University POLITEHNICA of Bucharest
Faculty of Automatic Control and Computers
Computer Science and Engineering Department

Searching for Topical Authorities on Twitter

Author:
Andrei-Niculae Petre

Scientific Coordinators:
Prof. Dr. Eng. Adina Magda Florea
Dr. Eng. Andrei-Adnan Ismail

Bucharest
June 2014

Contents

Abstract

Twitter is a very active and diverse Microblogging community, with an average of 58 million tweets per day[1]. Being able to obtain information only from the most important users offers a great advantage of staying informed and in some cases (e.g. war) it is a matter of life or death. We propose a model of finding topical authorities on Twitter that computes textual and link analysis features and calculates a weighted mean on the features by which it ranks the top (authoritative) users. Features are easy to understand and the model's best performance is to return users from which 100% of them represent true authorities on the given topic, on two different data sets.

Keywords— Twitter, Authority, Influencer, Features

[1]http://www.statisticbrain.com/twitter-statistics/

Chapter 1

Introduction

In this time of high volume of information it becomes more and more important to be able to filter out the irrelevant data. Most of us want to keep only the high quality information, from first responders, the actual influencers that spread out the particular information in the first place. For e.g., searching for *ukraine russia* topic does bring a lot of information out of which a lot is repeated (retweeted) or reformulated from some main sources, so as to spread it to their followers. Only a few of the information is actually original. Finding the sources that can be trusted to offer accurate information is very important to stay informed and, in the above example, may also be a *life or death* situation.

A different application where identifying topical authorities/influencers[1] may be crucial is in the business area because it can influence "public opinion"[7], "adoption of innovations"[12], "new product market share"[2], or "brand awareness"[8] (in Bakshy et. al.[1]).

The purpose of the thesis is to be able to identify, in a simple to explain kind of way, topical authorities on social platforms. The attention is directed towards Twitter.com because of its richness in metadata of communication and because everything is public, unlike other Microblogging platforms (e.g. Facebook) where conversation can also be private. On Twitter, a user can interact with other users or pages by referencing (@name) or with a topic by adding a hashtag (#topic) in the Tweet itself. Moreover, the user relationships of friends-followers makes it possible to gather additional features in identifying authorities.

In Twitter terminology, a **follower** is one that follows the current user, while

[1] In the following thesis, we shall use interchangeably influencer and authority or authoritative.

a **friend** is one that the current user is following. You can see how a tweet looks like in Section 4.2.2.

The Twitter authorities are computed in the following way:

1. compute metrics and features for each user we consider a potential authority
2. we aggregate each user's features in a way to obtain a ranking (like a mean average, weighted average, clustering etc.)
3. we get a top of users to represent the authority based on the resulted rank. We can do this because the features are designed in such a way so that it is best to maximize each value of each feature (biggest is best).

The **structure of the thesis** is as follows. In Section 2, we start by introducing some of the related work both in Microblogging (Twitter especially) and outside of Microblogging (Q&A communities, blogs), including the works that most inspired the current thesis. In Section 3 we start by explaining in Section 3.1 what papers and how were they combined to obtain the features, then go to detailing what dataset we gathered and tested the authoritative algorithm on in Section 3.2. In the final part, in Section 3.3, we describe how we compute both the metrics and the features with mathematical formulas.

Section 4 is more oriented on offering algorithms on how metrics and features from Section 3 are computed. We start by detailing how we fetched Twitter data and what are challenges involved when using the Twitter Search API in Section 4.1. We then show in Section 4.2 how a tweet looks like and what are the most important fields and what they represent, so that one can understand how metrics from Section 3.3 were computed. Next in Section 4.3 we give a high level code implementation of how the authorities are computed.

After detailing on the features and how the features are implemented in the previous sections, we show in Section 5 what results we obtained and what experiments we did.

The last section, Section 6, concludes by briefly presenting the contributions of the thesis, as well as expressing what future work needs to be done in order to improve the results even further.

Chapter 2

Related Work

In order to classify a web page or a person as authoritative or not, there are two main approaches used extensively in the research community: link analysis based and text analysis based. The former is mainly an application of PageRank[4] or HITS[1][9]. The latter uses textual features combined in such a way so as to give the best results for the problem at hand.

There are some notable papers that have very important contributions to the field, bring new ideas and features to be able to find Influencers on Twitter, in the microblogging and outside the microblogging arena.
The first one, TwitterRank, by Jianshu et. al. [13], introduces a method of identifying Twitter Influencers, one that outperforms Twitter's own ranking model.
The second one, also focusing on Twitter, proposed by E. Bakshy et. al.[1] identifies the "local influence" of users by checking number of reposts (using URL matching) by 1-hop followers, and following the reposts chain to more than 1-hop for "total influence" per user.
One final Twitter related contribution is a model proposed by A. Pal and S. Counts[10], where they emphasize the importance of "both nodal and topical metrics".

Outside microblogging we note some other relevant contributions. One proposed by Farahat et. al., Augeans[5] is applied on web pages, the authors viewing it as "complementary to the Google search engine".
Q&A communities have as well been extensively studied, as they play an important role in communities like: StackOverflow, Duolingo, Wikipedia, Quora,

[1] http://en.wikipedia.org/wiki/HITS_algorithm

Yahoo! Answers, Ubuntu Forums, Stack Exchange, Linux Forums etc. Jurczyk et. al.[6] proposed a Q&A model that uses link analysis and identifies users as hubs and authorities, in accordance with the HITS[9] model.
The model proposed by Pal et. al.[11] deals with early identification of experts in Q&A in a supervised learning way with human feedback, proving the efficiency of their model.

