{ij} will be 1, otherwise 0. In this work, the graph is assumed to be connected if, not each separate graph is individually handled. Our scope in this work only considers unweighted simple graphs.

Most of the social networks are not randomly distributed networks, where relationships between actors are distributed randomly but Scale-free networks, where relations are distributed according to a power law. (Barabási, 2003) In a Scale-Free network, the probability of having k out-going edges in a vertex is $P(k){\sim}\ k^{-\gamma}$ where γ is the parameter that shapes the distribution. Twitter is assumed to be a scale-free network that follows the power law. Also, we have seen that the Turkish Twitter social graph forms a scale-free network.

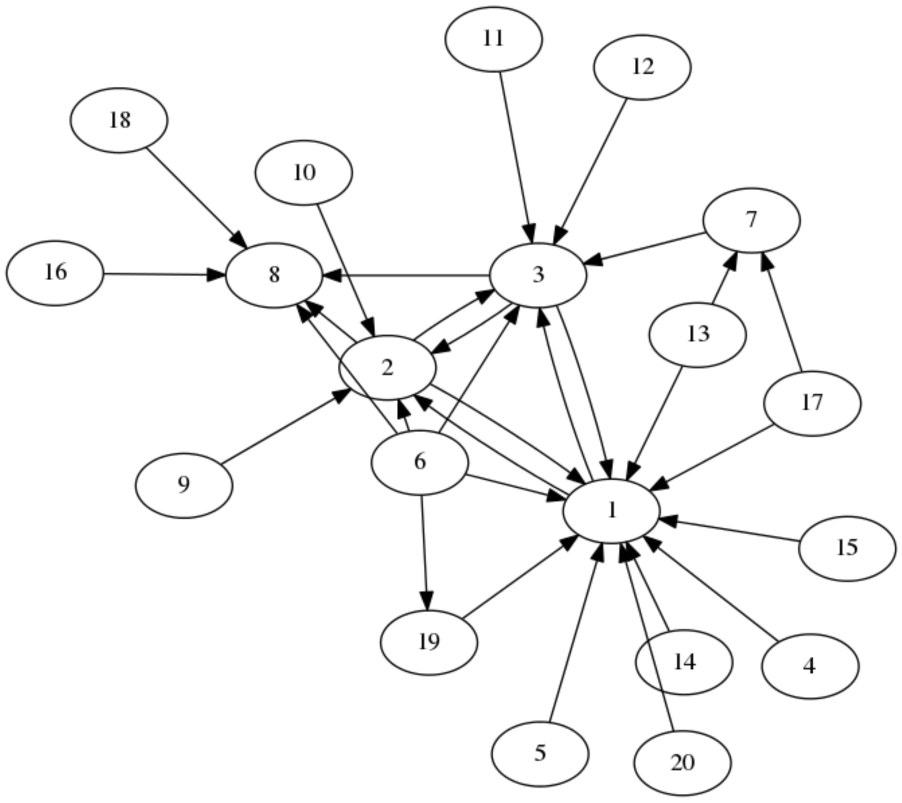


Figure 1 A scale-free network.

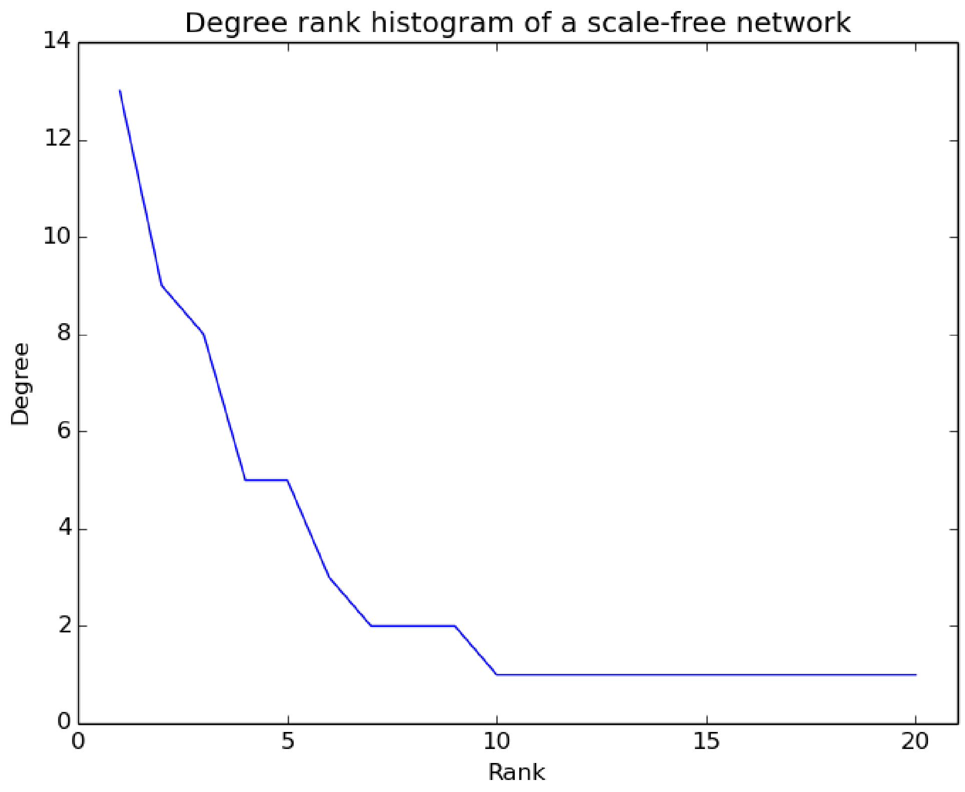


Figure 2 Degree rank histogram of a small scale-free network.

2.1.3. Influentials

Influentials are opinion shapers in society. Even though the definition of opinion leader differs among researchers, many note that opinion leaders are influential people that are more able to affect public opinion than others. Katz and Lazar define as influentials as "the individuals who were likely to influence other people in their immediate environment" (Katz & Lazarsfeld, Personal Influence, 1970) According to Watts and Dodds, influentials may play a critical role in driving large cascades as the early adopters, who make up the critical mass via which local cascades become global (Watts & Dodds, 2007)

The traditional approach for understanding opinion leaders and their social influence is based on Two-step flow of communication model. Two-step flow of communication model hypothesizes that option leaders shapes their opinion according to mass media, then influences wider population around them. (Lazarsfeld, Berelson, & Gaudet, 1948)

How opinion leaders influence people is often complex and multifarious. The traditional communication model seems obsolete in age of online social networks where consumers of the media are also its content creators and broadcasters/diffusers. By the same reason, residing on important communication paths is also an import factor of influentials. We define influentials as a small group of people that shapes other people's opinion via both direct communication and participating in communication paths.

2.2. Sentiment Analysis

Sentiment Analysis is measurement of people’s attitude according their writings with respect to some topic or context. The measured attitude of text may include emotions, ratings, and perceptions. Sentiment Analysis relies on Natural Language Processing and Machine Learning to make sense of documents and classify them accordingly.

Sentiment analysis is onerous due to natural languages’ high complexity; expressions are often hard to quantify and similar ideas could be written in so many ways that is hard for a computer to analyze pattern in the text. In addition to these difficulties, sentiment can be expressed with no apparent positive or negative words.

With the rise of social media and online abundance of ratings and reviews, online opinion has become an important issue for business and politics. Sentiment Analysis can measure public opinion and provide intelligence using a large amount of data available online. Recently, re-election campaign of Barack Obama has made use of sentiment analysis to organize millions of e-mails and messages according their issue topics to keep supporters engaged with the campaign. (Schectman, 2012) Another example could be Starbucks; Starbucks makes use of sentiment analysis to answer customer complaints at extraordinary rate. (Bort, 2012)

CHAPTER 3

Social Network Analysis

In the social network analysis part of this work, we have aimed to reach subset of users we are interested. Then, we tried to find influential users by calculating several centrality measures. Due to computational complexity of Betweenness Centrality and Eigenvector Centrality measures, we have used estimation methods that are much more feasible.

This work focuses on Turkish Twitter users due to rapid rise of online social network use in Turkey and spatial proximity to researcher.

3.1. Retrieving Turkish Twitter Social Graph

We wanted to reach Turkish Twitter Users to observe information flow on between them. One way of the obtaining such data is snowballing. The method we had used, snowballing, is simply a dept limited breadth-first search method.

First, in order to obtain Turkish Twitter users, we have selected approximately 6000 Twitter users that are known to be operating in Turkey as seed set to reach Turkish Twitter users; this seed set is generated using a few randomly selected Turkish user’s friends and followers on Twitter. Then, we have filtered resulting user set with human supervision to obtain only Turkish speaking users.

Screen Name	Real Identity
CMYLMZ	Cem Yılmaz, Stand-up Artist
Cbadbullahgul	Abdullah Gul, Turkish President
RT_Erdogan	Tayyip Erdogan, Turkish PM
Komedyieni	Ata Demirer, Actor
Hulyavsar	Hulya Avsar, Celebrity
DemetAkalin	Demet Akalin Kurt, Singer
GalatasaraySK	Galatasaray, Sports Club

Table 1 Sample of users from the seed set.

Second, we have followed connections, both followers and friends, of the seed set to reach users in first degree of separation. After, applied the same process to first degree set to reach our seed set's two degrees of separation. We reached approximately 15 million Twitter users' account information. Then, we retrieved those users' profiles.

Next, we have trained a decision tree classifier to identify Turkish users, using language, time zone, location, and last status message as features. Using a decision tree on these parameters we were able to extract approximately 10 million Turkish users. Most of the Turkish users were identified by few most common parameters such as the interface language as Turkish, or location names with in Turkey. We were able to build a classifier to successfully extract Turkish users using most common identifiers. In this case study, we have omitted links connected to non-Turkish users, spam bots, or inactive accounts that haven't got any activity in last two months.

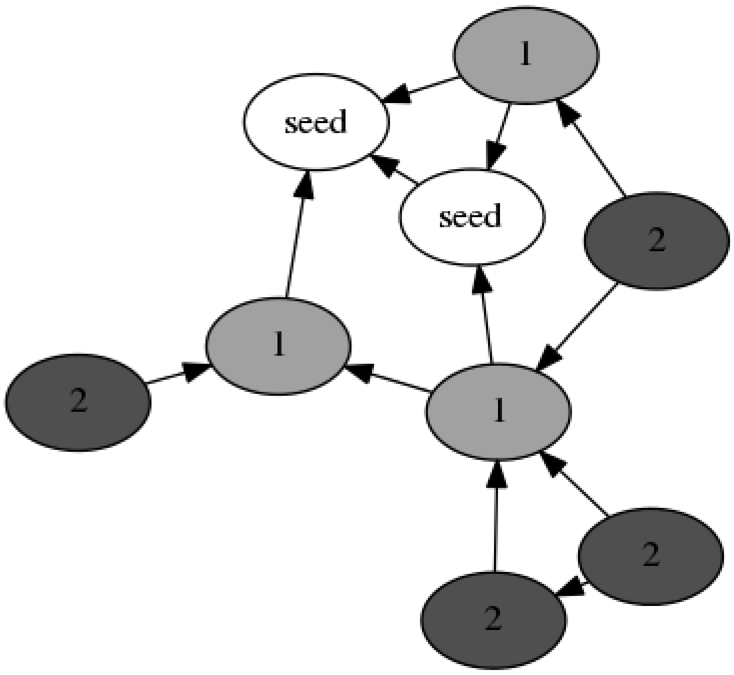


Figure 3 Two-degrees of separation, nodes are labeled as their separation degrees.

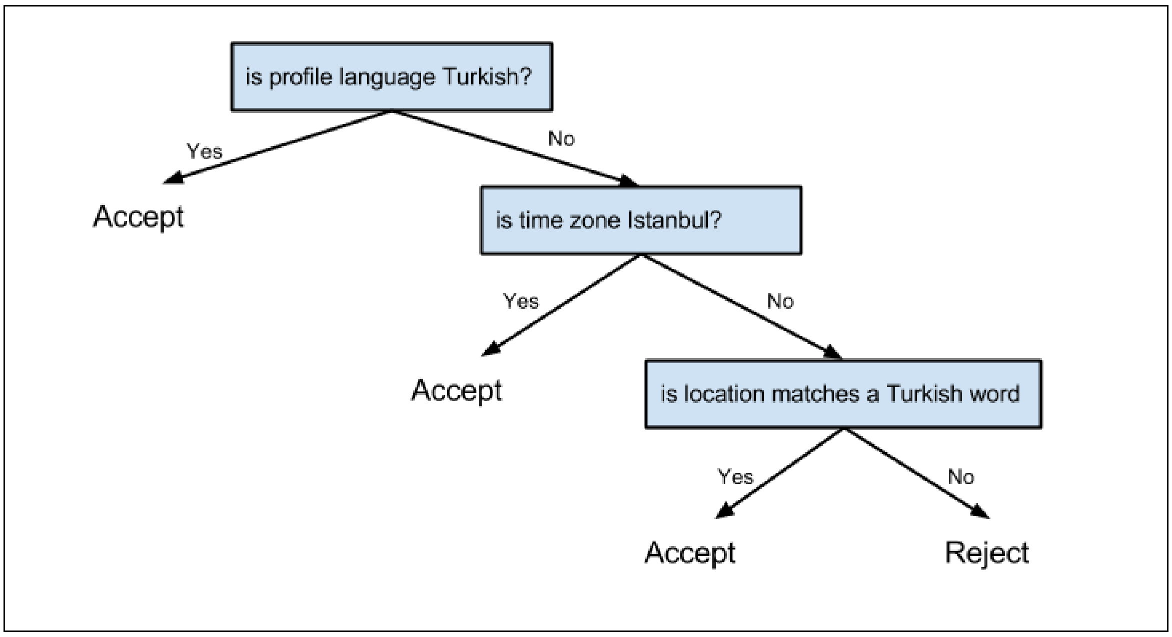

is profile language Turkish?
Yes
No
Accept
is time zone Istanbul?
Yes
No
Accept
is location matches a Turkish word
Yes
No
Accept
Reject

Figure 4 Decision Tree Model for Turkish Users (Simplified).

