ΑΣΟΕΕ 1920

ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS
DEPARTMENT OF INFORMATICS

November 2012

Collecting, identifying and classifying the Greek Twitter Community

Athanasios Rebelos
R.N. M4090045

Supervisor: Assistant Professor, Martha Sideri
Reviewer: Lecturer, Vangelis Markakis

MScIS
MSc in Information Systems
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

THESIS

Prologue

This paper engages the problem of identifying users' nationality in the social network Twitter. We address the problem properly, and suggest algorithms and tools to provide an accurate prediction of user nationality.

This project was carried out under the supervision and helpful guidance of assistant Professor Martha Sideri as a prerequisite for completion of the MSc in Information Systems of Athens University of Economics and Business. I would also like to thank Konstantina Galbogini for her support and insight in the subject and Ioannis Pavlopoulos for his guidance in Weka software package.

Athens, November 2012

Athanasios Rebelos

Abstract

We have created methods for collecting data from Twitter users from the Greek community. The algorithms include classifiers and graph-theoretic properties of the user graph. We have evaluated our results in our collections of tweets and found the most influential users, for non-overlapping sets of users per region.

Keywords: Location Identification in Twitter, Language Identification in Twitter, Nationality Identification in Twitter, Greeklish Classifier, Most Influential Twitter users

Table of Contents

1 Introduction 1

1.1 Scope 1

1.2 Chapters 2

2 Data Extraction from Twitter 3

2.1 Twitter – Social Network 3

2.2 Twitter's Attractive Features 4

2.3 Twitter APIs 5

2.3.1 The Streaming APIs 5

2.3.2 The REST API 7

2.4 Getting the data 8

2.5 Tor – The Anonymity Network 9

2.5.1 Using Tor for data extraction from Twitter 10

2.6 Getting and storing the data 11

2.6.1 Data retrieving module 11

2.6.2 Queue System and Data processing module 11

2.6.3 Platform's Advantages and disadvantage 12

2.6.4 Our Twitter Collection 13

2.6.5 Implementation Details 13

3 Identifying Greek Users 14

3.1 It's all Greek to me 14

3.2 "Greeklish Convertor" Classifier 15

3.2.1 The algorithm 15

3.2.2 Technology Used 16

3.2.3 Classifier Tuning 16

3.2.4 The final algorithm 18

3.2.5 "Greeklish Convertor" Classifier Results 18

3.3 Weka Classifier 19

3.3.1 Weka 19

3.3.2 Selected Algorithms *20*
3.3.3 Data Preparation *20*
3.3.4 Weka Classifier Results *21*
3.4 Classification Results 22
3.5 Live Online Classifiers 23
4 The Most Influential Users 24
4.1 Introduction 24
4.2 Algorithm Analysis 25
4.3 Results 26
5 Metric for Unidentified Users 29
5.1 The Uncategorized User Problem 29
5.2 The "Following" Metric 29
5.3 Testing the metric 30
5.3.1 Tuning *30*
5.3.2 Results *31*
5.3.3 Conclusions *32*
6 Conclusion 33
6.1 Summary and Results 33
6.2 Future work 34
7 Appendix 35
8 Glossary 39
9 References 41

1

Introduction

1.1 Scope

Identifying users' nationality is one of hottest open issues in social networks. It is easy to identify them using their personal information (location, place of birth etc.), but not all users fill in those fields and many of them even if they provide this information they do not belong in the scope of influence of that group. The language that the users use in their texts in posts can also be used for categorizing them in nationality groups, when this language is used only in one country. But also in this occasion we have to keep in mind that many languages when are used in the web might have a different alphabet. It is common for users to write in their native language using the English alphabet instead of their language's alphabet (i.e. Greek/Greeklish). Also users can be identified from their current location point that is used in their posts, a feature that was introduced by the use of social networks from smartphones. Finally, we can derive information from the users' connections to other already categorized users. It is safe to say that the group of users with which a user has more connection than others, it more probable to be the group that the user belongs.

In this paper we combine all the above characteristics of the social network Twitter to create algorithms and tools to provide an accurate prediction of user nationality.

1.2 Chapters

In the second chapter the data extraction process for Twitter Social Network is analyzed. We present all Twitter APIs and the processes needed to pull data. Also we provide with an overview of the platform that was developed for the data extraction.

In the third chapter we illustrate the classification methods that were developed for the identification of Greek Twitter users. We present two classifiers: one based on a Greeklish translator, and one using Weka machine learning software, and then their results.

In the fourth chapter, we analyze the “Most Influential Users” algorithm. We test the algorithm on a specific user set and we display the results.

In the fifth chapter, a metric for unidentified users is introduced, the “Following Metric”. After the analysis, the metric is tested on sets of identified and unidentified users to measure the effectiveness of the metric.

Finally in the sixth chapter, we conclude by summarizing our results and present open problems and future work.

2

Data Extraction from Twitter

2.1 Twitter – Social Network

Twitter is a real-time information network that connects you to the latest stories, ideas, opinions and news that the user finds interesting. The user simply selects the accounts he finds most compelling and follows their conversations. This micro−blogging service has quickly become the social media service that is most often used in strategic communication campaigns.

At the heart of Twitter are small bursts of information called Tweets. Each Tweet is 140 characters long. The user can share and see photos, videos and conversations directly in Tweets to get the whole story at a glance, and all in one place. Twitter allows users from across the globe to share information through private and public. The site−imposed character limit allows users' updates, or tweets, to be sent to cellular phones and other mobile devises as a text message.

The idea behind Twitter is that you broadcast to anyone who chooses to follow you simple messages also known as Tweets. The message could be as simple as what you are doing right now or just a quick question to your followers. Likewise you can choose to follow people and receive their messages.

Twitter also gives the ability to send direct messages to users instead of broadcasting to everyone and to search other Twitter users near your location. Twitter's API allows 3rd party developers to develop applications and upload them and, finally, by using mobile options someone can send and receive tweets via SMS messaging on his cell phone.

Twitter actions are:

- Tweet, as described above
- Follow a user, receive all Tweets from this user
- Mention a user, using @ and the username of the user to reply/interact/attach this user with a tweet
- Retweet a Tweet, broadcast usually another user's Tweet to the re-tweeters' followers
- Hashtag, the # symbol is used to mark keywords or topics in a Tweet. It was created organically by Twitter users as a way to categorize messages.
- Trends, words or hashtags that are hot at any time in the Twitter network
- Geolocation [G2], adding the location where the user is tweeting to your tweet
- Search, using a string (word, sentence, hashtag, username) to search in Twitter's data.

The REST [G] API enables developers to access some of the core primitives of Twitter including timelines, status updates, and user information. If you're building an application that leverages core Twitter objects, then this is the API for you. Imagine building a profile of a user: their name, their Twitter handle, their profile avatar, and the graph of people that they are following on Twitter - all with a few RESTful API calls. In addition to offering programmatic access to the timeline, status, and user objects, this API also enables developers a multitude of integration opportunities to interact with Twitter. Through the REST API, the user can create and post tweets back to Twitter, reply to tweets, favorite certain tweets, re-tweet other tweets, and more.