2.1 TwitterRank: Finding Topic-sensitive Influential Twitterers

TwitterRank is an algorithm that uses both topical analysis and link structure (based on PageRank[4]) to identify authorities on Twitter. According to Jianshu et. al.[13], the ranking model proposed by them "outperforms the one Twitter currently [in 2010] uses and other related algorithms, including the original PageRank and Topic-sensitive PageRank".

Their model does gather a set of authors located in Singapore and their 1-hop neighbours (friends and followers). For each author a topical distillation is realized using an unsupervised learning algorithm for obtaining a distribution of probabilities over a set of topics for each user, Latent Dirichlet Allocation (LDA) model[3]. It then constructs an associated graph of directed edges from *followers* to *friends* per each topic. The intuition here is that a user's influence is high if the sum of all the followers' influence is high too. But different from PageRank[4], the "influence on each follower is determined by the relative amount of content the follower received" (Jianshu et. al.[13]) from the author.

The approach imagines a random surfer (like PageRank[4]) going from twitterer to twitterer on the following chain. The transition probability from follower s_i to friend s_j is:

$$P_t(i,j) = \frac{|T_j|}{\sum\limits_{a\,:\,s_i \ follows \ s_a} |T_a|} * sim_t(i,j)$$

where $|T_j|$ is the number of tweets published by s_j, and $sim_t(i,j)$ is the similarity between authors s_i and s_j defined as:

$$sim_t(i,j) = 1 - |DT'_{it} - DT'_{jt}|$$

where DT_{ij} expresses how many times a word in author s_i's tweets is classified as part of topic t_j.

In addition, in a similar way to PageRank[4], they introduce a teleport factor to avoid infinite loops in the network of friends who follow their followers too.

Finally, Jianshu et. al.[13] state that their paper is "the first to report the phenomenon of homophily[2] in a community of Twitter". As a direct consequence of this, they introduced a PageRank-like algorithm that takes the following relationships into consideration.

2.2 AuGEAS (AUthoritativeness Grading, Estimation, and Sorting)

The AuGEAS[5] model proposed by Farahat et. al. uses both link analysis (PageRank[4]) and textual characteristics to retrieve web pages that are authoritative on a subject. According the the authors, "the method is suited to 'high-value' content search". Moreover, the topics given as examples in relation to the proposed model are more scientific-oriented and medical-related, for e.g. "Alcohol Addiction", "Internet Filtering", "Cancer Cure".

The AuGEAS model proposes four attributes: Review , Author's background, Audience, Author's affiliation and some associated values like Reviewed, Not Reviewed, Professional, General, Media, Commercial and None. All four attributes are assigned to web pages and based on the resulting combination of values, it is classified as belonging to one of the following classes:

- **Scientific documents**: written by professionals for other professionals
- **General information-scientific**: written by professionals, but for general public audience
- **Commercial pages**: also including newspaper editorials
- **Mail groups and discussion lists**
- **Home pages**

[2]In the context of Twitter, this means a user follows a friend because he is interested in the content of the friend.

- **Links pages**: contains links with descriptions associated to them

The model also proposes some textual features that help classify according to above mentioned classes: particular characters usage (question marks, semicolons), numbers, particular words ("I", "Mr.", "Dr." etc.), HTML features, readability indices.

Farahat et. al. proposed two prediction models. First, Augeas, is a linear regression supervised learning problem applied on manually labeled training set from which they also derived the best combination of features. The second one, Boost, is a "C4.5" decision-tree based model that classifies according to previous classes, again training on the manually labeled data set.

2.3 Discovering Authorities in Question Answer Communities by Using Link Analysis

The model proposed by Jurczyk et. al.[6] attempts to identify authorities in Q&A communities like Naver and Yahoo! Answers. The model proposed uses link analysis over the network of users. The graph resulted expresses the idea that a user $u1$ connected to user $u2$ with a direct edge means that user $u2$ has answered a particular questioned asked by $u1$.

The above graph was identified by Jurczyk et. al. to suggest "that nodes representing questions authors act as 'hubs' while nodes representing answer authors correspond to 'authorities.'. Using the HITS[9] model, two formulas resulted to compute the authority of the users:

$$H(i) = \sum_{j=0..K} A(j) \qquad A(j) = \sum_{i=0..M} H(i)$$

where "vectors H and A are initialized to all 0 and 1 respectively, and are updated iteratively using the equation above".

2.4 Early Detection of Potential Experts in Question Answering Communities

The motivation of the paper proposed by Pal et. al.[11] is that some potential experts not identified early on don't get appreciation (through badges, statuses, ranks etc.) from the community for their hard work on answering questions and thus leave the community after a while. The model is evaluated on TurboTax Live Community (TTLC)[3], a tax-related Q&A forum on which a team of experts from Intuit promote users to *superuser* status if they consider them skilled enough.

The model identifies *motivation* and *ability* as the most important qualities of a potential expert. In detail, the qualities per user are: quantity of contributions, frequency of contributions, commitment towards the community, domain knowledge of the user, trustworthiness of user's answers, politeness and clarity in response.