After that, we have crawled who these Turkish Twitter users are following. Due to rate limiting in Twitter, we have created a Twitter app that allowed us to utilize many users' rate limit. Then, we assigned look-up tasks for each user of our app.

Finally, the resulting social network gave us over 951 million connections of which more than 451 million are connected to our Turkish Twitter user set. In our observations, follower numbers were highly skewed and the resulting graph is a scale-free graph.

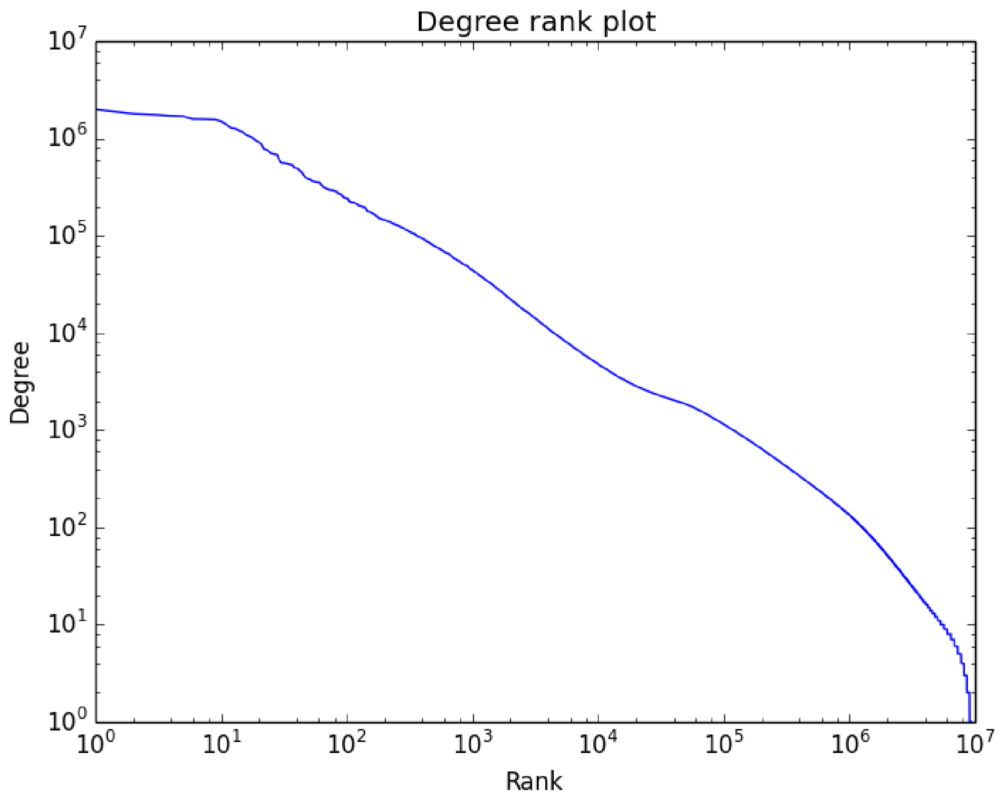


Figure 5 Degree Histogram of Turkish Twitter Network in log scale in both axes.

3.2. Indentifying Influentials

We argue that opinion leaders shape public opinion in online social platforms through two focal means; generating information by sharing ideas, and/or affecting flow information by propagating certain information and not propagating other. The literature review revealed that it is possible to find different characteristics of users by network centrality measures including; Degree Centrality, Betweenness Centrality, and Eigenvector Centrality.

3.2.1. Degree Centrality

The primer of centrality measures is degree centrality, which is the most intuitive and straight forward interpretation of importance in a graph; centrality identifies the number of connections associated with an individual account, or in network analysis terms, a node. The degree centrality can be interpreted as immediate interactions available to the node.

The degree centrality measure is often calculated as two separate measures: in-degree and out-degree centrality. In-degree is the number of connections going into a vertex and out-degree is the number of connections going out of a vertex. On simple directed graphs, Degree Centrality ranges between 0 and the cardinality of graph. The degree centrality has linear time complexity with respect to the number of edges, which is in $O(E)$, so we haven’t applied any sampling or convergence methods, because there is not much performance improvement yet information loss is significant. The in-degree centrality,C_{in} and out-degree centrality,C_{out} could be computed as follows;

$$C_{in}(i) = \sum_{i \neq j} e_{ji}$$

Equation 1 In-degree Centrality

$$C_{out}(i) = \sum_{i \neq j} e_{ij}$$

Equation 2 Out-Degree Centrality

The degree centrality, C_{degree}, is summation of these measures, and it could be computed as follows;

$$C_{degree}(i) = C_{out}(i) + C_{in}(i)$$

$$C_{degree}(i) = \sum_{i \neq j} e_{ij} + \sum_{i \neq j} e_{ji}$$

Equation 3 Degree Centrality

3.2.2. Eigenvector Centrality

Eigenvector centrality is also a measure of vertex importance. Eigenvector centrality, not only accounts for the important of the vertex, but also that of the importance adjacent vertices. This metric assumes that influential people are more likely to connect with other influential people

Each vertex’s eigenvector centrality equals the sum of the eigenvector centrality of all adjacent vertices. This measure can be calculated by converting a graph to an adjacency matrix and calculating the largest eigenvector. Eigenvector Centrality ranges between 0 and 1. This approach has cubic time complexity with respect to the number of vertices, which is $O(N^3)$. This approach is not feasible for very large graphs due to a long running time. However, eigenvector centrality can be estimated using power iteration method.

$$C_{eigen}(i) = \frac{1}{\lambda} \sum_{t \in \{j | e_{ji} = 1\}} C_{eigen}(t)$$

$$C_{eigen}(i) = \frac{1}{\lambda} \sum e_{ji} C_{eigen}(j)$$

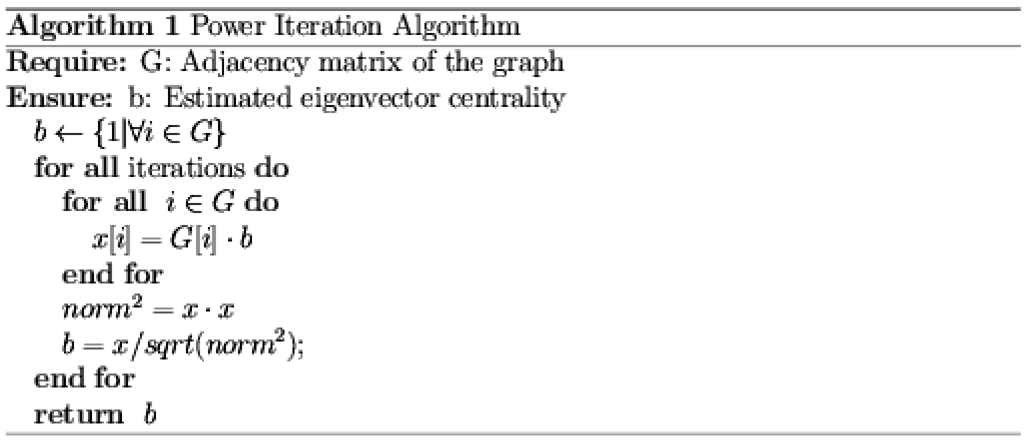
Equation 4 Eigenvalue Centrality

where λ is greatest eigenvalue and $t \in \{j | e_{ji} = 1\}$ is every vertices that are adjacent to vertex i. (Newman M. E., 2008)

3.2.2.1. Power Iteration

The power iteration method estimates eigenvector centrality by the following steps; initially eigenvector centrality value is assigned as 1.0, then in each iteration all vertices' centrality are assigned as the sum of all adjacent vertices' centrality from the previous iteration. The power iteration method can be terminated after a predefined number of iterations and/or a predefined tolerance value is satisfied between iterations.

Estimation would have linear time complexity with respect to the number of edges, which is in $O(E)$. The number of iterations required depends on the graph's diameter and tolerance. However, the number of iterations should not be too big for real world applications according to the Small World theorem.



```
Algorithm 1 Power Iteration Algorithm
Require: G: Adjacency matrix of the graph
Ensure: b: Estimated eigenvector centrality
  $b \leftarrow \{1|\forall i \in G\}$
  for all iterations do
    for all $i \in G$ do
      $x[i] = G[i] \cdot b$
    end for
    $norm^2 = x \cdot x$
    $b = x/sqrt(norm^2)$;
  end for
  return $b$
```

Figure 6 Power Iteration Algorithm.

3.2.3. Betweenness Centrality

Betweenness Centrality is a measure of vertex importance in networks. It commonly used in social network analysis. It is a shortest-path distance based method that shows how much a vertex is contained in the shortest paths connecting pairs of other vertices.

Betweenness Centrality counts every shortest path in graph G. The Betweenness centrality score of the each vertex is calculated as the number of the vertex occurrence on shortest-paths. Betweenness Centrality ranges between 0 and number of shortest paths available. The number of shortest-paths starts from vertex s and ends at vertex t, is denoted as σ(s,t) , if the path contains an intermediate vertex v, it is denoted as σ(s,t | v) where s and t are members of the vertex set V. The Betweenness centrality, C_b, is computed as follows (Brandes, 2001);

$$C_b(v) = \sum_{s \neq v \neq t} \frac{\sigma(s,t|v)}{\sigma(s,t)}$$

Equation 5 Betweenness Centrality

3.2.3.1. Sampling Betweenness Centrality

Even though the graphs we are expecting are sparse, the computational time complexity of the best algorithm, Brandes' Betweenness centrality algorithm, is $O(nm)$. Our experiments showed that on our 16 core server, our homegrown heavily optimized Brandes' Betweenness centrality implementation would have an estimated running time of 11 months.

To overcome long running time, we have used sampling method by Brandes et al.(Brandes, 2001) Brandes' work reformulates the Betweenness centrality by bringing new dependencies;

$$\sigma(\mathbf{s},\mathbf{v}|\mathbf{t}) = \frac{\boldsymbol{\sigma}(\mathbf{s},\mathbf{t}|\mathbf{v})}{\boldsymbol{\sigma}(\mathbf{s},\mathbf{t})}$$

$$\delta(\mathbf{s},\mathbf{t}|\mathbf{v}) = \sum_{s \neq v \neq t} \boldsymbol{\sigma}(\mathbf{s},\mathbf{t}|\mathbf{v})$$

Equation 6 Dependency for Betweenness Centrality

Then, Betweenness centrality can be represented as follows;

$$\mathbf{C_b}(v) = \sum_{s \neq v} \frac{\boldsymbol{\sigma}(\mathbf{s},\mathbf{v}|\mathbf{t})}{\boldsymbol{\sigma}(\mathbf{s},\mathbf{v})}$$

Equation 7 Reformulated Betweenness Centrality

After that, the whole problem could be solved as a single source shortest path problem for all vertices in V. After reformulation, it is possible to select a subset S, of all vertices

V with size of k=|S|, where each element in subset S, namely pivots, contribute to estimation of the C_b,. Estimated Betweenness centrality can be computed as follows;

$$\mathbf{C_b}(v) = \frac{\mathbf{1}}{k}\frac{n}{n-\mathbf{1}}\sum_{s \in S} \boldsymbol{\delta}(\mathbf{s}|\mathbf{v})$$

Equation 8 Sampling Betweenness Centrality

Assuming all single source shortest path computation in the algorithm are random experiments and the variables are independent identically distributed. The error of estimating Betweenness centrality using a pivot set with size of k and error margin ζ can be calculated using Hoeffding's formulas;

$$\zeta = \varsigma(n-2)$$

$$M = \frac{n}{n-1}(n-2)$$

$$P(|C_b - E(C_b)| \geq \zeta) \geq e^{-2k\frac{\zeta^2}{M}}$$

Equation 9 Hoeffding's formulas for finding sampling error

Pivot selection is also important, due to different selection strategies can affect convergence rate. (Schultes, Sanders, & Schultes, 2008) However, selecting non-random pivots may make each pivots contribution dependent to each other, and Hoeffding's formula would not hold on a dependent experiment.