2.2 Twitter's Attractive Features

The reason why Twitter was chosen for this thesis is that it has many attractive features that facilitate academic research.

Twitter is considered as the public social network. First of all, its data consist only text. Tweets are the only way to communicate with other users, and even though they can contain images or videos, they are embedded in tweets only as links. Given that, it has been proved that users of twitter have minor privacy concerns on using this social network. Most users keep their account public, as well as all their information.

Twitter data can easily be analyzed because of their length. Users have to create small and comprehensive sentences in order to keep up with twitter's length of text restrictions.

The hashtags define the subject of a tweet. Searching for a hashtag in a tweet is a straightforward way of categorizing tweets by subject. The twitter search service using a hashtag will also provide all conservations on a subject globally.

The @ mention feature can identify all users participating in conversation.

The re-tweet feature of Twitter gives the option to users of sharing another user's tweet. In just a few steps of data requests, all users that re-tweeted a tweet can be parsed.

The Geolocated tweets are available for all users. The location information that can be embedded in tweet is very useful for location centric research.

Twitter Trends are available to identify instant hot topics. It is the easiest way to find out the most shared over twitter topics.

Finally, Twitter provides APIs (application programming interface) that can be used primarily for third party applications but also for research. All twitter interactions and features can be accessed through the APIs that twitter has issued for general usage. Their usage is subject to several rules and limitations, and their use should be careful to avoid IP or account locks (Rate Limits).

2.3 Twitter APIs

Twitter has two main categories of APIs: the Streaming API and the REST API [G6].

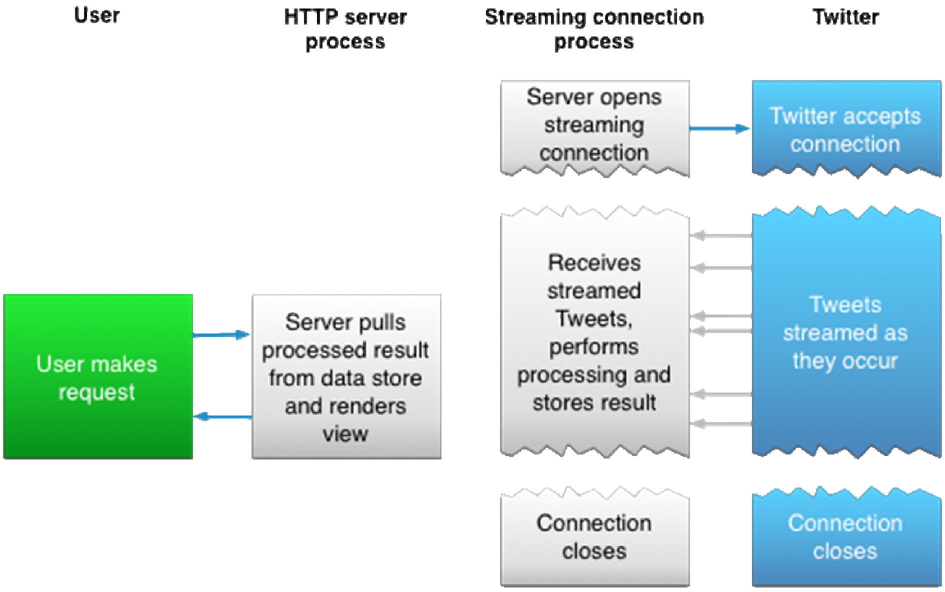
2.3.1 The Streaming APIs

The set of streaming APIs offered by Twitter give developers low latency access to Twitter's global stream of Tweet data. A proper implementation of a streaming client will be pushed messages indicating Tweets and other events that have occurred, without any of the overhead associated with polling a REST endpoint.

Twitter offers several streaming endpoints, each customized to certain use cases.

- Public streams: Streams of the public data flowing through Twitter. Suitable for following specific users or topics, and data mining.
- User streams: Single-user streams, containing roughly all of the data corresponding with a single user's view of Twitter.

- Site streams: The multi-user version of user streams. Site streams are intended for servers which must connect to Twitter on behalf of many users.


User
HTTP server process
Streaming connection process
Twitter
Server opens streaming connection
Twitter accepts connection
Receives streamed Tweets, performs processing and stores result
Tweets streamed as they occur
User makes request
Server pulls processed result from data store and renders view
Connection closes
Connection closes

Picture 2.1: Twitter Streaming API

The endpoint that is more suitable for our research is the «Public Streams», because it provides us a constant connection to the Twitter public data stream. The data stream can be filtered using keywords, hashtags, usernames, geographic location etc. So we can set a geographic location by defining its borders with geographical coordinates (latitude/longitude).

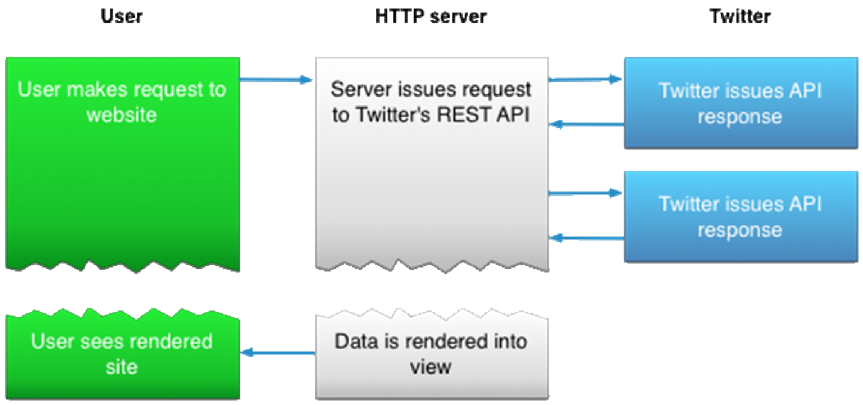
On this streaming endpoint some rate limits are applying. Through the streaming API, only 1% of all available data at any time is accessible. Taking into account the previous limit, it is wise to access the streaming endpoint using always filters that will minimize the loss of data. To make it clearer, when a filter is applied to the data stream, and the results are up to 1% of all available twitter data at that time, all the data will be pushed to the streaming endpoint. Filters are also subject of limits and the number of keywords, users, locations etc. used in filters is a finite set. Finally there are limitations associated with IP addresses; no more than one streaming endpoint can be used from a single IP address.

At this point, we have to point out that an alternative to Streaming API exists, the Twitter Firehose. "Firehose" is the name given to the massive, real-time stream of Tweets that flow from Twitter at any time. It has no limitation at all and it is the Mecca for all twitter developers/analysts/researchers. At some point in the past it was available for developers just by submitting a form to Twitter explaining the intended use. Now, it is a commercial feature that is only available for twitter partners. Anyone can gain access by buying a license of use from those partners.