On the features above the authors use SVM and DTree supervised learning models. They partition the input data using k-fold cross-validation, k=10, so as to avoid overfitting the training data. They calculated precision p, recall r and F-Measure as $\frac{2*p*r}{p+r}$ where they found the "F-measure of DTree is marginally better than SVM. [...] This makes the predictions of DTree as an attractive choice for further analysis.".

Another model approach was to use a ranking of users based on features, selecting a top of users, by calculating the Gaussian Cumulative Distribution Function (CDF) for each user:

$$R_G(x_i) = \prod_{f=1}^{d} \int_{-\infty}^{x_i^f} N(x; \mu_f, \sigma_f)$$

This ranking model was used both individually and on the resulting users from DTree and SVM for sorting purposes.

[3] http://ttlc.intuit.com

2.5 Everyone's an Influencer: Quantifying Influence on Twitter

The model proposed by E. Bakshy et. al.[1] to "quantify the influence of a given post by the number of users who subsequently repost the URL". Using this main idea, the authors "computed individual-level influence as the logarithm of the average size of all cascades for which that user was a seed". To test how well their model works, they also created a regression tree classifier which used some additional features: number of followers, number of friends, number of tweets, date of joining, total influence and local influence.

The local influence refers to the average number of reposts by 1-hop followers of a user, while total accounts for the complete chain of reposting, not limiting itself to 1-hop followers.

Using the classifier they observed that "individuals who have been influential in the past and who have many followers are indeed more likely to be influential in the future; however, this intuition is correct only on average".

The model was tested on twitter data of two months period from which they kept only bit.ly URLs containing posts. They used one month data for training and one for predicting.

Bakshy et. al.[1] noted that the spread of the posts ("cascade size") also depends on the type of URLs contained in the posts. Their assumption was validated through human feedback using Amazon's Mechanical Turk (AMT) that labeled the content's interestingness.

2.6 Identifying Topical Authorities in Microblogs

The model proposed by A. Pal and S. Counts[10] deals with both textual topical and nodal features of a potential influencer.
The work of the authors is a great inspiration as well for the current thesis, as it poses a very useful property, that the model is "computationally feasible in near real-time scenarios", thus we will reuse some of the features from [10], and credit the authors' work fully.

The authors categorize tweets into the following categories: original tweets, conversational tweets and repeated/retweeted tweets, abbreviated OT, CT and

RT respectively.

The features extracted for each potential influencer are:

- topical signal (TS), expressing the interest of the user to a given topic
- signal strength (SS), expressing how much does an author post his original tweets versus other's posts by retweeting
- non-chat signal (nCS), expressing how much does a user chat about the topic, not being important when chat was initiated by other user (as the reply may be out of politeness)
- retweet impact (RI), expressing how many times did others retweet author's post, greatly penalizing posts that are retweeted by a single (or few) author(s) all the time and also discarding retweets of the author (trying to discard retweet reciprocity, a concept highlighted in TwitterRank[13] about "following" relationship)
- mention impact (MI), expressing how many times was an author genuinely mentioned, discarding the time an author mentioned others (again, to avoid mention reciprocity, concept used in TwitterRank[13])
- information diffusion (ID), expressing how many users saw the information from the user, penalizing by the number of users who knew about this particular information before the author
- network score (NS), expressing the range of followers that are topically interested in following the user, discarding the possibility of following reciprocity mentioned in TwitterRank[13]
- number of links shared divided by author's original topical tweets
- self-similarity score (explained below) to avoid including spammers that repeat posts
- number of hashtags used divided by author's original topical tweets

Using the above features, the authors use Gaussian Mixture Model (GMM) to form Gaussian clusters over the users represented as n_features dimensional.

Moreover, to minimize the errors based on initial estimates, the initial parameters required are computed using KMeans.

The authors run a few instances of GMM and pick the one "with largest log likelihood". Given the probability for each point of being in a resulted cluster, the points that have a probability $p(x) \geq 0.9$ are kept. Then the cluster with the best (average) TS, RI, MI is chosen.

They then apply Gaussian ranking by assuming the features are Gaussian, and calculate the total score of a user by its features with:

$$R_G(x_i) = \prod_{f=1}^{d} \int_{-\infty}^{x_i^f} N(x; \mu_f, \sigma_f)$$

in a similar way Pal et. al.[11] did.

The results are compared to a PageRank[4] based implementation, a strictly textual properties model (using only TS, SS, nCS, and the last three features mentioned) and a randomly-selected users that aren't in the target cluster computed with GMM as explained above.

The results were decreasing in the order they were specified above, with the best one being the one using all features presented. The interestingness score was asserted using a set of human evaluation participants.

As a final note, the model proposed by A. Pal and S. Counts[10] does define a new similarity score, easy to compute in comparison with the one proposed by Weng et. al. in TwitterRank[13]. The similarity between two posts $s1$ and $s2$ is defined as:

$$S(s1, s2) = \frac{|s1 \cap s2|}{|s1|}$$

expressing "how much a user borrows words from her previous posts". So applying this as a sum over all combination of ordered posts by an author results in a similarity score computed for each user.