Algorithm 2 Sampling Betweenness centrality

```
Require: G: Adjacency matrix of the graph
Ensure: $C_b$: Estimated Betweenness centrality
  $C_b \leftarrow 0, i \in G$
  for all $s \in V$ do
    $S \leftarrow$ empty stack
    $P[w] \leftarrow$ empty list, $w \in V$
    $\sigma[w] \leftarrow 0, t \in V$
    $\sigma[s] \leftarrow 1$
    $Q \leftarrow$ empty queue
    enqueue $s \rightarrow Q$
    while Q not empty do
      dequeue $v \leftarrow Q$
      push $v \rightarrow S$;
      for all neighbor w of v do
        if d[w] ¡ 0 then
          enqueue $w \rightarrow Q$;
          $d[w] \leftarrow d[v] + 1$
        end if
        if d[w] = d[v] + 1 then
          $sigma[w] \leftarrow sigma[w] + 1$
          append $P[w] \leftarrow v$
        end if
      end for
    end while$\delta[v] \leftarrow 0, v \in V$
    while S not empty do
      pop $w \leftarrow S$
      for all $v \in P[w]$ do
        $\delta[v] \leftarrow \delta[v] + \frac{\sigma[v]}{\sigma[w]} \cdot (1 + \delta[w])$
      end for
      if w $\neq s$ then
        $C_b[w] \leftarrow C_b[w] + \delta[w]$
    end while
  end for
  return $b$
```

3.2.4. Identifying Opinion Shaper Roles using Centrality Measures

The actors with high Indegree Centrality are popular leaders that can disseminate ideas directly. On the other hand, the actors with high Eigenvector Centrality can reach a larger group than their surroundings if their ideas can infect their surroundings, and the people around them also disseminate the same idea. Besides, the actors with high Betweenness centrality play another role; they have much more control of the communication paths.

	Low In-Degree	Low Eigenvector	Low Betweenness
High Indegree		Marginal/Isolated leader: Individual is embedded in cluster that is far from the rest of the network (or key actors)	Conventional Ineffective: Individual's connections are redundant--communication bypasses them
High Eigenvector	Ghost opinion shaper: Low number of connections, but connected to important actors		Specialized opinion shaper: Individual might have unique access to central actors
High Betweenness	Influence-broker: Individual’s few ties are crucial for network flow	Gatekeeper: Gatekeeper to central actors	

Table 2 Actors’ roles in a network according to their centrality measures adapted from (Moody, 2012)

Marginal/Isolated Leaders are popular in some isolated group, but due to their group’s low connectivity, their Eigenvector Centrality is low and their ideas have difficulty in passing outside of the their surroundings.

Conventional/Ineffective Opinion Shapers are popular but their connections are redundant, the communication by-passes them.

Ghost Opinion Shapers are interesting because they do not directly disseminate ideas much but since they are able to sway in important actors, Ghost Opinion Shapers can reach larger audience.

Specialized opinion leaders are influential within their respective surroundings, but are weakly tied to others. Specialized opinion leader’s ideas circles through in the same crowd rather than reaching beyond of their surroundings.

Influence brokers are not directly influential but they control the flow of information, and their small number of connections is important to the communication paths.

Gatekeepers are connected to important actors; they are on important communication paths which allow them to control influence of other important actors.

CHAPTER 4

Sentiment Analysis

The sentiment analysis task is consists of several stages. Initially, some pre-processing is done on the text to improve quality of the input data; since data retrieval processing is often very forgiving and less strict to collect data as much as possible. Secondly, features are generated since most of the analysis algorithms can only work on structured data. Instead of giving arbitrary length character sequence to machine learning algorithms, a fixed sized numerical and/or nominal vector are much easier to utilize. After that, the resulting feature vectors are fed along with their corresponding labels to train classifiers. Finally, classifiers are tested using validation techniques such as cross-validation like in this work.

4.1. Pre-processing

The status messages obtained from Twitter often includes spelling errors. The Turkish status messages are often uses English letters that are looking similar. The status messages also may include hash tags or concatenated words which are harder to make sense computationally without preprocessing. In addition, consecutive punctuation marks, emoticons, are used show emotions but often not separated from the actual words with white spaces.

In pre-processing stage we have applied following transformations to overcome these problems;

- Asciification: We converted each non-English letter to a similar English variant due to common typing problems regarding mobile users. E.g. letter Ç converted to C, or letter İ converted to I
- URL removal: We removed any URLs in tweets to clean up unmatchable tokens
- Fix camel case hash tags: We split tokens before case change to match words in hash tags or similar uses of camel case.
- Group consecutive punctuation marks: We group words with consecutive punctuation marks to catch common tokens such as emoticons

Lowercase: We converted all letters to lowercase so matching is case insensitive.

Status Message	After Preprocessing
Örnek Tweet #SomeHASHTag t.co/FOOBAR	['ornek', 'tweet', '#', 'some', 'hash', 'tag']
Değistirdiğime çok sevindim :)	['degistirdigime', 'cok', 'sevindim', ':)']

Table 3 Preprocessing Examples

4.2. Feature Extraction

Feature extraction is transformation of arbitrary data to more meaningful, less redundant and clean form that is easier to be utilized by Machine Learning algorithms. The character or word sequence is not directly meaningful to computers; the sequence cannot be fed to the machine learning algorithms. The algorithms require rather fixed sized numerical feature vectors for each instance in the dataset. Even-though, algorithms such as Recursive Auto-Encoders and Recurrent Neural Networks can deal with arbitrary length, the input in each state should be a meaningful fixed sized numerical vector.

Feature extraction from text has few stages; sequence of symbols should be tokenized, then the token should be vectorized, finally the vectors can be normalized if needed.

In our case, most of the tokenization work is already done on pre-processing stage, the stateful regex substitutions allows us to tokenize the strings by simply splitting them on white spaces.

For vectorization, all methods except Recursive Auto-Encoder use Bag-of-words approach to represent status messages. Each token is indexed and represented as a sparse vector. Three BOW vector representations have been used including, TF vector, binarized TF vector, and TF/IDF vector. Recursive Auto-Encoder method used its own word embedding. The word embedding represents each word in a numerical space. The word embedding used in Recursive Auto-Encoder in this work, is a meaningful n-dimensional numerical mapping that an Auto-Encoder can make sense and combine with other words.

4.2.1. Term Frequency

Term Frequency vector contains the number of tokens' occurrences for each document. This approach allows us to represent unstructured text in a structured way, which we have applied common statistical operations. Due to the large size of tokens we have stored TF values as sparse vectors.

Also, many normalization and smoothing methods could be applied to increase performance and generalization of classifiers, such as; binarization where all values larger than 1 regarded as 1, or weighting frequency values using inverse document frequencies.

4.2.2. TF-IDF

In text classification, common words that occurred in too many documents might hold less information than less frequent words. To overcome this problem, weighting with inverse document frequencies is commonly used.

$$\mathrm{TFIDF}(t, D) = \mathrm{TF} * \log\left(\frac{N}{|\{d \in D, T \in d\}|}\right)$$

Equation 10 TF-IDF

where N is total number of documents in the data set and $|\{d \in D, T \in d\}|$ is number of documents where the term t appears.

4.3. Classification

4.3.1. Multinomial Naive Bayes Classifier

Naive Bayes Classification is a machine learning method that assumes given a class, each feature is independent. The Naive Bayes classifies documents by probability estimation due to its distribution. Multinomial Naïve Bayes uses the multinomial event model (Rennie, Shih, Teevan, & Karger, 2003); the feature vector of Multinomial Naïve Bayes Classifier represents multinomial distributed events. When feature vector F is observed, the likelihood estimation according to Multinomial Naïve Bayes is

$$p(F|C) = \frac{(\sum_i F_i)!}{\prod_i F_i!} \prod_i {p_i}^{F_i}$$

Equation 11 Maximum Likelihood Estimation for Multinomial Naïve Bayes Classifier

Often, instead of term frequency, pseudo-counts are used because, when a feature value does not appearing in a given class, the probability estimation would be zero. This special case of pseudo-count for Multinomial Naïve Bayes is called Additive Smoothing. Instead of F_i , smoothed estimation $\tilde{\theta}_i$ is used where μ is incident rates, α is smoothing parameter, and $\alpha > 1$.

$$\tilde{\theta}_i = \frac{x_i + \alpha}{N + \frac{\alpha}{\mu_i}}$$

Equation 12 Formula of Smoothed estimation $\tilde{\theta}_I$.

4.3.2. Linear Support Vector Machine

Support Vector Machines are discriminative classifiers that divide sample space into classes with hyper planes using instances as support vectors. Using class labels as -1 and 1, SVM tries to find maximum-margin hyper plane which separates data point's x_i to their classes, y_i.

The charm of Support Vector Machine is that, it allows getting a complex model with a simple Support Vector Machine. The Support Vector Machines, instead of generating a model via aggression, generation, or reproduction, uses the data points from the training dataset to produce the model. Subset of the data points are selected to be Support Vectors which are used to build a hyper plane that separates classes with a maximum margin. The intuition of maximal margin is model having fewer dichotomies and smaller VC dimensions. In addition maximum margin constrain allows optimization methods works much efficiently since margin should be on some of the data points. (Abu-Mostafa, 2012)

The hyper plane written as $w \cdot x - b = 0$ where w is the normal of hyper plane, and $\frac{b}{\|w\|}$.is offset of hyperplane. The margin is between $w \cdot x - b = 1$ and $w \cdot x - b = -1$. Consequently, class of a point x_i can be found using sign($w \cdot x_i - b$). Then, since the margin is $\frac{2}{||\mathrm{w}||}$, maximum-margin hyper plane can be found by minimizing ||w|| subject to $y_i(w \cdot x_i - b) \geq 1$, which is an optimization problem. (Alpaydın, 2010)

The optimization issue exhibited is hard to solve because it relies on upon ||w||, therefore optimization should deal with square roots. However, it is possible to adjust the comparison by substituting ||w|| with $\frac{||\mathrm{w}||^2}{2}$ without changing the result. The substitution converts original optimization problem to $\arg\min_{(w,b)} \frac{||\mathrm{w}||^2}{2}$ subject to $y_i(w \cdot x_i - b) \geq 1$ for all x_i in x, a quadratic programming optimization problem. This optimization problem can be solvable with Lagrange method;

$$L(w, b, \alpha) = \frac{||\mathrm{w}||^2}{2} - \sum \alpha_i [y_i(w \cdot x_i - b) - 1]$$

Equation 13 Lagrange form of Support Vector Machine margin optimization

where α_i is Lagrange multiplier associated with x_i, subject to all $x_i \in x$. The solution can be found by minimizing $L(w, b, \alpha)$ with respect to w and b while maximizing α with respect each $\alpha_i \geq 0$.

In this form, the problem can be solved by quadratic solvers using combination of training data points, due to Karush-Kuhn-Tucker stationarity condition. (Abu-Mostafa, 2012)

$$w = \sum_{i=1}^{n} \alpha_i y_i \, x_i$$

Equation 14 SVM margin expressed as combination of training data points.

After substituting w with $\sum_{i=1}^{n} \alpha_i y_i \, x_i$, we can maximize in α_i to find solution to Equation 15 with subject to $\alpha_i \geq 1$ and $\sum_{i=1}^{n} \alpha_i y_i = 0$

$$\hat{L}(\alpha) = \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j {x_i}^T x_j$$

Equation 15 Lagrange form of SVM margin optimization after substitution

4.3.3. Recursive Auto-Encoder

Recursive Auto-Encoder is one of the Deep Learning algorithms that utilize Auto-encoders recursively to learn structural representations in data. Deep Learning aims to build models that are representationally efficient. Each layer uses layer beneath learn non-local generalization about the data.

Auto-Encoders are one of the algorithms used in Deep Learning. Auto-Encoders are artificial neural networks that have an input layer, hidden layers, and an output layer that has same size as input layer. Auto-Encoder aims to reconstruct input using different number of nodes. Common practice is using fewer nodes in hidden layers to force learning relation between inputs to have a more compact representation.

Recursive Auto-Encoder utilizes Auto-Encoders to combine two embedding vectors to represent them in a single vector. Therefore, sequences can be represented by a single embedding by hieratically combining embedding vectors. This allows classifiers to work on sequential data such as natural language since arbitrary length hieratically structured data can be represented as a compact fixed length vector.

The major weaknesses of previous methods were the use of Bag-of-Words. Bag-of-Words features unfortunately cannot distinguish word ordering and how words affect each other in context, i.e. "I like A more than B" and "I like B more than A" would have same representations in Bag-of-Words approach.

Recursive Auto-encoders overcome this weakness by constructing a tree that has tokens for leafs and auto-encoders for intermediate nodes. Unsupervised Recursive Auto-Encoder would try to construct a dependency tree using auto-encoder reconstruction error to minimize total reconstruction by only giving edges between dependent nodes. After the whole tree constructed, the auto-encoder's representation on the root node is fed to a softmax regression as input.