2.3.2 The REST API

The REST API enables developers to access some of the core primitives of Twitter including timelines, status updates, and user information. If you're building an application that leverages core Twitter objects, then this is the API for you. Imagine building a profile of a user: their name, their Twitter handle, their profile avatar, and the graph of people that they are following on Twitter - all with a few RESTful API calls. In addition to offering programmatic access to the timeline, status, and user objects, this API also enables developers a multitude of integration opportunities to interact with Twitter. Through the REST API, the user can create and post tweets back to Twitter, reply to tweets, favorite certain tweets, retweet other tweets, and more [A1]. All twitter functions and information retrieval can be accessed using the REST API by making individual request to the server. Requests to the Twitter core can be made not only from authenticated twitter users (authenticated requests), but also from unauthenticated users (unauthenticated requests).

Rate limits are also applied to the REST API.


User
HTTP server
Twitter
User makes request to website
Server issues request to Twitter's REST API
Twitter issues API response
Twitter issues API response
User sees rendered site
Data is rendered into view

Picture 2.2: Twitter REST API

REST API version 1

- 150 unauthenticated requests per hour can be made from a single IP address
- 350 authenticated requests per hour can be made from a single IP address
- Twitter search queries, 1 request per second from a single IP address

The first version of the API is now depreciated, and will cease functioning. From now on the version 1.1 of the API will be used.

Rest API version 1.1

- 180 requests per bucket window from a single IP. A bucket window for twitter applications is defined as 15 minutes.

2.4 Getting the data

In order to get Twitter data for our purposes, we have used a combination of all available APIs.

- Streaming API with geolocation filter

  We have used a streaming endpoint to get tweets within the geographical location of Greece. Drawbacks: many of the fetched tweets were outside of the bounds that we have set. We had to refilter all tweet coming from this endpoint.
- Search requests (REST API) for language specific tweets

  We made one request every second, to get language specific tweets (Greek Tweets). The rate limit was never excided, but the data response was limited.
- REST requests for extra data

  To retrieve information (followers, friends etc.) about the users that have been fetched using the previous methods, to get the maximum available timeline for those users and to find the users that have retweeted tweets from our users, we have used REST requests. We had issues with rate limits, since the data we were trying to access was huge, but we came up with a workaround solution, using the TOR Anonymity Network to make our requests.
- Getting access to the Firehose

  We have to note here that we have tried to gain access to these features by contacting all Twitter partners, and even after explaining the purpose of use (academic research, not commercial), we have not been granted with one. The replies we got were that it is strictly for commercial use, and that the “regular” streaming API should be sufficient. The only one that got interested in our project was Dr. Stuart W. Shulman, Assistant Professor at University of Massachusetts Amherst. He is the Founder and CEO of Texifter (current name DiscoverText) that is a cloud-based, collaborative text analytics solution, generating insights about customers, products, employees, news, citizens, and more using data from social networks and the web. This service has access to the Twitter Firehose through GNIP, a Twitter partner. He granted us a 60-day access to GNIP using DiscoverText as a small sponsorship to our research. Till this day, we have not used this grant, because of the time and data limitations. We are planning to use it on a specific experiment in which we could convert the limitations to our advantage.

2.5 Tor – The Anonymity Network

Tor (originally short for The Onion Router) is a system intended to enable online anonymity. Tor client software directs internet traffic through a worldwide volunteer network of servers to conceal a user's location or usage from anyone conducting network surveillance or traffic analysis. Using Tor makes it more difficult to trace Internet activity, including "visits to Web sites, online posts, instant messages and other communication forms", back to the user and is intended to protect users' personal privacy, freedom, and ability to conduct confidential business by keeping their internet activities from being monitored.

"Onion Routing" refers to the layered nature of the encryption service: The original data are encrypted and re-encrypted multiple times, then sent through successive Tor relays, each one of which decrypts a "layer" of encryption before passing the data on to the next relay and, ultimately, its destination. This reduces the possibility of the original data being unscrambled or understood in transit.

Tor was originally designed, implemented, and deployed as a third-generation onion routing project of the U.S. Naval Research Laboratory. It was originally developed with the U.S. Navy in mind, for the primary purpose of protecting government communications. Today, it is used every day for a wide variety of purposes by normal people, the military, journalists, law enforcement officers, activists, and many others.

Tor is a network of virtual tunnels that allows people and groups to improve their privacy and security on the Internet. It also enables software developers to create new communication tools with built-in privacy features. Tor provides the foundation for a range of applications that allow organizations and individuals to share information over public networks without compromising their privacy.

To create a private network pathway with Tor, the user's software or client incrementally builds a circuit of encrypted connections through relays on the network. The circuit is extended one hop at a time, and each relay along the way knows only which relay gave it data and which relay it is giving data to. No individual relay ever knows the complete path that a data packet has taken. The client negotiates a separate set of encryption keys for each hop along the circuit to ensure that each hop cannot trace these connections as they pass through.

Once a circuit has been established, many kinds of data can be exchanged and several different sorts of software applications can be deployed over the Tor network. Because each relay sees no more than one hop in the circuit, neither an eavesdropper nor a compromised relay can use traffic analysis to link the connection's source and destination. Tor only works for TCP streams and can be used by any application with SOCKS support.

For efficiency, the Tor software uses the same circuit for connections that happen within the same ten minutes or so. Later requests are given a new circuit, to keep people from linking your earlier actions to the new ones.

2.5.1 Using Tor for data extraction from Twitter

As previously stated, we had rate limit problems using the Twitter REST API. We have mainly used the REST API to pull more information about users that have already been identified as Greek. For all users, we gathered the user list of followers and friends, their tweets (as many as twitter allows retrieving), their retweeters and those users they have retweeted. For the returned data of these requests pagination occurs, so to get all the available information more than one request is necessary. At this point, when using only one IP address to pull these data, the rate limit is always overpassed and we had to wait until more request were allowed. There was also the danger of getting banned when persisting on making requests at a point when no more is allowed.

In order to bypass this setback, we had to use Tor, as an intermediate network, to make our requests. When connected to Tor, internet traffic is routed through its peers and all requests are made from other peers and its results are routed back to us. It is an easy way to mask our IP address and use other peer’s IP address to pull the information we want. At the time when the rate limit of our currently connected peer is approaching, we make a request at the Tor network to change our endpoint IP address. This way we managed not only to get more data in less time but also not to get banned by Twitter. The average request rate we reached was 526 requests per hour. We have to note that when we measured this metric, the process of making the requests was not dedicated only on this task, but also did some data processing on our database, so we logically presume that higher request rate can be reached when using this method.

	Request Rate (request/hour)
Unauthorized Requests	150
Authorized Request	350
Unauthorized Requests using Tor	526

2.6 Getting and storing the data

Retrieving and handling big loads of data, such as twitter data, in an efficient way requires a carefully designed platform. The system should be consisted of two main modules: a data retrieving module and a data processing module. The first one will make requests to the Twitter server and the second one will process the fetched data and decide the order of future requests. Every module should run on a different thread to ensure fast data retrieving and processing.

2.6.1 Data retrieving module

This module is responsible of getting the appropriate data from Twitter. It is consisted of three threads: the first gets tweets using the streaming API, the second one using the REST API and the third using the Twitter Search.