Chapter 3

User Metrics and Dataset

In this chapter we will explain in order:

1. How we combined ideas from other papers and what the overall idea of the model is.
2. What data we tested on and some characteristics about it.
3. The metrics and features used to get authorities on Twitter

3.1 Overview

Our model's goal from the start was to keep it "computationally feasible in near real-time scenarios", as Pal et. al. did in their model[10]. But along the way we saw that the idea is kind of exaggerated, because in real implementations used by clients (e.g. marketing), in order to infer something on a topic, you first need to gather the data on that specific topic (which may take a while due to Twitter's API rate limiting, see section 3.2).

Moreover, some metrics from Pal et. al.[10] cannot be inferred easily by just looking on topical data, but additional API calls to Twitter need to be done (which may take a while due to rate limiting). Because of that, we dropped the idea of **"near real-time"** and consider the algorithm as **"simple to explain to potential marketing clients"**.

The model proposed in the following thesis is not our contribution to the field, but rather put together, in a way it hasn't been done before, from contributions presented in the previous section, especially from Pal et. al.[10] and from the

work of Bakshy et. al.[1]. We even try to see how applying some weights to features influences the final result, a wish expressed in the future work of Pal et. al. model.

From the former paper of Pal et. al.[10] we used all metrics except graph based metrics which would have been hard to detect without access to the entire dataset (as Pal et. al. might have had). See section 4.1.1 for details on why gathering all the data is difficult and costly.

From the latter paper of Bakshy et. al.[1] we came with the idea of tracing the number of unique URLs users posted, related to the searched topic. This is an original contribution that resembles in a way the approach from Bakshy et. al.[1], but it distinguishes from it in that we don't go deep into tracing how the influence spreads throughout the network, but just compute a simple textual metric.

Our model contains some features extracted from tweets that we gathered (see section 3.2). Based on those metrics we form a set of features inspired from Pal et. al.[10] and apply a simple non-weighted mean on all the features, forming a TOP users rank. We also scaled the features as some have higher values than others, so avoiding data skews.

3.2 Dataset

We chose the dataset from table 3.1 due to both some advantages and some restrictions:

- **sports** and **world concerns politics/war**, are among the best talked about in general
- **sports** include a lot of small-talk and spammers and bots which we wanted to see if we can exclude with the features
- the particular topics are best talked about in June-July 2014, and the Twitter Search API restrictions (see section 4.1.1) didn't allow us to get other past topics.
- **"halep"** is chosen for two reasons: few results (and thus quickly computable features) and national pride

- **"ukraine gas russia"** was chosen because it includes some subtile influencers (like reporters) which we wanted to see if we can find

Query	"halep"	"ukraine gas russia"
# Tweets	2,736	16,203
# Authors	1,507	10,967
Time frame	5 x 24 h (24-29.06.2014)	14 h (15-16.06.2014)

Table 3.1: Dataset gathered and used for results.

3.3 Metrics and Features

3.3.1 Metrics

In this chapter we only present the metrics and features we use for finding topical authorities. We first use some metrics defined by Pal et. al.[10], presented in table 3.2. The last metric (**U**) is inspired from Bakshy et. al., although it distinguishes from their work [1]. See section 3.1 for an overview and more details on what papers we used and how it is different.

Like in Pal et. al.[10], the tweets are classified in the following categories:

- conversational tweet (CT) is a tweet that begins with "@username", so as to express that the conversation is with the "username" user; this way the author notifies the "username" user about the tweet; the two users can keep a conversation this way, through public tweets (unlike Facebook and others which can have private conversations)

- repeated tweet (RT) is a tweet that begins with "RT @username", expressing the fact that the content is not original, but has been copied from "username" and one gives credit to that user using the **@ mention** sign (used in CT too)

- original tweet (OT), one that a user has posted without him retweeting or copying content from others (RT) and which is neither a conversational tweet (CT)

OT1	Number of original tweets
OT2	Number of links shared
OT3	Self-similarity score that computes how similar is author's recent tweet w.r.t her previous tweets
OT4	Number of keyword hashtags used
CT1	Number of conversational tweets
CT2	Number of conversational tweets where conversation is initiated by the author
RT1	Number of retweets of other's tweet
RT2	Number of unique tweets (OT1) retweeted by other users
RT3	Number of unique users who retweeted author's tweets
M1	Number of mentions of other users by the author
M2	Number of unique users mentioned by the author
M3	Number of mentions by others of the author
M4	Number of unique users mentioning the author
U	Number of URLs the user posted before any other

Table 3.2: User Metrics used for feature computation.

- mentions (M) distinguish themselves from CT and RT, as we count a "@username" mention only if it's not at the beginning of the tweet and it doesn't begin with "RT @username", because these two are automatically added headers by Twitter. In short, we only count as mentions those "@username" who've been mentioned intentionally, not automatically added by Twitter.

The self-similarity score OT3 between two texts $s1$ and $s2$ is computed like in Pal et. al.[10]:

$$S(s1, s2) = \frac{|s1 \cap s2|}{|s1|}$$

expressing "how much a user borrows words from her previous posts". To apply this for each user, we use:

$$S(a) = \frac{2 * \sum_{i=1}^{n} \sum_{j=1}^{i-1} S(s_i, s_j)}{(n-1) * n}$$

where $time(s_i) < time(s_j), \forall\ i < j$.

Computing and understanding the metrics from table 3.2 is straightforward, and it shall become clear how each one is computed in Section 4, especially by going through the tweet's details in Section 4.2.2.