According to Socher et al(2011), each token is represented with a vector of real numbers $x_i = Lb_k \in \mathbb{R}^n$ where L is vocabulary embedding matrix; b_k is one hot representation of token where only *k'th* index is 1. The embedding matrix can either be learned from training or it can be given. Given the sequence of words x = (x_1 ….,x_n), the tree is build by pairing p -> c_1c_2 using auto-encoder, where c can be a terminal embedding of a word or a conterminal node in the tree. Parent vectors are computed from their children as follows;

$$p = f(W^{(1)}[c_1; c_2] + b^{(1)})$$

Equation 16 Parent vector calculation with Auto-Encoder

where $W^{(1)} \in \mathbb{R}^{n\times 2n}$ is input to hidden layer transformation weights of auto-encoder, $b^{(1)}$ is bias term, and f is activation function such as tanh in our work. One can use any other non-linear activation function such as sigmoid function. Then, the auto-encoder can reconstruct inputs as follows;

$$[c`_1; c`_2] = W^{(2)}p + b^{(2)}$$

Equation 17 Auto-Encoder input reconstruction where $W^{(2)} \in \mathbb{R}^{n\times 2n}$ is hidden to output layer transformation weights of auto-encoder, $b^{(2)}$ is bias term of reconstruction. After that, the auto-encoders error can be calculated using distance between input and output as follows;

$$E_{rec}([c_1; c_2]) = \frac{1}{2}\|[c_1; c_2] - [c`_1; c`_2]\|^2$$

Equation 18 Auto-encoder error

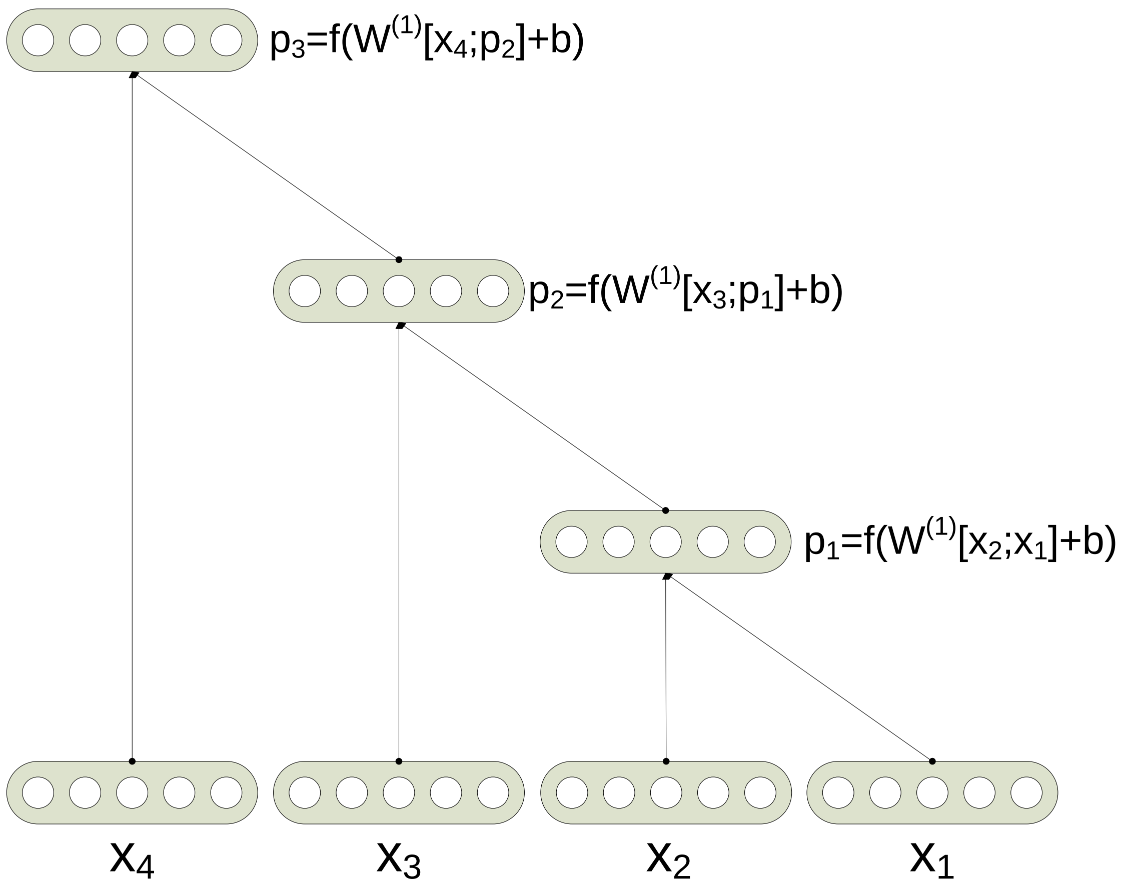
The autoencoder model shown illustrates n-dimensional vector representation of parent node that has 2 children nodes with n-dimensional vector representation (c_1, c_2). The autoencoder is recursively applied until there is single n-dimensional vector that represents whole tree.

Recursive Autoencoder can either utilize given parsing tree or generate a parsing tree without supervision if the parsing tree is unavailable. Unsupervised tree prediction would try to pair consecutive nodes that have minimum reconstruction error when paired together. Let A(x) be set of all possible parsing trees can be build on sequence and T(y) be triplets of all the non-terminal nodes in the tree that is indexed by s, Recursive Autoencoder can be computed as follows; (Socher, Pennington, Huang, Ng, & Manning, 2011)

$$RAE_\theta(x) = \underset{y \epsilon A(x)}{\operatorname{argmin}} \sum_{s \in T(y)} E_{rec}([c_1; c_2]_s)$$

Equation 19 Recursive Auto-Encoder optimization target

A simple example for word sequence x= [x_1, x_2, x_3, x_4] can be seen in Figure 7 Recursive Auto-Encoderbelow;



$p_3=f(W^{(1)}[x_4;p_2]+b)$
$p_2=f(W^{(1)}[x_3;p_1]+b)$
$p_1=f(W^{(1)}[x_2;x_1]+b)$
x_4
x_3
x_2
x_1

Figure 7 Recursive Auto-Encoder

In the example, Recursive Auto-Encoder combines x_1 and x_2 to $p_{1,}$ which is parent node that represents both x_2 and x_1 in respective order. Then, it combines p_1 and x_3 to p_2 which is another parent node that represents x_3 and p_1 in respective order. Also, since p_1 is previously constructed as representation of x_1 and $x_{2,}$ p_2 represents x_3, x_2, and x_1. In addition, the hieratical structure is also learned; if x_3 and x_2 had combined first, then resulting node had combined with x_1, resulting vector should be different than p_2. Finally, Recursive Auto-Encoder combines x_4 with p_2, and resulting node represents whole sequence.

The optimization target of Recursive Autoencoder structure prediction includes an argmin term, which makes it hard to optimize. When it is infeasible to find tree structure that minimizes reconstruction error, which is often the case, one can construct the tree greedily. Greedy unsupervised Recursive Autoencoder can apply the autoencoder on nodes iteratively selecting which minimizes error on the step until a connected tree is constructed. The algorithm makes several passes, in each step reconstruction error is calculated for all non terminal nodes available, the pair with least reconstruction error is selected and combined to a parent node. Then, resulting node is again added to available nodes and it is repeated until the tree is constructed. A good example can be found at (Socher, Pennington, Huang, Ng, & Manning, 2011), let's assume there is a sequence (x_1,x_2,x_3,x_4), it can be processed as follows; first, the reconstruction error is calculated for all consecutive pairs of two, [x_1,x_2] , [x_2,x_3], [x_3,x_4], then the pair with least reconstruction error, let's say [x_1,x_2], is combined with autoencoder. Then, resulting vector, $p_{1,2}$ is added back, so new sequence is ($p_{1,2}$, x_3,x_4). Again, repeating the same process, we select the pair with least reconstruction error from all consecutive pairs of two, [$p_{1,2}$,x_3], [x_3,x_4]. Now, there are two possible trees, $p_{((1,2),3),4)}$, $p_{((1,2),(3,4))}$. Let's assume [$p_{1,2}$,x_3] is the pair with least reconstruction error. Combining [x_3,x_4] gives us following, $p_{(1,2)}$, $p_{(3,4)}$. Since there is there is only one possible combination available, $p_{((1,2),(3,4))}$; Finally, we can combine [$p_{(1,2)}$, $p_{(3,4)}$] as $p_{((1,2),(3,4))}$.

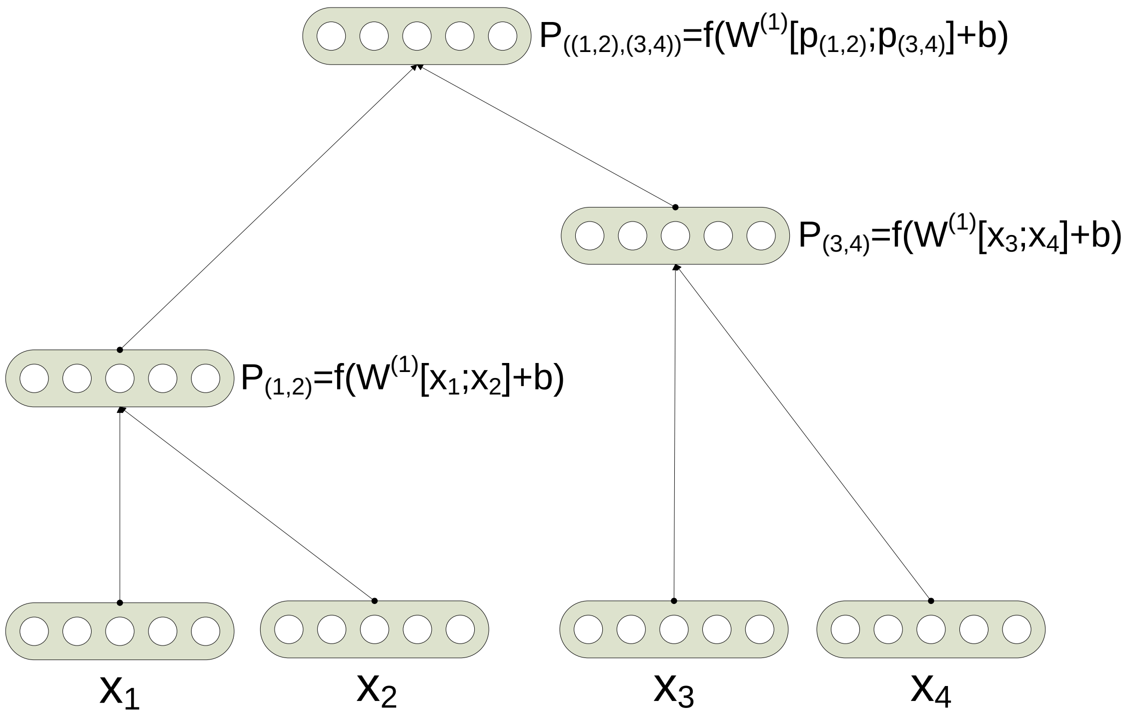

$P_{((1,2),(3,4))}=f(W^{(1)}[p_{(1,2)};p_{(3,4)}]+b)$
$P_{(3,4)}=f(W^{(1)}[x_3;x_4]+b)$
$P_{(1,2)}=f(W^{(1)}[x_1;x_2]+b)$
x_1
x_2
x_3
x_4

Figure 8 Greedy Unsupervised Recursive Autoencoder for structure prediction

4.3.3.1. Semi-Supervised Recursive Autoencoder

Recursive Autoencoder can be trained in semi-supervised way to predict class distributions. Since, the root node in Recursive Autoencoder represents the whole phrase/sentence as an n-dimensional vector, it is possible the feed this vector representation to a classifier/regression to predict class distributions.

Socher et al.(2011) shows a method to use Semi-supervised Auto Encoders to predict class distributions. The parent vector which represents the phrase, p, is utilized by adding a simple softmax later to predict class distributions.

$$d(p;\theta) = softmax(W^{label}p)$$

Equation 20 Class distribution prediction using Semi-supervised Recursive Autoencoder

The prediction d is an m-dimensional vector where there are K labels and its' elements are prediction probabilities of instance being in corresponding class. So, $d_k = p(k|[c_1,c_2])$ represents probability of a distribution k given phrase $[c_1,c_2]$. Then, cross-entropy error, where t_k is *k'th* element of target label distribution, can be calculated as follows;

$$E_{cE}(p,t;\theta) = -\sum_{k=1}^{K} t_k \, log d_k(p;\theta)$$

Equation 21 Cross-entropy error of Semi-supervised Recursive Autoencoder

Finally, the objective function to minimize for Semi-supervised Recursive Autoencoder over phrase x and label t pair can be calculated as follows;

$$J = \frac{1}{N}\sum_{(x,t)} E(p,t;\theta) + \frac{\lambda}{2}||\theta||^2$$

Equation 22 Objective function for Semi-supervised Recursive Autoencoder

where $T(RAE_\theta(x))$ is set of nodes that constructed by greedy Recursive Autoencoder, the error can be calculated as follows;

$$E(p,t;\theta) = \sum_{s \epsilon T(RAE_\theta(x))} E([c_1;c_2]_s, p_s, t, \theta)$$

Equation 23 Error of greedy Recursive Autoencoder in Semi-supervised method

For each non-terminal node, the error can be calculated as weighted average of reconstruction and cross-entropy errors.

$$E([c_1;c_2]_s, p_s, t, \theta) = \alpha E_{rec}([c_1;c_2]_s;\theta) + (1-\alpha)E_{cE}([c_1;c_2]_s, t;\theta)$$

Equation 24 Error at each node in Semi-supervised Recursive Autoencoder

where α is the parameter that weight of reconstruction and cross-entropy errors. The α parameter allows us to weight synaptic and sentimental information. (Socher, Pennington, Huang, Ng, & Manning, 2011)

CHAPTER 5

System Design

The system used is designed as multiple communicating micro-services instead of single monolithic software. The choice of multitude-ness is made for better fault tolerance, scalability, and ease of development. Each micro-service was modeled as an actor that sends and receives messages between other actors. In response to each message; an actor can make its own decisions. Instead of short lived light-weight actors, we have implemented long living heavy-weight actors that make their own computational choices; all services except Social Network Analysis services implemented as Communicating Sequential Processes for parallelism and concurrency, whereas SNA services implement map/reduce for parallelism.