The streaming API thread collects all tweets broadcasted in the Greek geographical location, and pushes a new user item in the queue, so that all their information can be retrieved using REST requests.

The REST API thread collects specific data (tweets, followers and friends user ids, retweters) by "consuming" items from the queue.

The Search thread collects tweets in Greek using the Twitter Search and a language filter.

2.6.2 Queue System and Data processing module

The main feature of our platform is a queue in which all requests are stored. The process that makes the requests to the twitter server uses this queue as input to determine what it should pull from twitter. When the request is made, the process updates the status of the last request and if the data that was pulled is paged, it creates a new queue item to retrieve the next page.

We have also created flag items that define the last fetched data concerning any date entity on our queue. This way we can keep track of the last inserted information.

In short:

- Each request is stored in a queue
- Each queue row contains
    - User id
    - Type of request
    - Cursor – if necessary (cursors are used by twitter for the pagination of the results)
- Keeping track of the newest inserted information using flags

The data processing module in responsible for managing and distributing our database, and also for feeding the queue with new REST requests. When a new user who fulfills the location/language criteria is found, the module pushes new items into the queue, so that their tweets etc. are retrieved. The newly fetched tweets are stored in a temporary table, in order not to burden/overload with more processes the data-retrieving module, and this module parses all new tweet information.

2.6.3 Platform's Advantages and disadvantage

Advantages:

- At each time point the data retrieving module is designated to one task: to bring all available data suitable for our research. It searches for suitable data and dumps them as quickly as possible to continue with its task. The data processing is designated to the other module.
- Using this queue system the twitter world is recreated uniformly in our database. The system does not fetch one available data for each user and then moves on to the next, but it fetches parts of their data, as provided from twitter's pagination, according to the queue system. For example, when the first page of a user's tweets is fetched, then the new request for the second page of tweets is pushed into the queue, and is retrieved after all previously pushed requests are retrieved. This way we prioritize to fetch some information for all identified users, and then slowly dig in for more information.

- Using flags to keep track of the latest information imported, the data that is collected can easily be compared with recent/real-time twitter data. At any time point we can determine our data lagging from real time data.
- Making requests that contain duplicate information are avoided (rate limits). The use of the queue system combined with the flags of last inserted data, determines the data for request, without overlapping with already fetched data.

The main disadvantage of out platform, which was intended, is that it is not suitable for rapid full data collection. In cases when the dataset has to be complete (all user available data to be included) new processes have to run, bypassing the queue system, that will fetch all available data for a user and then move on to the next one.

2.6.4 Our Twitter Collection

Using the previously explained method, we have managed to create a very considerable twitter collection. It contains:

- 1.324 users that tweeted in Greek
- 2.264 users that tweeted using geolocation
- 9.669 retweeters of our users
- 17.014 users that have been retweeted by our users
- 4.399 Candidate users
- About 300.000 tweets containg Greek text
- About 30.000 geolocated tweets
- More than 17.000.000 following edges
- More than 36.000 retweets

2.6.5 Implementation Details

The Twitter Collection System was implemented in PHP. It is set on an Apache server with a MySQL server. Those choices were driven from the fact that we wanted to create a fast online system.

3

Identifying Greek Users

3.1 It's all Greek to me

In this section we will analyze all possible ways to identify Greek users from their tweets-texts. Even though we cannot apply our own language identification algorithms directly to the twitter APIs to get more efficient results, we can use them on other twitter datasets to filter the information that we need from those datasets. Also by analyzing the language-text relation it is the best way to get the bigger picture of our problem.

Every language has its specificities many of those being common in more than one language. All these language features should be used properly in order to create better language identification mechanisms. Another thing that we should always keep in mind is that, in terms of syntax, all languages are used differently in writing and differently on the internet. So we have to consider not only the standard features of a language, but also its internet features.

The «official» internet/computer language (the most common one) is English. That had as a result the creation of an internet version for many languages, which do not use the Latin alphabet. Every new version of a language is the same with the original, but uses the Latin alphabet as a replacement of the original alphabet.

More specifically, the ways of identifying texts from Greek users are the following:

1. Texts that contain the Greek alphabet. The uniqueness of the Greek alphabet is a solid identification feature.
2. Using language identificators. The most common category of algorithms in language identification is the n-grams.
3. Texts that are in Greeklish. Greeklish is Internet language for Greek. It is Greek, but instead of using the Greek alphabet, the Latin alphabet is used.

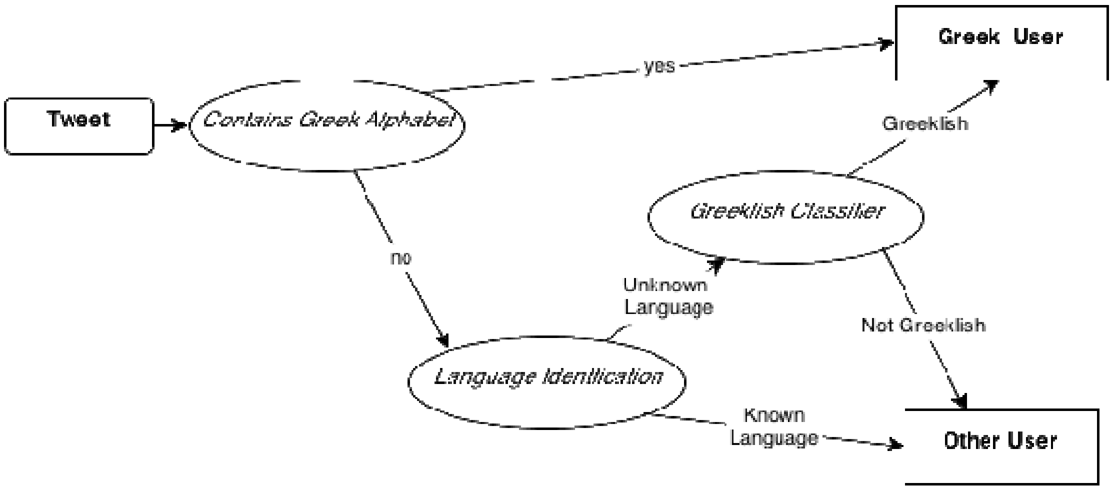
To identify the third tweet category we had to develop Greeklish classifiers.

3.2 "Greeklish Convertor" Classifier

The «Greeklish Convertor» Classifier was the first Greeklish classification algorithm that we developed. Its philosophy is simple: texts that we cannot determine the language used, we use a Greeklish Convertor to «translate» the text to Greek and depending on the number of words translated, we categorize it to Greeklish or Non-Greeklish. The feature of the convertor that we use is that words that are not in Greeklish, remain the same after the translation. The Greeklish convertor that we have used in our classifier, is the Greeklish Convertor developed by Vaggelis Pterneas and Giorgos Karakatsiotis [R7].