3.3.2 Features

The features are computed from the metrics from table 3.2 and have been designed in such a way so that the biggest value for any feature is the best. This implies a simple idea that trying to maximize the average of all the features would give a potential authority.

For metrics that are best to be lower than higher (like similarity), we can solve that with sign change or subtracting it from one (as we did).

See each feature in detail below:

Topical Signal

Topical Signal estimates the degree of an author's involvement in the topic:

$$TS = \frac{OT1 + CT1 + RT1}{|\#tweets|}$$

This may be highly useful to use when we want to find those true authorities that post about a specific topic that is broad enough. If however, the topic is rather specific, like "ukraine gas russia" from table 3.1, then we won't be able to find any authorities that post exclusively about that, but rather there are authorieties that post about a wider topic, like "ukraine russia", a good example being *KyivPost*.

Signal Strength

Signal Strength estimates how original is the author. For a true authority, this should be close to one because we consider an authority one that finds the news early, not one that finds information from other twitter sources.

$$SS = \frac{OT1}{OT1 + RT1}$$

Non-Chat Signal

$$nCS = \frac{OT1}{OT1 + CT1} + \lambda * \frac{CT1 - CT2}{CT1 + 1}$$

The above formula suggests that we want to take into consideration the chat that a user has produced. But if the user did not start the conversations itself, then the second term ($\frac{CT1-CT2}{CT1+1}$) will be close to 1, so the result will be the biggest when the author hasn't started chats itself; the λ parameter controls the importance of the second term.

Retweet Impact

Retweet Impact estimates how retweeted were an author's tweets. It catches the case when only a few users are often retweeting and it penalizes it.

$$RI = RT2 * \log RT3$$

From the above formula we can observe that if a user's posts are retweeted 100 times ($RT2 = 100$), but by a single user ($RT3 = 1$), then $RI = 0$.

Mention Impact

Mention Impact estimates how mentioned is a user by others. By mentioned we mean those "@username" mentions that were not added by twitter, but explicitly by the user (as said at the beginning of section 3.3).

The feature's formula discards the mentions that may be due to reciprocity, concept found in Weng et. al.'s work, TwitterRank[13], about the "following" relationship, but it can be thought as politeness that applies to mentioning too.

$$MI = \max(0, M3 * \log M4 - M1 * \log M2)$$

Similar to the retweeting impact from above, if the number of users mentioning the author is small (M4), then the overall score is also small.

Hashtag Ratio

Hashtag Ratio accounts for the average number of hashtags per post. It should keep close to 1 and not diverge too much.

$$HR = \frac{OT4}{OT1}$$

It helps avoid spammers/bots with very big ratio, as this seems to be their common practice.

Links Ratio

Links Ratio accounts for the average number of links per post. It should not go high above 2 and diverge too much, just like for hashtags. It doesn't really make sense to have more than two URLs per post (considering one is an image, one is an article at most).

$$LR = \frac{OT2}{OT1}$$

It is not clear whether spammers/bots use more than 1-2 URLs per post. At a first glance it seemed not to be the case. So the *LR* feature may help in demoting those that don't post any URLs in their posts, as that doesn't offer any credibility. Moreover, you would expect that a tweet with 140 characters[1] on average would not be enough to report everything in detail that happened, but rather point to an external resource where to go into details. This greatly depends on topic.

Non-Similarity

Non-Similarity close to 0 means that an author's posts are highly similar to one another. The probability of a user being a spammer/bot raises with the similairity of the posts. Similarity metrics is between 0 and 1 and is better to be closest to 0. So because we want to maximize the features, we invert it to be better for *nSIM* to be close to 1.

$$nSIM = 1 - SIM$$

This can catch only easy bots, not the most sophisticated ones.

URL

This feature is inspired by the work of Bakshy et. al. in [1]. The idea is to count the number of URLs a user has posted before any other user. The URL feature

[1] https://dev.twitter.com/docs/counting-characters

highlights how many original contributions does a user have with respect to the searched topic.

This feature can cover some cases where a user would not give credit to another one's original post by using the "@" mention sign or the "RT @" feature, but instead would copy the content without giving credit.

The URL feature expresses how many URLs did an author post on topic before any other user. So given a news article, the credit is received by the author who posts a link to the news article first. The others receive nothing, moreover, they get penalized.

Before using the feature we first considered what's the probability of a URL to be included in Twitter posts. Did this for all three queries, results in table 3.3.

Query	"halep"	"ukraine gas russia"
# Tweets	2,736	16,203
# URL Tweets	1,312	13,775
%	48%	85%

Table 3.3: Percent of URLs appearing in tweets.

Indeed, adding URLs in tweets is also supported by the fact that Twitter imposes a 140 character limit[1], so one generally uses Twitter by briefly describing an idea and then adding a URL to an external resource where it's explained more thoroughly.

We can see in table 3.3 that in topics where we want to support our claim (like politics), a URL is very frequent, while in sports URLs are less likely to appear due to the nature of the tweets that are mainly status updates that don't need to offer any additional details or references.

Chapter 4

Implementation Overview

The infrastructure I used is an amazon EC2 *t1.micro* instance, with low CPU capacity but a volume of 20 GiB[1]. The advantage of using Amazon EC2 over a usual PC is that the former can fetch and process data non-stop. I found very useful a Linux command called `screen`, which allows closing the SSH connection to EC2 and returning later to find the background process running and to access it once again before exiting. It is an improved version of `nohup`.