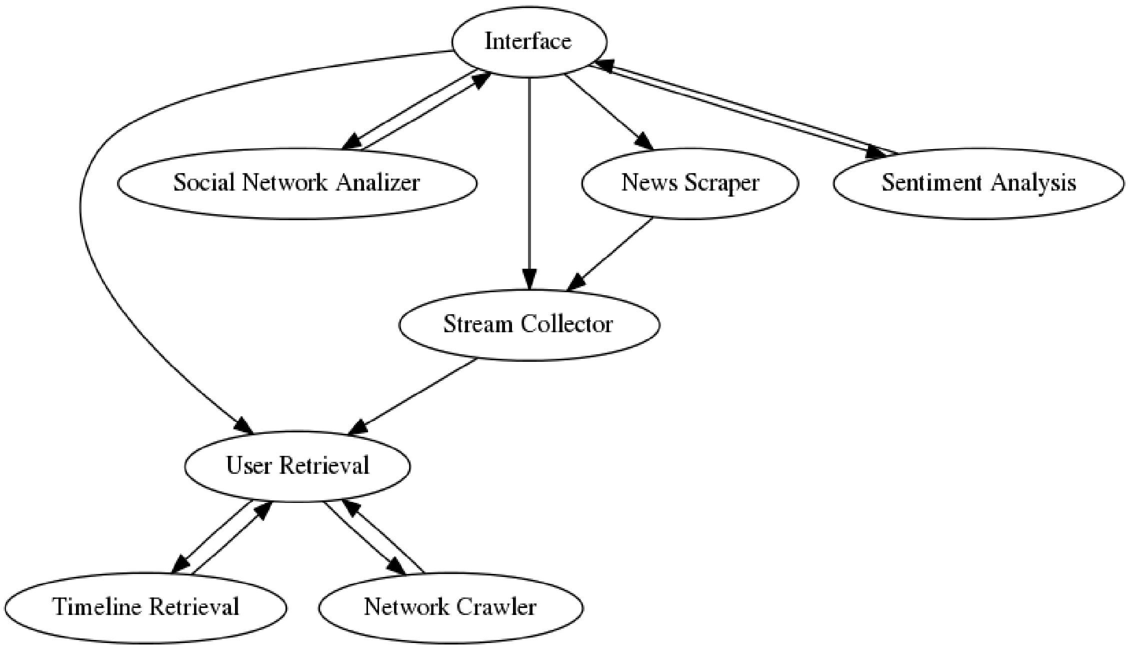


Figure 9 Inter-process Messaging Diagram

5.1. Interface

Sentiment Analysis tasks such as status message labeling, dataset and model manipulation were provided by web interfaces. We developed two different interfaces; one for optimized for faster status message labeling, and another for both labeling and modeling tasks. More complicated tasks are provided by an interactive REPL shell.

5.2. News Scraper

Web Scraping is a method of automatic information extraction from The Internet. Web Scrapers often crawl through web pages' links to retrieve the pages, then parses HTML to extract information from relevant pages. We have used web scraping to generate set of keywords of issues that we are interested.

We have crawled major online news papers to fetch articles and their categories. We have generated our keyword sets by sorting co-occurring words with seed sets and ranking with respect to their scores after removing stop-words. For keyword extraction, RAKE algorithm (Rose, 2010), is used with articles that contain words that are on our seed set.

$$Score_{RAKE} = \frac{\mathrm{deg}\,(\mathrm{w})}{\mathrm{freq}(\mathrm{w})}$$

Equation 25 Keyword scores of Rake algorithm.

5.3. Twitter Network Crawler

Our Twitter Network Crawler manages many Twitter app users' accounts, to retrieve account information, followers, and friends of given users. This service allows us to snowball a small sample set of users, and produce a large social network of people whom we filtered with previously mentioned criteria. The Twitter network crawling service listens to the processing queue for crawling jobs; each crawling job stores data about whom to be crawled and how many steps of separation is going to be followed. After execution of each job, the resulting social network persisted on the database with timestamps, and then the processing queue is informed after the job is done.

5.4. Timeline Retrieval

The Timeline Retrieval service collects status messages from the given users' timelines. Due to rate limiting and other potential problems, the service manages multiple app accounts. The processing queue is listened to retrieval jobs, that stores which timeline is going to be retrieved, and if the timeline is partially retrieved, and which parts are to be retrieved. After each job executed, the resulting status messages are persisted on the database with timestamps, then the processing queue is informed after the job is done.

5.5. Twitter User Retrieval

The Twitter User Retrieval service is similar to the other data retrieval services. It manages multiple accounts to collect user profile information from Twitter API, and stores it on database.

5.6. Twitter Stream Collector

Besides collecting status messages from the users' timelines, we also collected status messages using Twitter Streaming API. Streaming API allows its users to filter keywords, but the data is rate limited and rate limits are kept as a business secret.

We have also implemented software that fetches Trending Topics in Turkey every 15 minutes and combines them with our interested keywords set. All tweets and those tweets' user information are persisted on our database.

5.7. Sentiment Analysis

Sentiment Analysis module processes datasets consisting of obtained status messages and class labels associated to classify status messages. The module uses several vectorizers such as; TF, TF/IDF, Embedding Matrix, and machine learning algorithms including; Multinomial Naïve Bayes, Linear Support Vector Machines and Recursive Auto-Encoders. After a model is constructed by using a dataset and one of the Machine Learning algorithms mentioned, the model is serialized and persisted on the database, so that the constructed model can be used to classify other datasets later on.

5.8. Social Network Analysis

Centrality measures are used for quantifying complex interactions and communication between nodes in the network. For quantified analysis of structural importance in the social network, we have used network centrality measures including; Indegree, Betweenness, and eigenvector centrality. The Social Network Analysis module calculates these measures on the graph we have retrieved and stores results on the database.

CHAPTER 6

Implementation

The system implemented as multiple services that dispatches jobs to distributed task queue, and the workers, processes those tasks or dispatches tasks of their own. Implementation is done mainly on Python programming language but also Go and Java programming languages are used. In addition many open source libraries/frameworks used during the development; Tweepy Twitter Library, Scikit-learn Machine Learning Library, Django web framework, Celery task queue, Scrapy framework, lxml and matlotlib plotting library are used in Python code. One of the labeling interfaces is written in Go programming language with Revel Web Framework. Also, some of the data retrieval services written with help of kukrik/OAuth2 library in Go programming language. The centrality measures are implemented in C++ programming language since they are memory allocation and array access heavy computation.

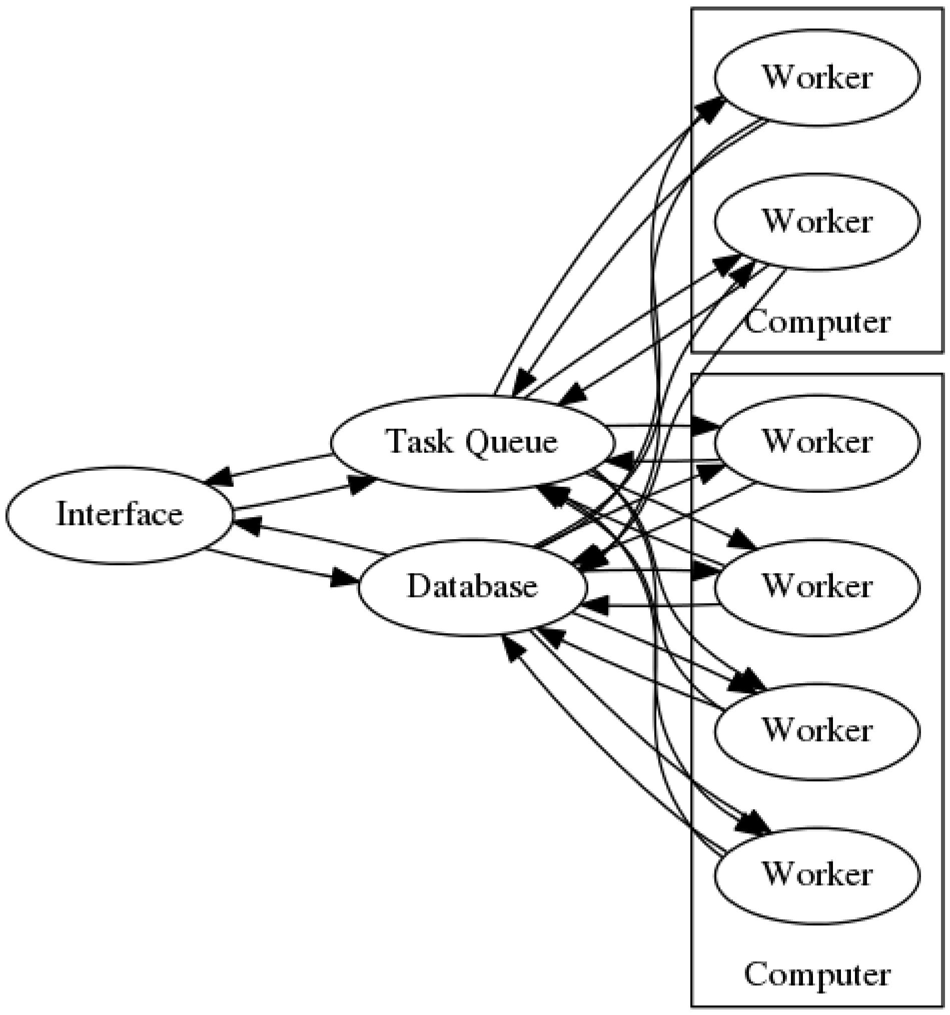


Figure 10 Messaging Scheme on Task Distribution Architecture (Simplified).

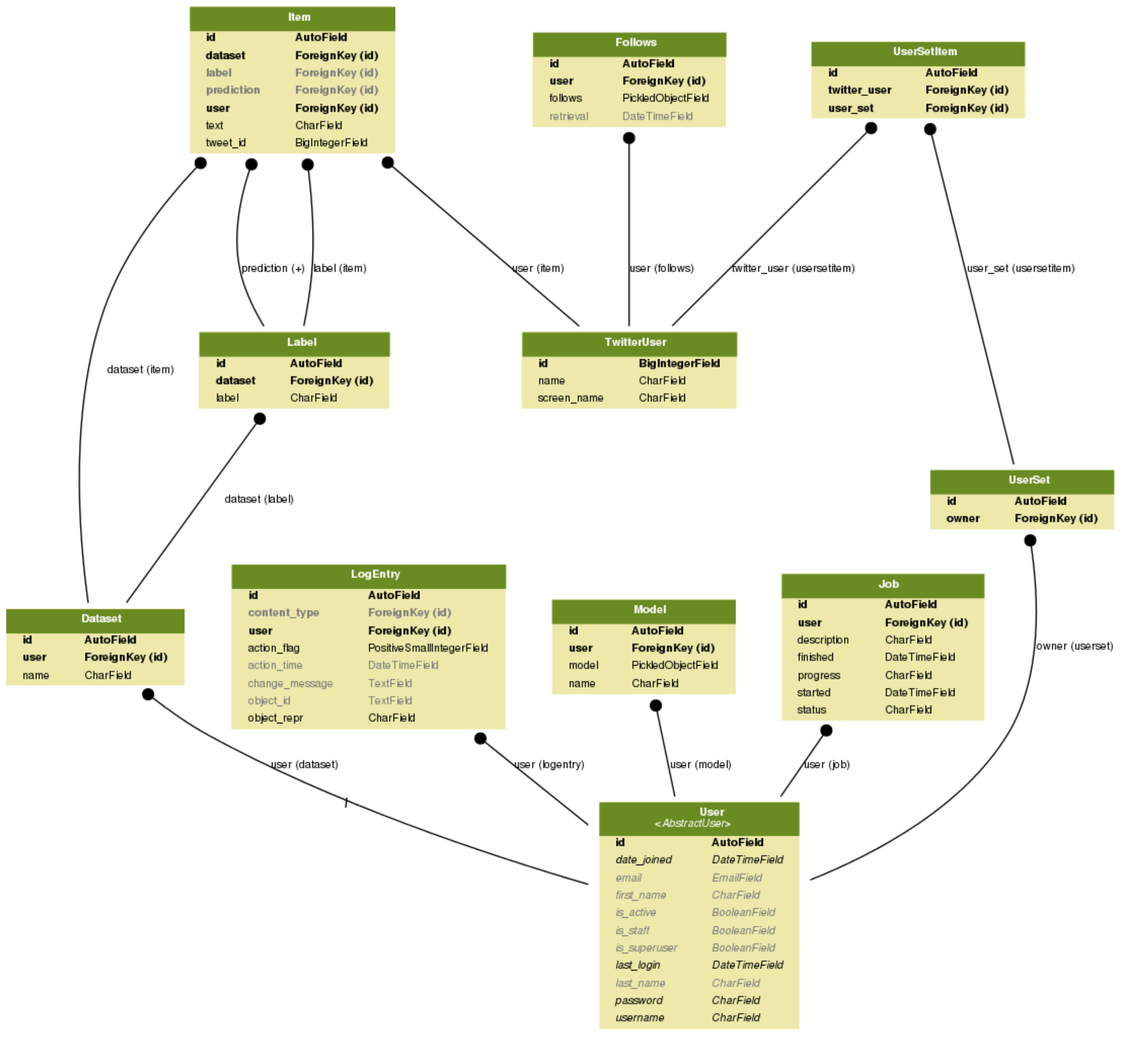
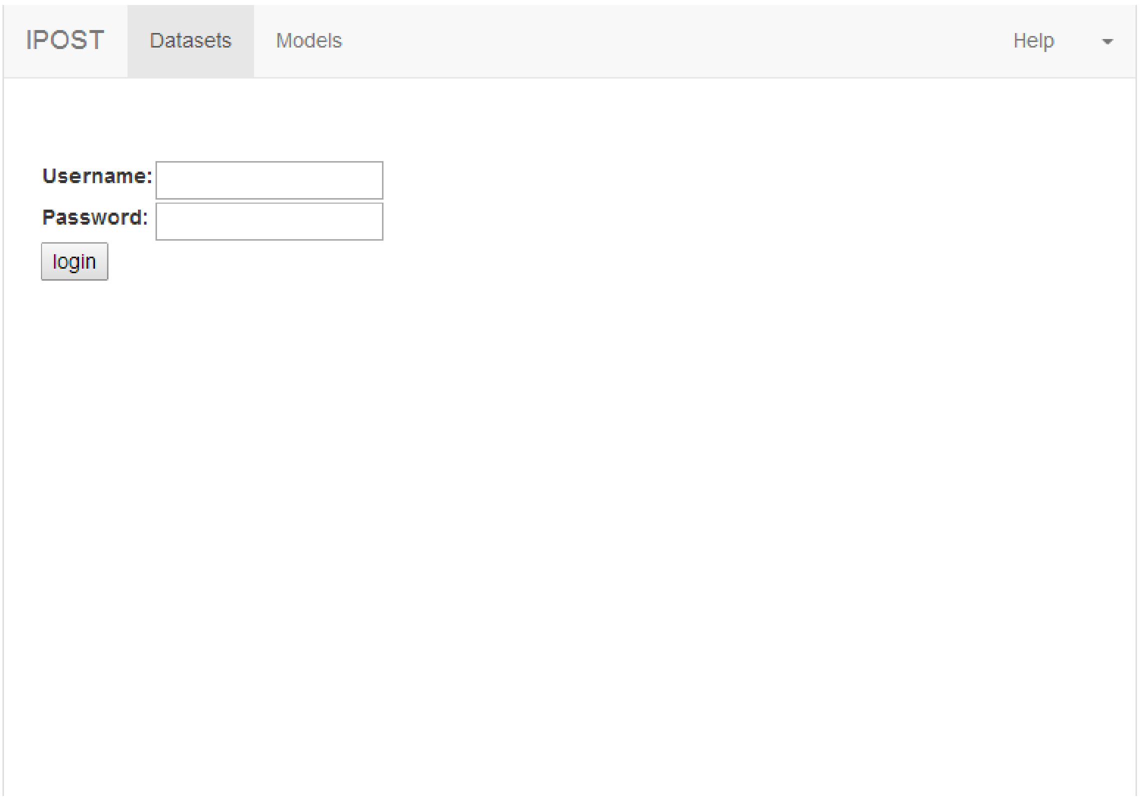