3.2.1 The algorithm

1. Check if the text contains Greek characters. In that case, the text is categorized as Greek, as well as the user. If not, we proceed to the second step.
2. Run two n-gram language identification algorithms using the text as input. If both algorithms provide the same result, then the text is categorized as non-Greek, because the language scope of text identification algorithms is on official languages. If the results differ we proceed to the third step.
3. Use the Greeklish convertor to find the percentage of the words that can be "translated" from Greeklish to Greek. If the number is bigger than the threshold that we have computed, the text is categorized as Greeklish. If not, the text is categorized as Non-Greeklish.



The “Greeklish Convertor” Classifier Flow Diagram

3.2.2 *Technology Used*

The n-gram language identification applications that we have chosen for the second step are the following:

1. PEAR Language Detection Library [R3]
   PEAR is short for "PHP Extension and Application Repository" and is a structured library of open-source code for PHP. The library we have used detects the language of a given piece of text. The package attempts to detect the language of a sample of text by correlating ranked 3-gram frequencies to a table of 3-gram frequencies of known languages. It implements a version of a technique originally proposed by Cavnar & Trenkle (1994): "N-Gram-Based Text Categorization".
2. Xerox Language Detection API [R4]
   The Xerox language identifier tool mainly uses n-grams and word frequency to calculate the language probability. The n-gram and word frequency resources are based on larges corpus (http://open.xerox.com/Services/LanguageIdentifier).

3.2.3 *Classifier Tuning*

At the third and final step of the algorithm, we determine if a tweet is Greeklish or not based on the number of translated words. This threshold cannot be arbitrary, so we had to create a tuning process for our algorithm to decide its value.

Our tuning metric is the ratio of the number of converted words over the total number of words in a tweet:

$$tuning\ metric = \frac{(\text{number converted words})}{(total\ number\ of\ words)}$$

We created a baseline test on 13735 tweets of Greek users. We manually classified these tweets to Greeklish or not:

- 1673 tweets in Greeklish
- 12062 tweets in other (not Greeklish) languages

Then we run the Greeklish convertor on these tweets to see how the metric behaves:

- On Greeklish tweets

  Mean Value (Average): 0.89383

  Standard deviation: 0.1601
- On Other tweets

  Mean (Average): 0.51869

  Standard deviation: 0.22861

These results do not provide an adequately safe threshold for the classifier. But we have observed that non-Greeklish words with more than two characters are almost always translated to Greek. So we had to turn this error in our advantage by removing all words with less than four letters from the original texts/tweets.

Then we run again the Greeklish convertor on the new modified tweets and the results we got were:

- On Greeklish tweets

  Mean Value (Average): 0.85772

  Standard deviation: 0.23026
- On Other tweets

  Mean (Average): 0.31319

  Standard deviation: 0.2636

As a result we significantly decreased the average of our tuning metric in non-Greeklish tweets. Now we can set the threshold to 0.5 and classify the tweets that their metric is over 0.5 as Greeklish and the ones that the metric is below 0.5 to non-Greeklish. To do so, we have to add another step in the algorithm, step 2.5, in which all words with less than four letters are removed.

3.2.4 The final algorithm

1. Check if the text contains Greek characters. In that case, the text is categorized as Greek, as well as the user. If not, we proceed to the second step.
2. Run two n-gram language identification algorithms using the text as input. If both algorithms provide the same result, then the text is categorized as non-Greek, because the language scope of text identification algorithms is on official languages. If the results differ we proceed to the third step.
3. Remove all (<=3) letter words to set a better threshold.
4. Use the Greeklish convertor to find the percentage of the words that can be "translated" from Greeklish to Greek. If the number is bigger than the threshold that we have computed, the text is categorized as Greeklish. If not, the text is categorized as Non-Greeklish.

3.2.5 "Greeklish Convertor" Classifier Results

The input data that was used for the classifier are from 273 Greek Twitter users. Our corpus consists of 13740 tweets, from which 12368 tweets were used as training dataset and the 1372 as test dataset.

		Actual	
		Greeklish	**Non Greeklish**
Predicted	**Greeklish**	123	114
	Non Greeklish	38	1097

	Greeklish	**Non Greeklish**
accuracy	0.88	0.88
precision	0.76	0.91
F1 score	0.82	0.86

3.3 Weka Classifier

The Weka classifier is the second classifier that we have created to identify Greeklish tweets. The classifier uses the Weka machine learning software that is used for data analysis and predictive modeling.

3.3.1 Weka

Weka (Waikato Environment for Knowledge Analysis) is a popular suite of machine learning software written in Java, developed at the University of Waikato, New Zealand. Weka is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well suited for developing new machine learning schemes.

Weka supports several standard data mining tasks, more specifically, data preprocessing, clustering, classification, regression, visualization, and feature selection. All of Weka's techniques are predicated on the assumption that the data is available as a single flat file or relation, where each data point is described by a fixed number of attributes (normally, numeric or nominal attributes, but some other attribute types are also supported). Weka provides access to SQL databases using Java Database Connectivity and can process the result returned by a database query. It is not capable of multi-relational data mining, but there is separate software for converting a collection of linked database tables into a single table that is suitable for processing using Weka.

Weka's main user interface is the Explorer, but essentially the same functionality can be accessed through the component-based Knowledge Flow interface and from the command line. There is also the Experimenter, which allows the systematic comparison of the predictive performance of Weka's machine learning algorithms on a collection of datasets.

Weka has also a java library from which all its features are accessible. That is the main reason that we have selected to use Weka for our classifier. Another attractive feature of Weka is that it is not only easy to use, but it is also fast. The only real drawback is the complex input file formats, which are tricky to create.

3.3.2 Selected Algorithms

Weka has prebuilt many classification algorithms from which the user can select the appropriate for his problem. We decided that we will create two versions of the classifier, one using the Naive Bayes algorithm [R8] and the second using the Sequential Minimal Optimization algorithm [R9]. The reason for our selection was that they fitted our prerequisites: a very good and fast algorithm, Naive Bayes and the most powerful classification algorithms, Sequential Minimal Optimization.

3.3.3 Data Preparation

Weka uses as input a specific type of file: Attribute Relationship File Format (ARFF, *.arff file). The ARFF file contains two sections: the header and the data section. The first line of the header tells us the relation name. Then there is the list of attributes (@attribute...). Each attribute is associated with a unique name and type. The latter describes the kind of data contained in the variable and what values it can have. The variables' types are: numeric, nominal, string and date. The class attribute is by default the last one on the list. In the header section there can also be some comment lines, identified with a '%' at the beginning, which can describe the database content or give the reader information about the author. After that, there is the data itself (@data), where each line stores the attribute of a single entry separated by a comma.

The tweets, as given, are not in a form amenable to feature extraction for classification (there is too much `noise'). In order to make tweets compatible with the ARFF file syntax and to sanitize them for data analysis, we have created some general specifications that will take a tweet in its provided form and convert it to a normalized form.

General Specifications

1. All html tags and web page URLs are removed
2. All punctuation symbols are removed
3. Mention, retweet and hashtags symbols are removed
4. Sequences of the same letter are removed
5. Convert all text to lowercase

3.3.4 Weka Classifier Results

The input data that was used for the classifier are from 273 Greek Twitter users. Our corpus consists of 13740 tweets, from which 12368 tweets were used as training dataset and the 1372 as test dataset.