Using Python, I mainly turned my attention towards the following packages:

scikit-learn A machine-learning open source library from which I used scaling, clustering, dimension reduction

nltk A natural language toolkit from which I used tokenization, stemming

numpy,scipy math general purpose used with sklearn

tweepy Used for making Twitter API calls

pymongo A mongoDB[2] driver used for queries to the database

The entire `code` is Open Source and available `on Github`[3]. It was written in Python as it is easier to start development and less verbose than Java.

The application is structured in modules, so it can be easier to read/write and less prone to errors:

1. A fetching module that receives a query and fetches data from Twitter

[1] 1 GB = 10^9 bytes, 1 GiB = 2^{30} bytes.

[2] A NoSQL document based database, very friendly, efficient, open source

[3] https://github.com/andreip/tweeter-authorities/

2. A storing database where tweets received from Twitter API go after fetch

3. A computing module that receives a query for which to compute authorities (same with the one the data was fetched from Twitter).

4.1 Fetching data from Twitter

4.1.1 Challenges

The tweets in the dataset were gathered by using the Twitter API. It was difficult to gather more than one hundred thousand per topic due to some challenges we did not expect at first:

1. rate limits imposed by Twitter Search API[4]

2. not being able to get older than a week data using Twitter Search API[5]

The above restrictions that we found along the way required a very well designed infrastructure to be able to do this in an efficient way. Needless to say, we did not have that kind of infrastructure to parallelize the computation (which is as well very hard to setup and maintain in a cost-efficient way).

To get a sense of how enormouse amount of data Twitter has been generating, please note that according to Alexa ranking in July 2014[6], Twitter.com is the 9th most visited site in the world, the 8th most visited in the United States and the 11th most visited in India.

We found two approaches that would solve the above problems, but which were hard to solve correctly or very expensive:

- Weng et. al. had a different approach of gathering tweets in TwitterRank[13]. They parsed HTML Twitter web pages in order to avoid being rate limited. This, however, surely imposed a lot of work and was prone to errors and exceptions, although not mentioned in their paper. It would have been very appreciated, had Weng et. al. shared their work in the Open Source community.

[4]https://dev.twitter.com/docs/rate-limiting/1.1
[5]https://dev.twitter.com/docs/using-search
[6]http://www.alexa.com/siteinfo/twitter.com

- One other (expensive) approach suggested by Twitter is to use the Twitter Firehose API. The difference to the Search API that we used is that the Firehose API is **guaranteed to deliver all the tweets** that you searched for. The Twitter Firehose is offered by two Twitter providers, GNIP and DataSift, that you have to contact in order to get access.

- Another approach to using the expansive Firehose API is to remember the data yourself, by using the Twitter Streaming API that pushes tweets once they get posted on Twitter. High volume of tweets are received, so the infrastructure needed is very complex and hard to manage without a team of experienced programmers and machines that scale according to needs.

4.1.2 Method

We ended up using Tweepy, a python Open Source library that helps in making HTTP requests to Twitter's API. In order to avoid rate limiting, there are at least a couple of options:

- As we did, wait between calls to Twitter endpoint.

- Or use multiple applications and do a **round-robin** between them, as rate limiting is per application.

The second option is better as you can get more data faster without having to wait between calls, but it requires to create additional applications and make a *scheduling algorithm* between them. This is also risky because Twitter detects fraudulent multiple applications and bans/deletes them.

The first one is the simplest one and involves a code as the following. Suppose you make an API call to the search endpoint which allows 180 queries per 15 minutes[7]. This means we are allowed one query every five seconds, like in Algorithm 1.

Taking the Algorithm 1 by lines, we detail a little further:

line 7 *api.search* is a Tweepy[??] method that makes an API call to endpoint `https://api.twitter.com/1.1/search/tweets.json` [8]

[7] https://dev.twitter.com/docs/rate-limiting/1.1

[8] https://dev.twitter.com/docs/api/1.1/get/search/tweets

Algorithm 1 Fetching Twitter data

```
1:  # query: The query to fetch.
2:  # pages: One page may contain 100 tweets.
3:  procedure FETCH_TWEETS(query, pages)
4:      page_count ← 0
5:      while page_count < pages do
6:          page_count ← page_count + 1
7:          tweets ← api.search(query)
8:          # ... process tweets
9:          # Wait 5 seconds before the next fetch to avoid being rate limited.
10:         sleep(5)
11:     end while
12: end procedure
```

line 8 Processing the tweets may mean a couple of things:

- saving them in the database
- modifying the tweets before or after inserting them in the database, for e.g. expanding the `entities.expanded_url` (see section 4.2.2 for more on tweet format) to the maximum before inserting in db.

line 10 Current operating system thread waits for 5 seconds ; this is the easiest hack one can use to avoid complicated mechanisms of avoiding rate limits.

4.2 Storing tweets

4.2.1 About MongoDB

We use Open Source mongoDB[2] as a storage database, and pymongo as its driver for Python programming.

We've computed some stats in table 4.1 in order to show the compression efficiency of mongoDB (only about 1GB of storage). More info about the significance of the table columns can be viewed at MongoDB's official documentation[9].