Figure 11 Entity Relation Diagram.

6.1. Web Interface

The web interface is implemented using Django web framework, inter-process message passing is done with RabbitMQ, and data persisted on PostgreSQL database. The web interface allows users to login with their passwords, manipulate tweet datasets, label tweets, train machine learning models, and predict any given tweets' sentiment. All long running tasks, such as training and prediction are done asynchronously by worker processes on several servers. In the front end side, JQuery and Twitter Bootstrap are used.

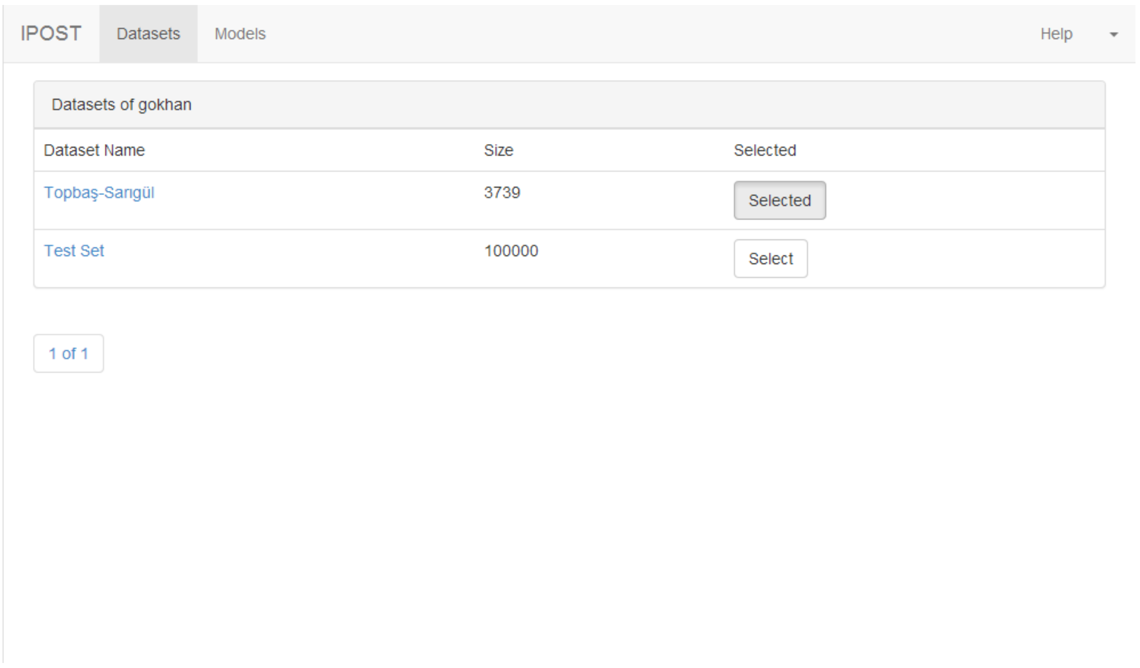


Username:

Password:

login

Figure 12 Login page

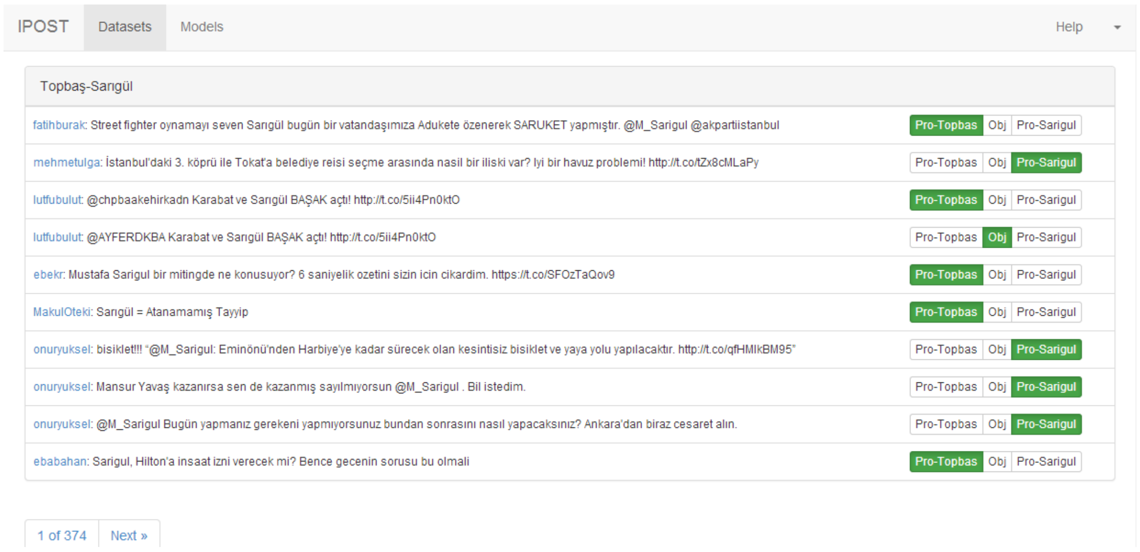


Datasets of gokhan

Dataset Name	Size	Selected
Topbaş-Sarıgül	3739	Selected
Test Set	100000	Select

1 of 1

Figure 13 Dataset page

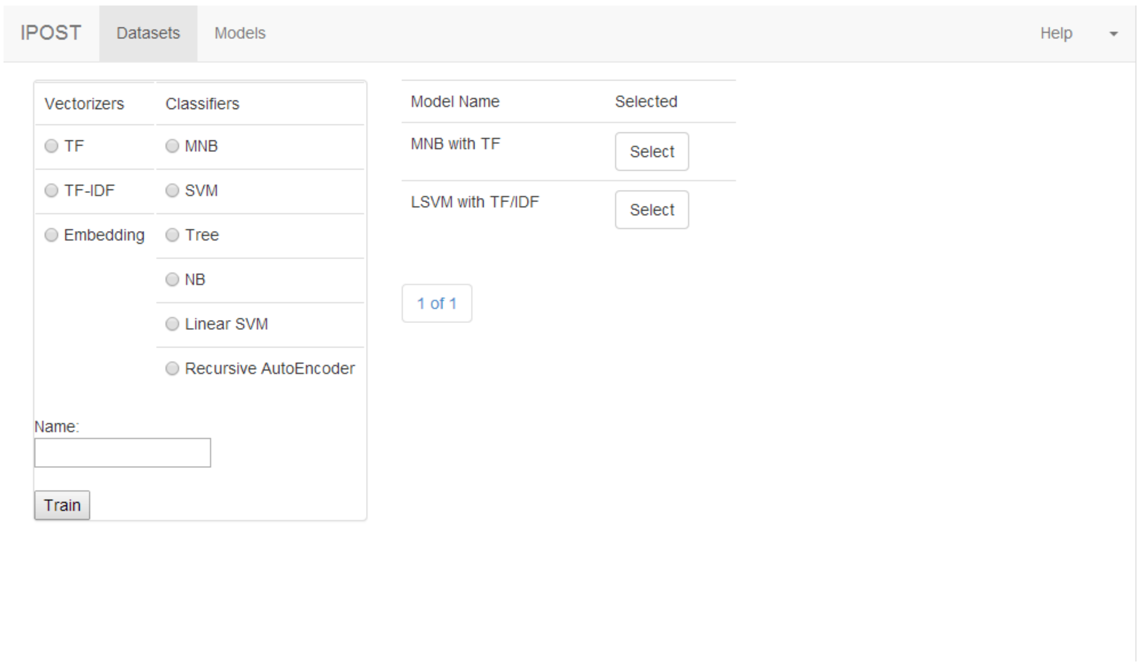

IPOST | Datasets | Models | Help

Topbaş-Sarıgül

fatihburak: Street fighter oynamayı seven Sarıgül bugün bir vatandaşımıza Adukete özenerek SARUKET yapmıştır. @M_Sarigul @akpartiistanbul	[✓ Pro-Topbas] Obj Pro-Sarigul
mehmetulga: İstanbul'daki 3. köprü ile Tokat'a belediye reisi seçme arasında nasil bir iliski var? Iyi bir havuz problemi! http://t.co/tZx8cMLaPy	Pro-Topbas Obj [✓ Pro-Sarigul]
lutfubulut: @chpbaakehirkadn Karabat ve Sarıgül BAŞAK açtı! http://t.co/5ii4Pn0ktO	[✓ Pro-Topbas] Obj Pro-Sarigul
lutfubulut: @AYFERDKBA Karabat ve Sarıgül BAŞAK açtı! http://t.co/5ii4Pn0ktO	Pro-Topbas [✓ Obj] Pro-Sarigul
ebekr: Mustafa Sarigul bir mitingde ne konusuyor? 6 saniyelik ozetini sizin icin cikardim. https://t.co/SFOzTaQov9	[✓ Pro-Topbas] Obj Pro-Sarigul
MakulOteki: Sarıgül = Atanamamış Tayyip	[✓ Pro-Topbas] Obj Pro-Sarigul
onuryuksel: bisiklet!!! "@M_Sarigul: Eminönü'nden Harbiye'ye kadar sürecek olan kesintisiz bisiklet ve yaya yolu yapılacaktır. http://t.co/qfHMIkBM95"	Pro-Topbas Obj [✓ Pro-Sarigul]
onuryuksel: Mansur Yavaş kazanırsa sen de kazanmış sayılmıyorsun @M_Sarigul . Bil istedim.	Pro-Topbas Obj [✓ Pro-Sarigul]
onuryuksel: @M_Sarigul Bugün yapmanız gerekeni yapmıyorsunuz bundan sonrasını nasıl yapacaksınız? Ankara'dan biraz cesaret alın.	Pro-Topbas Obj [✓ Pro-Sarigul]
ebabahan: Sarigul, Hilton'a insaat izni verecek mi? Bence gecenin sorusu bu olmali	[✓ Pro-Topbas] Obj Pro-Sarigul

1 of 374 | Next »

Figure 14 Label page


IPOST | Datasets | Models | Help

Vectorizers	Classifiers
TF	MNB
TF-IDF	SVM
Embedding	Tree
	NB
	Linear SVM
	Recursive AutoEncoder

Name:

Train

Model Name	Selected
MNB with TF	Select
LSVM with TF/IDF	Select

1 of 1

Figure 15Models page

6.2. News Scrapper

The news crawler was implemented using Scrapy framework and lxml. News scrapper periodically, visited news papers, and fetched articles within a given depth from seed URLs. Content from each page visited is extracted with XPATH queries. Then, the articles including our seed set are selected and RAKE scores are calculated. Finally, top scoring keywords are sent to Twitter Stream Collector module.

Interval between runs chosen as 1 hour and crawling depth has been 3 links from the main page of the site. Since maximum of 400 words can be tracked by Twitter Streaming API as implementation is done, only top scoring 400 keywords are send to Stream Collector.