3.3.4.1 Naïve Bayes Algorithm

		Actual	
		Greeklish	**Non Greeklish**
Predicted	**Greeklish**	116	29
	Non Greeklish	45	1182

	Greeklish	**Non Greeklish**
accuracy	0.95	0.95
precision	0.72	0.98
F1 score	0.82	0.96

3.3.4.2 SMO Algorithm

		Actual	
		Greeklish	**Non Greeklish**
Predicted	**Greeklish**	142	0
	Non Greeklish	19	1211

	Greeklish	**Non Greeklish**
accuracy	0.99	0.99
precision	0.88	1
F1 score	0.93	0.99

3.4 Classification Results

All classification algorithms we have created give very good results with high scores. From the PivotTable we can conclude that the Sequential Minimal Optimization Classifier gives better results than all the others. The «Greeklish Convertor» classifier has the lowest scores of all three, but the score is not a dissuasive for usage. Finally the fastest classifier was the Naïve Bayes classifier. Taking all parameters into consideration we can conclude that the overall better classifier was the Naive Bayes.

	"Greeklish Convertor"		**Naive Bayes**		**SMO**	
	Greeklish	**Non Greeklish**	**Greeklish**	**Non Greeklish**	**Greeklish**	**Non Greeklish**
accuracy	0.88	0.88	0.95	0.95	0.99	0.99
precision	0.76	0.91	0.72	0.98	0.88	1
F1 score	0.82	0.86	0.82	0.96	0.93	0.99

3.5 Live Online Classifiers

The Weka Naive Bayes Classifier can be tested live at the following link, http://195.251.252.39/rebelos/naivebayes.php or by using the following QR code:

The Weka SMO Greeklish classifier can also be tested live at the following link, http://195.251.252.39/rebelos/smo.php or by using the folling QR code:

4

The Most Influential Users

4.1 Introduction

In this section we are going to present an algorithm for finding the influential users among a community. We were motivated by the fact that advertisers want to find the set of users in a certain community who are much more capable of affecting the rest or even a satisfactory subset of the whole community. This is a policy, which is adopted by an increasing number of advertising agencies in order to make their products known to the public with very low advertising cost. The algorithm that we are going to present was introduced by Konstantina Galvogini in "Crawling and Analysis of Social Network data" [R5]. This algorithm presented promising results in defining the set of most influential users in a community.

The only other paper [R6], which uses a definition of a user influence as a combination of the number of retweets and mentions to him and the number of his indegree has been made by Meeyoung Cha, Hamed Haddadi, Fabricio Benevenuto and Krishna P. Gummadi who measured user influence in Twitter using a large amount of data collected from Twitter. They presented an in depth comparison of three measures of influence: indegree, retweets and mentions. Their observations were the following. Firstly, popular users who have high indegree are not necessarily influential in terms of spawning retweets or mentions. Secondly, most influential users can hold significant influence over a variety of topics. And thirdly, influence is not gained spontaneously or accidentally, but through concerted effort such as limiting tweets to a single topic.

First of all, we are going to define the concept of influence that a user has in a community. Then, we will present the algorithm and its complexity in the worst case. Finally, we will discuss the results of our algorithm and we will compare them with other influence metrics.

4.2 Algorithm Analysis

Here, we present the definition of Influence using a different approach. A user can affect those users who can read his tweets. For each user we calculate his Influence set that consists of the users who he affects. So someone's Influence set is defined by the following sets:

1. Followers
2. Retweeters
3. Followers of retweeters

We calculate the Influence set for each user and we run a variation of Greedy Set Cover. We have to remind that the previous algorithm is implemented upon a set of nodes, which is called universe, with each node having a cover set. Its output is the minimum set of nodes that covers the entire universe. In this case, the universe is the set of users and each node's set is the influence set of each user.

The Algorithm:

1. Find the user with the greatest influence in the set of all our users
2. Remove the users that this user influences
3. Repeat the first step in the new user set, until a number of influential users or until an adequate cover of the graph

Pseudocode of Algorithm for finding m Influential users:

```
Sj: the influence set of node j where $(1 \leq j \leq n)$
for (int i=1; i<m; i++){
        find node j with largest influence set Sj
        for each k in Sj {
                /* remove k from influence sets of all other nodes */
                for (inth=1; h<n; h++){
                        $S_h = S_h$fflk;
                }
        }
        Outputj;
}
```

Next we are going to give a detailed interpretation of the pseudocode. As mentioned above every user has his influence set, which is defined by subsets of the origin set of community. User's influence set consists of three different subsets, the set of his followers, his retweeters and the followers of his retweeters. Therefore, we first find the influence set for every user in the graph. We check the first user with the biggest influence set and we keep him as the more influential user. In order to find the rank of the rest users in the community we remove every user who exists in the first user's influence set from the influence sets of the other users. When all the users of the first influence set are removed from the rest influence sets, we now find the user with the biggest influence set and we remove him from our universe. Then we take his influence set and follow the same procedure as we did for the first user in rank. Our algorithm is parametric, as the number of the first m influential users can be given as parameter. Thus, the output of the algorithm is the rank of influential user in a certain community and the sets of users that they contribute to the final result.

If we want to answer a question like: "How many "advertising gifts" are needed to cover all users? ", we have to take into account that it is a hard problem and it belongs to NP-Complete problems. The algorithm is a greedy variation and more certain than the Greedy set cover algorithm. So, we know it cannot be worse than O (OPT*logn), where OPT is the minimum set of advertising gifts needed to cover all users.

4.3 Results

We decided to test the algorithm on a specific set of users to make the experiment more meaningful. The feature of the user set that we are going to use is the geographic location. This information can be extracted from the user's personal information or from user tweet's location. The user's personal information is just a declaration, a field that the user fills in, but the location where the user tweets is more substantive. That is the reason why we selected to set the user's location by their tweet's location.

Using the streaming API with the geo-location filter set on Greece, we have gathered users that tweeted in territorial Greece. Then we partitioned the users according to geographical regions depending on their tweets' location, and gathered all the necessary input data (followers, retweeters and followers of retweeters) for these users that are required to run the algorithm.

The following table displays the algorithm's results on users that have tweeted in the geographical region of Attica. We have calculated their influence in this set of users and extracted the M influential users according to the method. We compare our influence metric against the widely accepted influence ranking system, Klout. Klout is a service that provides social media analytics to measure a user's influence across his or her social network. The analysis is done on data taken from sites such as Twitter, Facebook, and Google+, and measures the size of a person's network, the content created, and purports to measure how other people interact with that content.

RANK	User ID	User Name	Influence	Klout Rank	User Category
0	575558887	avraampapas	3793	58.3611	retweeter
1	237719164	kanekos69	1312	60.6986	retweeter
2	448264593	GOULIELMOS_	727	40.9131	retweeter
3	591744503	frerakos	465	61.9871	retweeter
4	355422767	nikosp20	220	56.4131	retweeter
5	15841461	prezatv	190	55.7631	geo + retweeter
6	108365678	Nektarios_	158	53.7477	geo
7	56812543	dpapangel	123	57.5559	retweeter
8	181698945	nikosofficiel	118	81.691	retweeter
9	297049579	jonagiu12	107	49.8503	geo

From the results we can see that the in-group influence is not proportional with the Klout rank. Also the two ranked influence metrics do not share a monotonic relationship. This is the reason why this algorithm is important when trying to identify influential users in a specific group. If we had selected the most influential users via the Klout rank, we would not have gotten as much coverage over the group that we were interested in, and the field of influence of the selected users would overlap.