[9] http://docs.mongodb.org/manual/reference/command/dbStats/

Collections	13
Objects	135,822
avgObjSize (bytes)	6,061.386
storageSize (bytes)	1,032,364,032
indexes	453,7680

Table 4.1: User MongoDB stats.

Using on a Linux machine is very straight-forward, it needs only to know a *"–dbpath"*, the path where the database files are going to be stored.

The documents that mongoDB expects to receive are dictionaries. Fortunately, tweets received from Twitter are JSON-like and we can easily convert them to Python dictionaries and insert them in the database. A tweet with only the essential fields kept can be seen in Listing 4.1. Once it is inserted in mongoDB, it automatically receives an additional field as in Listing 4.2.

Listing 4.1: Tweet Reduced

```
{
        'created_at': 'Tue Jun 24 18:39:00 +0000 2014',
        'entities': {
                'hashtags': [],
                'symbols': [],
                'urls': [],
                'user_mentions': []
        },
        'favorite_count': 2,
        'id': 481506793332305920,
        'retweet_count': 1,
        'text': '#Halep may be a late bloomer but she is
          the future! She is one of my new favorites.
          Beautiful playing. #Wimbledon14 #thehill',
        'user': {
                'id': 33069310,
                'name': 'Victoria Trower
                'favourites_count': 250,
                'friends_count': 388,
```

```
                'followers_count': 111,
                'location': 'Philadelphia, PA',
                'screen_name': 'vitaluv0505',
                'statuses_count': 3110,
                ...
        }
        ...
}
```

Listing 4.2: MongoDB _id

```
{
        "_id" : ObjectId("53b06ef0d6bbb370bf6efa8f"),
        ...
}
```

4.2.2 Tweet format

Using Listing 4.1, we will review some important fields that are used for computing the metrics from Section 3.3:

text The actual content the user tweeted on Twitter.com

entities The text field is parsed by Twitter and categorised accordingly. We can see what URLs the user included in his post by looking up the entities.urls field

favorite_count How many times has the tweet been favorited by others

retweet_count How many times has the tweet been retweeted by others

user The user that posted the tweet and its metadatas, like:

screen_name Represents the "@username" of the user who tweeted the tweet; can see a user's profile at `http://twitter.com/screen_name`

favourites_count Number of tweets the author has marked as favorite

friends_count Number of users the user follows

followers_count Number of users that follow the user

statuses_count Total number of tweets posted by user

id This one is very important as it identifies the tweet itself. You can also view the tweet at `http://tweeter.com/screen_name/status/id`

4.3 Processing data to find Authorities

The overall steps to generating an authority are the ones in Algorithm 2. This can happen only after we've gone through Algorithm 1 that fetches tweets for a given query and stores them in the database.

Going by the lines, we shall go into more details about the Algorithm 2:

line 5 The usernames that we compute features and metrics for in the current implementation are those that **have an above average number of topical posts**. E.g. if we have three users `a, b, c` in total, where they have `a=2, b=3, c=4` posted posts about a query; average of posts is 3, so we only compute metrics and features for user `c with 4 posts` .

lines 9-10 use both mongoDB and the Twitter API to gather needed information

line 14 Scaling features is important because some values may be bigger than others. We use `sklearn.preprocessing.MinMaxScaler` [10] that transforms all values to be in `[0,1]`

line 16 the number of top users is currently not fixed e.g. 10, but varies with the number of total users we calculate metrics and features.

[10] http://scikit-learn.org/stable/modules/preprocessing.html#scaling-features-to-a-range

Algorithm 2 Computing Topic Authorities

```
1:  # query: The query to get tweets for, from db (already fetched).
2:  procedure FIND_AUTHORITIES(query)
3:      # Get a list of users that are eligible to have
4:      # metrics and features calculated for them.
5:      usernames ← get_usernames()
6:
7:      all_features ← ∅
8:      for username ∈ usernames do
9:          metrics ← compute_metrics(username)
10:         features ← compute_features(metrics)
11:         all_features ← all_features ∪ features
12:     end for
13:
14:     final_features ← scale_features(all_features)
15:     # top_no: the number of final users to select as authorities; e.g. 10
16:     authorities_users ← get_top_users(usernames, final_features, top_no)
17:
18:     return authorities_users
19: end procedure
```

Chapter 5

Experiments and Results

This chapter is dedicated to detailing more on the final step from line 16 in Algorithm 2 (page 27). This step of using the resulted features in order to obtain a top of authority users can be done in multiple ways:

1. Using a non-weighted mean over the scaled features
2. Using a weighted mean over the scaled features
3. Using some clustering algorithms and decide:
    - what to do with the clusters
    - which cluster(s) to pick as final
    - which users to pick from the final cluster(s)

Taking the above steps, we propose followup sections for each, treat them and show results obtained.

5.1 Non-Weighted Features Mean

Using a `mean(features)` approach, we got the following results from table 5.1. What **good users** like WTA, Wimbledon etc. have in common is that they are well above the median with the features:

- Mention Impact: WTA - 12.54, Wimbledon - 364.5
- Retweet Impact: WTA - 26, Wimbledon - 94.18

So notice that these two are incredibly popular among tennis lovers and these metrics are highly useful for promoting these two. Spammers have absolute **0** on these ones, the lowest possible value.