6.3. Twitter Information Retrieval

Twitter user, timeline, network retrieval services are very similar. Our Twitter information retrieval services assign jobs to many Twitter app users, while keeping track of rate limits of each app user. These services are implemented in Go programming language, with OAuth2 library. The data is persisted on PostgreSQL database with the exception of status messages. Status messages are persisted on a small Apache Cassandra Cluster with GoCQL driver, which gave us better performance and space efficiency on database write operations, since only small portion of the status messages are used in the sentiment analysis and needs to be in a relational database.

6.4. Sentiment Analysis

Sentiment Analysis service was implemented mostly by utilizing Scikit-learn. Multinomial Naïve Bayes classifier and Linear SVM used from Scikit library. On the other hand Semi-supervised Recursive Auto-Encoder is based on Sanjeev Satheesh's work and it is written in Java programming language. (Satheesh, 2014)

6.5. Social Network Analysis

Social network analysis services are the computationally heaviest part of our systems. For efficient computation, it is implemented in C++ programming language, instead of common practice use of hash-maps; we have used faster arrays to implement adjacency lists. Also, Eigenvector centrality and Betweenness centrality implementations are parallelized to utilize multiple processors and computers.

The Eigenvector Centrality is parallelized by splitting computation of sparse dot product operation in power iteration algorithm to multiple processors. The result array sliced into smaller chunks, and each chunk is calculated in a different processor.

On the other hand, the Betweenness Centrality algorithm was calculating shortest paths starting from a sample set of nodes. The implementation was parallelized by slicing sample node set into smaller sets, calculating Betweenness Centrality for sub-samples, and then normalizing results. Another alternative distributed memory method from Edmonds et al (Edmonds, Hoefler, & Lumsdaine, 2009) is also experimented.

Social Network Analysis jobs are scheduled from command line interface, scheduled by job processing queue then calculated from the worker daemons in our servers. Then, results are persisted on the database.

6.6. Software Stack

Due to high traffic from Twitter, large size of the dataset and unusual access patterns of our project, infrastructure was a very important issue. We have run several benchmarks to assess current software options. Overall results showed us that it is best to store transactional data in PostgreSQL due to its speed and overall features, and non-transactional, update heavy data in Cassandra. The production system persistent data on two databases, large dataset of timeline status messages are stored on a Cassandra cluster, but the rest of the data is persisted on PostgreSQL database.

Messaging between services is carried by RabbitMQ message broker.

CHAPTER 7

Results

7.1. Social Network Analysis Results

The results given below are calculated on Turkish Twitter social network, which was retrieved in March 2013. Edges associated with non-Turkish users, spam bots, and inactive accounts are omitted.

7.1.1. Indegree Centrality

Name	Indegree Rank	Eigenvector Rank	Betweenness Rank
Cem Yılmaz	1	29	195
Abdullah Gül	2	101	1592473
atademirer	3	50	42
Demet Akalin Kurt	4	72	3
Recep Tayyip Erdoğan	5	195	6216422
NTV Spor	6	232	481
Gülben	7	112	4
Galatasaray SK	8	172	377
Sertab Erener	9	121	48
okan bayulgen	10	37	23
Murat Boz	11	109	123
ayse ozyilmazel	12	126	101
Fenerbahçe SK	13	262	202
NTV	14	145	380
Nil Karaibrahimgil	15	108	600
Kenan Doğulu	16	147	52

Yalın	17	179	485
Ece Erken	18	230	39
Yılmaz Erdoğan	19	149	1065
Hande Yener	20	156	10
hilal cebeci	21	465	56
Ozan Doğulu	22	247	382
Kemal Kılıçdaroğlu	23	341	55
Alex10	24	337	318
cüneyt özdemir	25	226	280

Table 4 Most influential users according to Indegree Centrality

7.1.2. Betweenness Centrality

Name	Indegree Rank	Eigenvector Rank	Betweenness Rank
Bak ne demiş ?	129	1	5
Lady	171	2	15
Erostroloji	175	3	16
gaf ebesi	64	4	1
Yetkin Acar	255	5	86
Kiss'li Sözler	257	6	107
Düşersem tutun beni	314	7	91
Ferudun ÖZDEMİR	325	8	83
Edebiyat Kulübü	322	9	152
RenkliTweetler	224	10	47
Kelebek Etkisi	443	11	142
❤ LoVely ❤	434	12	162
Cümle Doktoru	424	13	156
Öz'lü Söz'lü	342	14	130
Tweet Günlüğü	352	15	80
Keskin Sözler	285	16	59
Bay Enteresan	296	17	36
Sevdim Bunu	332	18	82
Twitine Geldim★	435	19	133
Sence	345	20	102
edebiyat felsefe	183	21	41
Çekici Tweet	333	22	128
mehmet hacıbeyoğlu	262	23	32
Özel Cümleler	476	24	176
Bay Empati	334	25	62

Table 5 Most influential users according to Betweenness Centrality

7.1.3. Eigenvalue Centrality

Name	Indegree Rank	Eigenvector Rank	Betweenness Rank
gaf ebesi	64	4	1
HUYSUZ AYI	63	69	2
Demet Akalin Kurt	4	72	3
Gülben	7	112	4
Bak ne demiş ?	129	1	5
Her gün 1Yeni Bilgi	181	243	6
KitapCümleleri	147	93	7
Ümit Yaşar Oğuzcan	202	58	8
Burcunuz Ne Diyor?	233	373	9
Hande Yener	20	156	10
Cebimdeki Kelimeler	189	327	11
Hayat Felsefesi	89	169	12
Can Yücel	169	148	13
Film Replikleri	188	54	14
Lady	171	2	15
Erostroloji	175	3	16
Kaliteli Tweetler	245	90	17
PopulerTwitler	187	53	18
fragman.web.tr	378	212	19
ahmet hakan	53	357	20
Türkçe Olimpiyatları	474	871	21
Melih Bayram Dede	878	787	22
okan bayulgen	10	37	23
allahcc	420	909	27
battin	787	297	28

Table 6 Most influential users according to Eigenvector Centrality

7.2. Case Study: 2014 Turkish Municipality Elections

7.2.1. Sentiment Analysis Results

We have used 2014 Istanbul local elections as our test case and collected tweets for a month before the elections. We have used 3739 status messages to train our models. MNB classifier is tested with Term Frequency vectors; on the other hand, SVM classifier with cost value as 3 is trained with TF-IDF vectors with L2 normalization.

Recursive Auto-Encoder is trained with 100 BFGS iterations, alpha value as 0.5 and with cost value as 1.

Class	# of status messages
Pro-Topbaş	2396
Pro-Sarıgül	882
Other	461
Total	3739

Table 7 Class distribution in training set

Classifier	Accuracy
MLP*	74.9%
NB*	78.4%
MNB	80.1%
SVM	82.0%
SSRAE	88.3%

Table 8 Classifier accuracy on training set (5-fold CV)

	Ratio
Topbaş	72.54%
Sarıgül	27.46%

Table 9 Distribution prediction of status messages shows support on whole dataset

	Ratio
Topbaş	60.91%
Sarıgül	39.09%

Table 10 Distribution prediction of users that sent status messages on whole dataset

7.2.2. Centrality Distribution

Our analysis shows that both groups have similar rank distributions in all the centrality measures we have calculated. The correlations between distributions are high and shapes of the histograms are nearly identical.