Another thing that is worthy of mentioning is the user category of most influential users. We would have never guessed that retweeters would have that much of an influence on a selected group. That is why research should direct towards redefining the influence calculation process, in such ways as to include the retweet action.

5

Metric for Unidentified Users

5.1 The Uncategorized User Problem

In the previous sections we have analyzed ways to collect users from twitter that use the Greek language or Greeklish in the tweets, they use geolocation tagging when tweeting, but we haven't found a way to collect those who tweet in another language and do not use geolocation in their tweets. In order to complete our thesis subject we have to efficiently address that problem too. We call it «the Uncategorized User Problem».

5.2 The "Following" Metric

Since there is not any other primal information that we can pump from twitter, from which the location of the user can be extracted, we had to find another way to assume the location of a user. The idea is that the user's "nationality" can be identified by the user's connections to groups of already identified users. In every user connection (followers, friends' connections) underlies information about real life connections between users. So, by identifying how much connected a user is to a specific group of users, in comparison to the connectivity that users of the group have with the whole group, will provide a meaningful metric about the "distance" of a user from a group.

The user connection we have selected for this metric is the friend connection. Twitter friends are the users that a user chooses to follow. This is a more powerful connection than the user's followers, because users follow other users:

- when their tweets have value for them (twitter authority)
- when they have a real life connection

The "following" metric is defined as:

$$\mathrm{m} = \frac{(\text{number of users that a user follows that are categorized in a group of interest})}{(\text{number of users that a user follows})}$$

The basic properties of this metric are:

- As the number gets bigger, the user's connection with a group gets stronger
- It is a way to convert the following edges to a meaningful number
- As the group of users gets bigger, the metric's "value" gets stronger

5.3 Testing the metric

To test the validity of our new metric we had to use it on a real set of users. First we have to see how this metric behaves for users that belong to a group, and then for the users outside a group. This way we are going to set a threshold for the metric value that will determine if the user belongs to this group or not.

5.3.1 Tuning

For our tuning set we have used all the users that we have collected, which use the Greek language for tweeting and also tweet in Greece's geographical area (geolocation).

The metric for users that have already been classified as Greek (3630 users):

- Average: 0.07154
- Standard deviation: 0.0804

For our test set we have used users that we had collected using geolocation, in places near Greece's boarder. We have selected this dataset to take advantage of Twitter's API bug, through which not all of results using geolocated filter are in the filtered area. For instance, when collecting tweets from Rhodes island, many of the tweets that were pulled were located in Turkey.

The metric for the “candidate” users (4339 users):

- Average: 0.00091
- Standard deviation: 0.00728

At this point we made an observation to the candidate users’ metric: in most of the non-Greek users this metric’s value is zero. We have used this value as our classification threshold:

- When the metric is 0, classified as non-Greek user
- When the metric is not 0, classified as Greek user

5.3.2 Results

		Actual	
		Greek	**Non Greek**
Predicted	**Greek**	86	42
	Non Greek	27	4184

	Greek	**Non Greek**
accuracy	0.98	0.98
precision	0.76	0.99
F1 score	0.86	0.99

5.3.3 Conclusions

For a preliminary test of the metric in small set, the classification results are very good. The basic problem is that precision of detecting the Greek users is relatively low. But we believe that in a larger Greek set of users, the precision results will possibly go better. When the set of identified users with a common attribute gets bigger, so does the connectivity between the users. A candidate user for this group, who belongs to this group, will have more connection to the group, as the group gets bigger.

6

Conclusion

6.1 Summary and Results

In this paper we have combined all the characteristics of the Twitter social network to create algorithms and tools to provide an accurate prediction of a user's nationality. We presented a system that can collect twitter information from users depending on their nationality, that uses all Twitter APIs, and that its collect rate is amplified through the Tor network. To identify users that tweet in informal languages like Greeklish (Greek language using the Latin alphabet), we developed three efficient classifiers, based either on a Greeklish Convertor either on the Weka machine learning software. We have also tested the Konstantina Galbogini's "M Influential Users" algorithm on a set of users from a specific geographic area, and validated the algorithm's importance to determine the maximum field of influence to a set of users. Finally we presented a new metric that defines a user's distance from a group of users, in order to be able to categorize users that their information does not give out their nationality. We computed this metric and used it for classification of a set of users that are difficult to categorize, and through this process we explored some of its potentials.

6.2 *Future work*

In this last section we will defile the future work that should be done, to expand our research. The future work can be divided in two main pillars:

Advance the research work

- Expand the tweets' dataset, to create better Weka classifiers.

  As bigger as the training set gets, the more accurate results the clasiffier gives.
- Test the influential algorithm in more geographical areas.
- Test the "Following" metric's value and it's features in larger user sets,

Upgrade the data collection system

- Create a new queue system with which the graph's evolution in time would be easily represented when required.
- Multithreading integration throughout the system, in order to make the system faster.
- Integrate the Twitter REST API v1.1 to the system and explore ways to make the collection rate faster.

7

Appendix

[A1] REST API Resources

It has many resources but we will show the most important for us that we used in crawling process. The categories are Timelines, Search, Tweets, Direct Messages, Friends & Followers, Users, Suggested Users, Favorites, Lists, Accounts, Notification, Saved Searches, Places & Geo, Trends, Block, Spam Reporting, OAuth, Help, Legal and Deprecated.

Timelines

Timelines are collections of Tweets, ordered with the most recent first.

- GET statuses/home timeline:Returns the most recent statuses, including retweets if they exist, posted by the authenticating user and the users they follow.
- GET statuses/mentions: Returns the 20 most recent mentions (status containing @username) for the authenticating user.
- GET statuses/retweeted by me: Returns the 20 most recent retweets posted by the authenticating user
- GET statuses/retweeted to me: Returns the 20 most recent retweets posted by users the authenticating user follow.
- GET statuses/retweets of me: Returns the 20 most recent tweets of the authenticated user that have been retweeted by others.

- GET statuses/user timeline: Returns the 20 most recent statuses posted by the authenticating user.
- GET statuses/retweeted to user: Returns the 20 most recent retweets posted by users the specified user follows.
- GET statuses/retweeted by user: Returns the 20 most recent retweets posted by the specified user.

Tweets

Tweets are the atomic building blocks of Twitter, 140character status updates with additional associated metadata. People tweet for a variety of reasons about a multitude of topics.