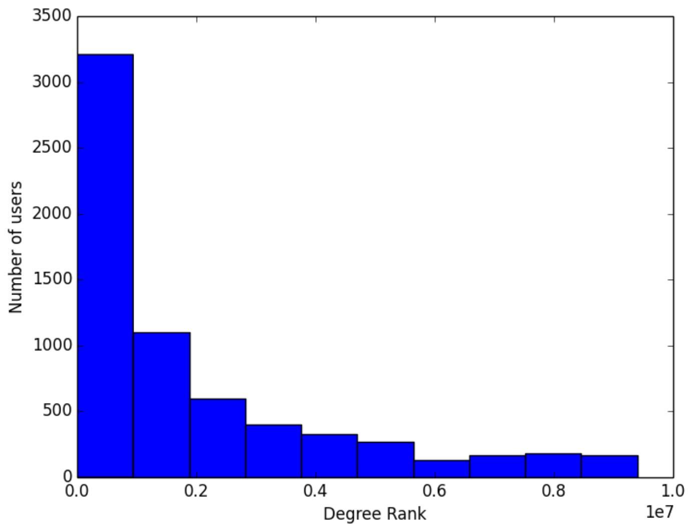


Figure 16 Pro-Topbaş users Indegree Centrality Rank Histogram

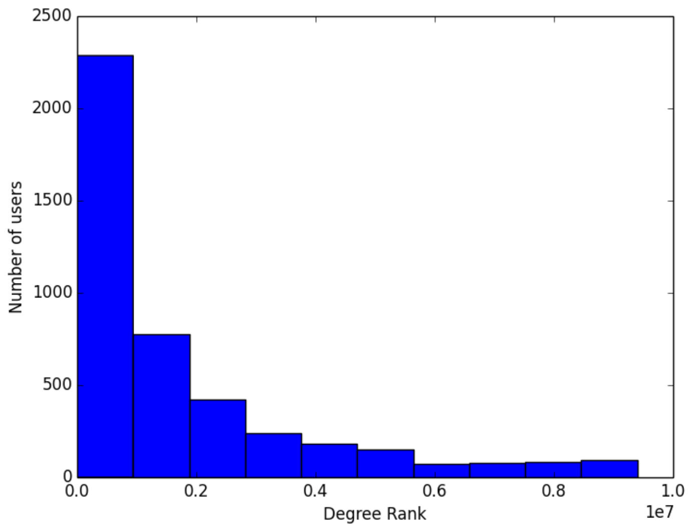


Figure 17 Pro-Sarıgül users Degree Centrality Rank Histogram

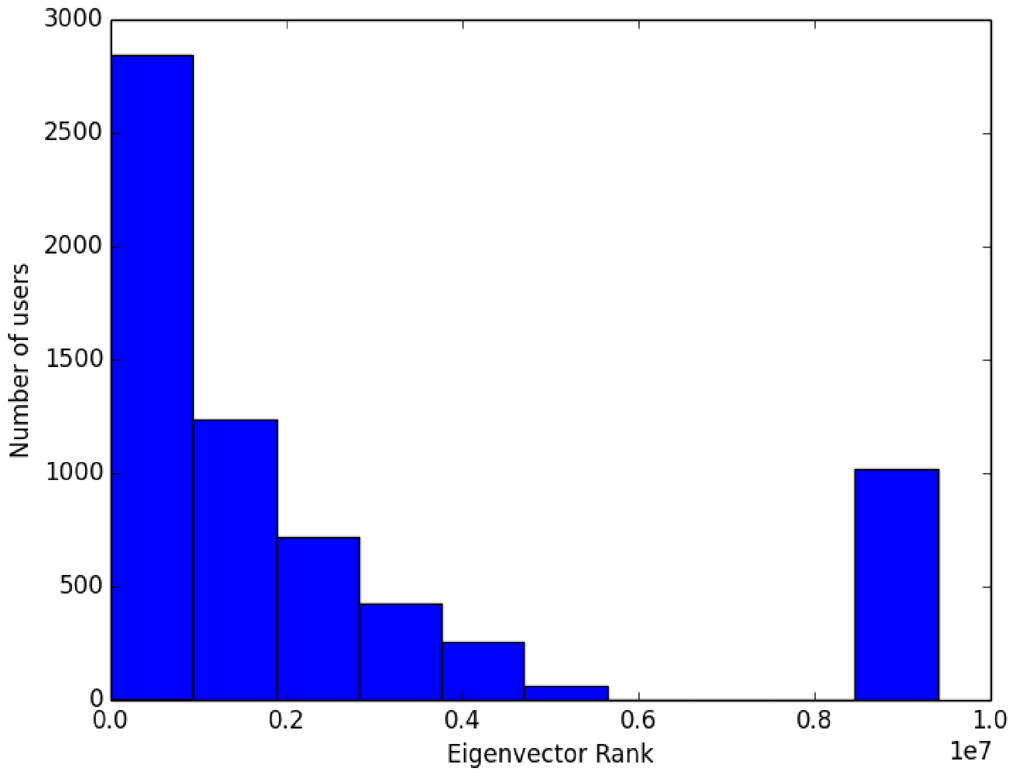


Figure 18 Pro-Topbaş users’ Eigenvector Centrality Rank Histogram

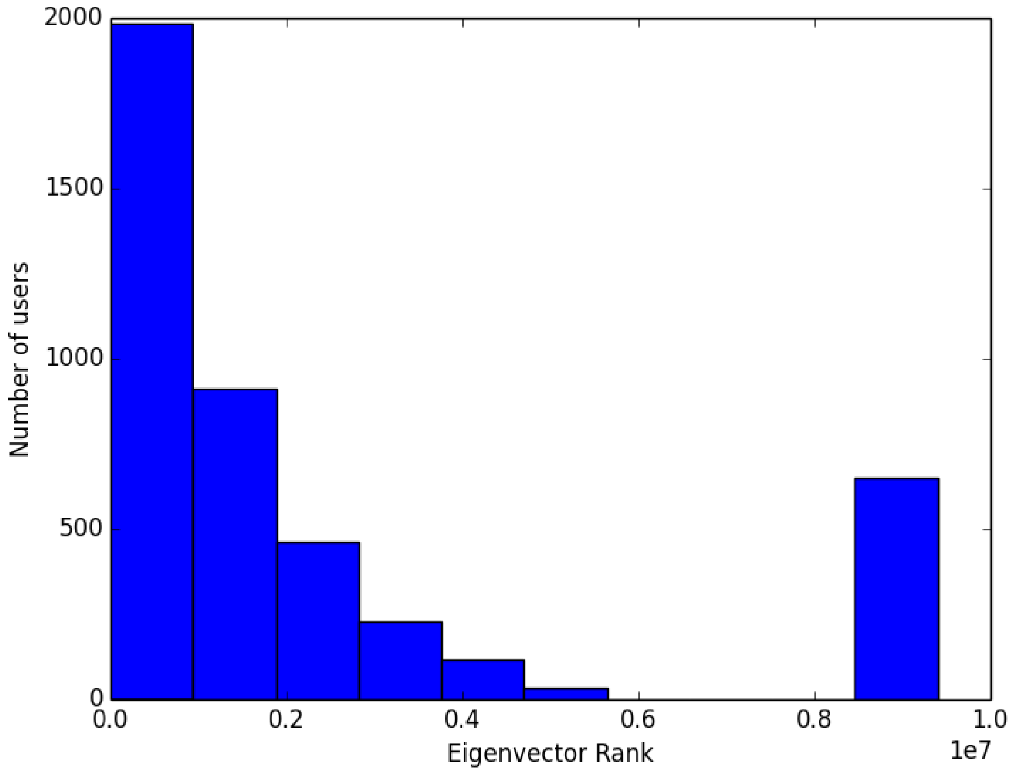


Figure 19 Pro-Sarıgül users’ Eigenvector Centrality Rank Histogram

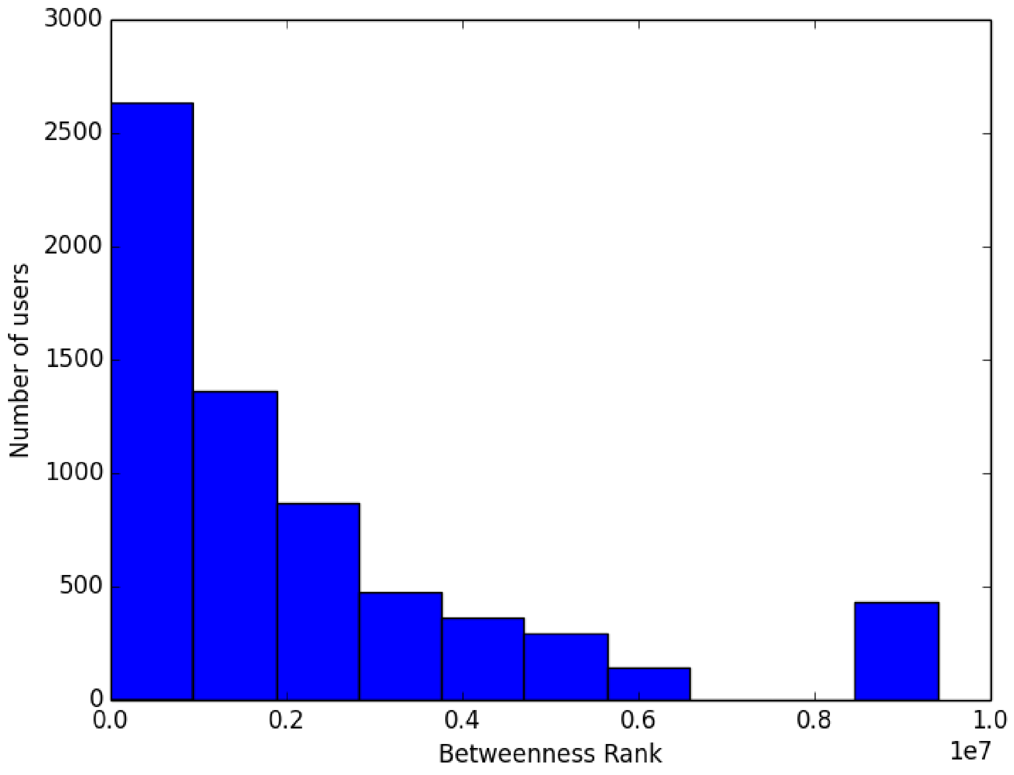


Figure 20 Pro-Topbaş users' Betweenness Centrality Rank Histogram

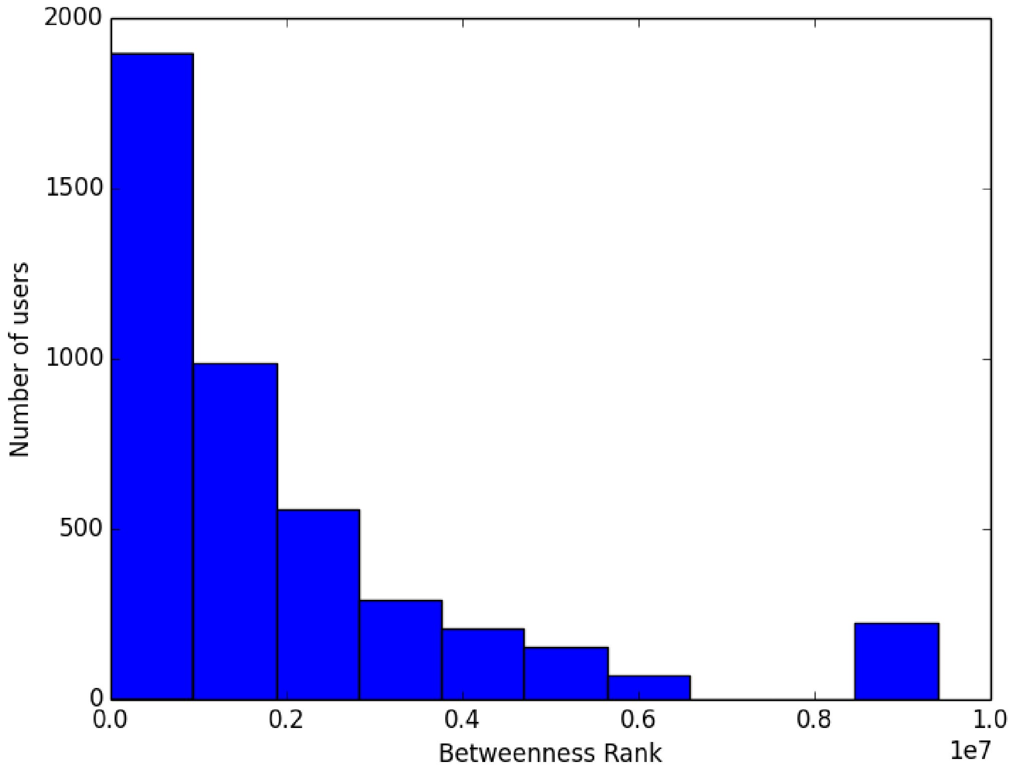


Figure 21 Pro-Sarıgül users' Betweenness Centrality Rank Histogram

The users are classified with users' centrality rank differences. Actors' roles are classified according to the structural indicators that are mention in

. If rank differences between two centrality measures are greater than some given value; rank of the lesser percentile is accounted for high centrality, and greater percentile is accounted for low centrality. After using centrality rank classification, users are assigned to their network role classes. For example, if a user's Indegree centrality rank is in 5% percentile, Eigenvector centrality is in 55%, rank difference parameter used in classification is 50 percent; the user is classified as "Marginal/Isolated Leader".

Actor Role	Pro-Topbaş	Pro-Sarıgül
Marginal/Isolated Leader	0.81%	0.57%
Conventional Ineffective	2.13%	2.14%
Ghost Opinion Shaper	2.33%	1.87%
Specialized Opinion Shaper	0.72%	0.62%
Influence Broker	8.23%	9.26%
Gatekeeper	8.12%	9.33%

Table 11 Role distribution of the case study, with rank diff=0.4

The analysis shows that Topbaş has more Marginal/Isolated leaders and Ghost Opinion Shapers; on the other hand, Sarıgül has more Infuence Brokers and Gate keepers. The Ineffective/Conventional and Specialized Opinion Shapers are distributed similarly for both parties. Even though nearly 3 times Topbaş supporting tweets are published and twice as many users are supporting him, the election results were %47.91 for Topbaş and %40.05 for Sarıgül, a lesser difference than their Twitter presence.

CHAPTER 8

Conclusion

In this thesis, we aimed to show an interdisciplinary work that provides empirical quantifiable answers for social science problems using network analysis and machine learning. We built a software system that can generate social graph of online platforms, target specific group, and analyze both their structural importance and content they are generating/disseminating. While doing so, we tried to make the system bias free and its result quantifiable. Also, the system should be able to process data with little supervision as possible.

The dataset is generated from Twitter. The graph retrieval started from a small set of users and snowballed into most of the Turkish users. The account information provided for Twitter users allows us to classify them using machine learning. After generating a social graph of Turkish Twitter users, we used centrality measures to find structural importance and roles.

After selecting an event, namely the Istanbul 2014 local elections: we focused on the content people are sharing on Twitter, classified status messages using a state of the art sentiment analysis techniques, and developed easy to use collaborative user interface for sentiment analysis.

Finally, we combined two parts, network analysis and sentiment analysis to analyze an actual political event. The results showed us the performance of our sentiment analysis system and structural differences between two parties in a political event.

REFERENCES

Abu-Mostafa, Y. (2012, May 18). *Support Vector Machines - Hard Margin.* Retrieved from Machine Learning Video Library: http://work.caltech.edu/library/140.html

Alpaydın, E. (2010). Introduction to Machine Learning. The MIT Press.

Barabási, A.-L. (2003). *Linked: how everything is connected to everything else and what it means for business, science, and everyday life.* New York, NY: Plum.

Bort, J. (2012, December 4). *How Starbucks And Other Companies Use Complex Math Algorithms To Read Your Feelings Online.* Retrieved from Business Insider: http://www.businessinsider.com/twitter-facebook-monitoring-2012-11

Brandes, U. (2001). A Faster Algorithm for Betweenness Centrality. *The Journal of Mathematical Sociology, Volume 25, Issue 2* , 163-177.

Brandes, U., & Pich, C. (2007). Centrality estimation in large networks. *International Journal of Bifurcation and Chaos , 17* (07), 2303--2318.

Edmonds, N., Hoefler, T., & Lumsdaine, A. (2009). A Space-Efficient Parallel Algorithm for Computing Betweenness Centrality in Sparse Networks. *Indiana University tech report* .

Freeman, L. C. (1979). Centrality in Social Networks: Conceptual Clarification. *Social Networks. n.1.* , pp. 215-239.

Hauskrecht, M. (2003). *Support Vector Machines.* Retrieved March 15, 2013, from Lecture notes of Machine Learning: http://people.cs.pitt.edu/~milos/courses/cs2750-Spring03/lectures/class11.pdf

Katz, E. (1957). The two-step flow of communication: An up-to-date report on an hypothesis. *Public Opinion Quarterly , 21* (1), 61--78.

Katz, E., & Lazarsfeld, P. F. (1970). *Personal Influence, The part played by people in the flow of mass communications.* New York: Transaction Publishers.

Lazarsfeld, P. F., Berelson, B., & Gaudet, H. (1948). *The People's Choice: How the Voter Makes Up His Mind in a Presidential Campaign.* Columbia University Press.

McCallum, A., & Nigam, K. (1998). A comparison of event models for Naive Bayes text classification. *AAAI-98 workshop on learning for text categorization*, (p. 752).

Moody, J. (2012). *Social Network Analysis Lecture Notes.* Retrieved March 15, 2014, from www.soc.duke.edu/~jmoody77/s884/notes/class_centrality.ppt

Newman, M. E. (2008). The mathematics of networks. *The new palgrave encyclopedia of economics* .

Newman, M., Barabási, A.-L., & Watts, D. J. (2006). *The Structure and Dynamics of Networks.* Princeton University Press.

Rennie, J. D., Shih, L., Teevan, J., & Karger, D. R. (2003). Tackling the poor assumptions of naive bayes text classifiers. *Proceedings of the Twentieth International Conference on Machine Learning.* Washington DC.

Rose, S. E. (2010). Automatic Keyword Extraction from Individual Documents. In S. E. Rose, *Text Mining: Theory and Applications* (pp. 5-11). John Wiley & Sons.

Satheesh, S. (2014, January 26). *jrae.* Retrieved from github.com/sancha/jrae

Schectman, J. (2012, December 7). *Wall Street Journal.* Retrieved from Obama's Campaign Used Salesforce.com To Gauge Feelings of Core Voters: http://blogs.wsj.com/cio/2012/12/07/obamas-campaign-used-salesforce-com-to-gauge-feelings-of-core-voters/

Schultes, R. G., Sanders, P., & Schultes, D. (2008). Better Approximation of Betweenness Centrality. *Proceedings of the 10th Workshop on Algorithm Engineering and Experimentation*, (pp. 90–100). Siam.

Socher, R., Pennington, J., Huang, E. H., Ng, A. Y., & Manning, C. D. (2011). Semi-supervised recursive autoencoders for predicting sentiment distributions. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, (pp. 151-161).

Watts, D. J., & Dodds, P. S. (2007). Influentials, Networks, and Public Opinion. *Journal of consumer research* , 441--458.

We Are Social. (2014, January 9). *Social, Digital & Mobile Worldwide in 2014*. Retrieved May 17, 2014, from We Are Social: http://wearesocial.net/blog/2014/01/social-digital-mobile-worldwide-2014/