- GET statuses/s:id/retweeted by: Show user objects of up to 100 members who retweeted the status.
- GET statuses/:id/retweeted by/ids: Show user ids of up to 100 users who retweeted the status.
- GET statuses/retweets/:id: Returns up to 100 of the first retweets of a given tweet.
- GET statuses/show/:id:Returns a single status, specified by the id parameter below. The status's author will be returned inline.
- POST statuses/destroy/:id: Destroys the status specified by the required ID parameter.
- POST statuses/retweet/:id: Retweets a tweet. Returns the original tweet with retweet details embedded.
- POST statuses/update: Updates the authenticating user's status, also known as tweeting.
- POST statuses/update with media: Updates the authenticating user's status and attaches media for upload.
- GET statuses/oembed: Returns information allowing the creation of an embedded representation of a Tweet on third party sites.

Search

Find relevant Tweets based on queries performed by your users.

- GET search: Returns tweets that match a specified query.

Friends & Followers

Users follow their interests on Twitter through both one-way and mutual following relationships.

- GET followers/ids: Returns an array of numeric IDs for every user following the specified user.
- GET friends/ids: Returns an array of numeric IDs for every user the specified user is following.
- GET friendships/exists: Test for the existence of friendship between two users.
- GET friendships/incoming: Returns an array of numeric IDs for every user who has a pending request to follow the authenticating user.
- GET friendships/outgoing: Returns an array of numeric IDs for every protected user for whom the authenticating user has a pending follow request.
- GET friendships/show: Returns detailed information about the relationship between two users.
- GET friendships/no retweet ids: Returns an array of user ids that the currently authenticated user does not want to see retweets from.
- POST friendships/create: Allows the authenticating users to follow the user specified in the ID parameter. Returns the befriended user in the requested format when successful.
- POST friendships/destroy: Allows the authenticating users to unfollow the user specified in the ID parameter. Returns the unfollowed user in the requested format when successful.
- GET friendships/lookup: Returns the relationship of the authenticating user to the comma separated list of up to 100 screen names or user ids provided.
- POST friendships/update: Allows one to enable or disable retweets and device notifications from the specified user.

[A2] n-Gram Language Models for Text Classification

In the fields of computational linguistics and probability, an n-gram is a contiguous sequence of n items from a given sequence of text or speech. An n-gram could be any combination of letters. However, the items in question can be phonemes, syllables, letters, words or base pairs according to the application. The n-grams typically are collected from a text or speech corpus.

Human languages invariably have some words which occur more frequently than others. One of the most common ways of expressing this idea has become known as Zipf's Law, which we can re-state as follows:

The nth most common word in a human language text occurs with a frequency inversely proportional to n.

The implication of this law is that there is always a set of words which dominates most of the other words of the language in terms of frequency of use. This is true both for words in general, and for words that are specific to a particular subject. Furthermore, there is a smooth continuum of dominance from most frequent to least. The smooth nature of the frequency curves helps us in some ways, because it implies that we do not have to worry too much about specific frequency thresholds. This same law holds, at least approximately, for other aspects of human languages. In particular, it is true for the frequency of occurrence of N-grams, both as inflection forms and as morpheme-like word components which carry meaning. (See Figure 1 for an example of a Zipfian distribution of N-gram frequencies from a technical document.) Zipf's Law implies that classifying documents with N-gram frequency statistics will not be very sensitive to cutting off the distributions at a particular rank. It also implies that if we are comparing documents from the same category they should have similar N-gram frequency distributions.

An n-gram of size 1 is referred to as a "unigram"; size 2 is a "bigram" (or, less commonly, a "digram"); size 3 is a "trigram". Larger sizes are sometimes referred to by the value of n, e.g., "four-gram", "five-gram", and so on.

8

Glossary

[G1] API

An application programming interface (API) is a protocol intended to be used as an interface by software components to communicate with each other. An API is a library that may include specification for routines, data structures, object classes, and variables.

[G2] Geolocation

Geolocation is the identification of the real-world geographic location of an object, such as a radar, mobile phone or an Internet-connected computer terminal. Geolocation may refer to the practice of assessing the location, or to the actual assessed location. Geolocation is closely related to the use of positioning systems but can be distinguished from it by a greater emphasis on determining a meaningful location (e.g. a street address) rather than just a set of geographic coordinates.

[G3] Accuracy

In the fields of science, engineering, industry, and statistics, the accuracy of a measurement system is the degree of closeness of measurements of a quantity to that quantity's actual (true) value. It is the proportion of true results (both true positives and true negatives) in the population.

$$\text{accuracy} = \frac{\text{number of true positives} + \text{number of true negatives}}{\text{number of true positives} + \text{false positives} + \text{false negatives} + \text{true negatives}}$$

[G4] Precision

In the fields of science, engineering, industry, and statistics, the precision of a measurement system, also called reproducibility or repeatability, is the degree to which repeated measurements under unchanged conditions show the same results. Precision or positive predictive value is defined as the proportion of the true positives against all the positive results (both true positives and false positives).

$$\text{precision} - \frac{\text{number of true positives}}{\text{number of true positives} - \text{false positives}}$$

[G5] F1 score

The F1 score (also F-score or F-measure) is a measure of a test's accuracy. It considers both the precision p and the recall r of the test to compute the score: p is the number of correct results divided by the number of all returned results and r is the number of correct results divided by the number of results that should have been returned. The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0.

The traditional F-measure or balanced F-score (F1 score) is the harmonic mean of precision and recall:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

[G6] Representational state transfer

Representational state transfer (REST) is a coordinated set of architectural constraints that attempts to minimize latency and network communication while at the same time maximizing the independence and scalability of component implementations. This is achieved by placing constraints on connector semantics where other styles have focused on component semantics. REST enables the caching and reuse of interactions, dynamic substitutability of components, and processing of actions by intermediaries, thereby meeting the needs of an Internet-scale distributed hypermedia system.

9

References

[R1] Ilias Foudalis, Kamal Jain, Christos Papadimitriou, and Martha Sideri. Modeling Social Networks Through User Background and Behavior.

[R2] Krzysztof R. Apt, Evangelos Markakis. Difusion in Social Networks with Competing Products, July 27, 2011

[R3] “Pear PHP Package: Text_LanguageDetect. Internet: http://pear.php.net/package/Text_LanguageDetect, Jan. 1, 2012 [Mar. 28, 2013].

[R4] Bruno M. Schulze. “ Automatic language identification using both N-gram and word information” U.S. Patent 6 167 369, Dec. 26, 2000.

[R5] Constantinia Galbogini. “Crawling and Analysis of Social Network data” M.A. thesis, Athens University of Economics and Business, Athens, 2012.

[R6] Meeyoung Cha, Hamed Haddadi, Fabricio Benevenuto and Krishna P. Gummadi. Measuring User Influence in Twitter: The Million Follower Fallacy.

[R7] George Karakatsiotis & Vangos Pterneas, Greeklish Convertor. Internet: http://greeklishconverter.vangos.eu/ [Mar. 28, 2013].

[R8] Harry Zhang. The Optimality of Naive Bayes. Internet: http://www.aaai.org/Papers/FLAIRS/2004/Flairs04-097.pdf [Mar. 28, 2013].

[R9] John C. Platt. Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines, April 21, 1998.