We detected the spams/bots from table 5.1 using an external app **BotOrNot**[1] developed at the Indiana University. I asked whether they have an API available and responded that it's still in development and testing. This tool[1] is indeed useful for removing spammers from the list of top users and will help in improving the list significantly when an API will be available.

5.1.1 With URL feature

The following results from table 5.1 are obtained using all features from Section 3.3.2.

Query	"halep"	"ukraine gas russia"
Users	2,736	10,967
Users for which computed features	481	2,266
Percent	18%	20%
# Authorities	13	16
Good Users	Wimbledon, WTA, HalepFans	RT_com, Reuters CNNMoney, KyivPost MaximEristavi, guardian STForeignDesk
Good (%)	23%	44%
Spam/Bot Users	TennisWorld3, yoursportman, Tennisfansclub1,	AdeyeyeFaleye, CollectedN WatchingaBuzz newzlasz
Spam/Bot (%)	23%	25%

Table 5.1: Query results with URL feature.

[1] http://truthy.indiana.edu/botornot/

5.1.2 Without URL feature

The following results from table 5.2 are obtained using all features from Section 3.3.2 except the URL feature.

Query	"halep"	"ukraine gas russia"
Users	2,736	10,967
Users for which computed features	481	2,266
Percent	18%	20%
# Authorities	13	16
Good Users	Wimbledon, WTA, HalepFans	RT_com, Reuters CNNMoney, KyivPost MaximEristavi, guardian STForeignDesk
Good (%)	23%	44%
Spam/Bot Users	TennisWorld3, yoursportman, Tennisfansclub1,	AdeyeyeFaleye, CollectedN WatchingaBuzz newzlasz
Spam/Bot (%)	23%	25%

Table 5.2: Query results without URL feature.

5.1.3 Conclusions

Conclusions per feature to decide on this first experiment:

Mention Impact highly useful in identifying authorities

Retweet Impact highly useful in identifying authorities

Topical Signal is not useful at all, because:

- for users like "AdeyeyeFaleye", the topical signal is 0.25, compared to others which have the Topical Signal $< 10^{-3}$
- this happens because "AdeyeyeFaleye" has few tweets posted, compared to other influential users who have a lot of tweets

Non-Chat Signal, Signal Strength close to one or one for all users (after scaling), but somewhat useful for avoiding users that have low Signal Strength or Non-Chat Signal

URL As we can see, no visible differences in percentages have been obtained by using or stop using the URL feature alone. We cannot thus say anything about its usefulness.

5.2 Weighted Features Mean

5.2.1 "halep" dataset

The table 5.3 represents various cases of applying weights and seeing how the spam/not spam percentages vary. In the first row we have the case where Retweet Impact has weight one, and all other features have weight zero. The users found (noted here for reference) are:

row 1 only Retweet Impact weight one, all others have weights zero
Good Users: Wimbledon, WTA, tikitaka_ro, SuperSportBlitz, bbctennis, BenRothenberg, HalepFanpage, SI_Tennis, Tennis, ESPNTennis, MindTheRacket, HalepMania, adidastennis

row 2 Retweet Impact and Mention Impact both have same weight, all others have weight 0
Good Users: same as row 1, but instead of adidastennis, we got livetennis

row 3 Good Users: 1stTennisNews, AP_Sports, ASBandara

5.2.2 "ukraine gas russia" dataset

The table 5.4 represents various cases of applying weights and seeing how the spam/not spam percentages vary. In the first row we have the case where Retweet Impact has weight one, and all other features have weight zero. The users found (noted here for reference) are:

row 1 only Retweet Impact weight one, all others have weights zero
Good Users: MaximEristavi, RT_com, KyivPost, BBCSteveR, Reuters, mchancecnn,

Weights	Row	Good (%)	Spam (%)
RI=1	1	100%	0%
RI=1,MI=1	2	100%	0%
SS=1,nCS=1	3	23%	30%
SS=1,nCS=1,MI=1	4	46%	15%
SS=1,nCS=1,RI=1	5	100%	0%
SS=1,nCS=1,U=1	6	0%	46%
SS=1,nCS=1,U=1	7	0%	46%

Table 5.3: Query weighted results for "halep" dataset.

EuromaidanPR, itvnews, AJELive, Steiner1776, FT, BlogsofWar, BBCBreaking, guardian, gbazov, RenieriArts

row 2 Retweet Impact and Mention Impact both have same weight, all others have weight 0
Good Users: same as row 1, but instead found USATODAY and MoscowTimes instead of Steiner1776 and BlogsofWar

row 3 Good Users: like row 1, but found STForeignDesk, Jonnyhibberd, henrikholben, PressTV instead of others

Weights	Row	Good (%)	Spam (%)
RI=1	1	100%	0%
RI=1,MI=1	2	100%	0%
MI=1	3	100%	0%

Table 5.4: Query weighted results for "ukraine gas russia" dataset.

Similar to experiments on "halep" dataset from table 5.3, the **Retweet Impact** and **Mention Impact** bring some very interesting and useful results. And by combining other features with RI or MI, we get more noisy results, but maybe some that are not that popular but have good sources and public to influence. It remains a trade-off between exploration and exploitation as in hill climbing problem.

5.2.3 URL Feature

We can see that the **URL** feature brings no influencers, but rather spam users. It may be because the URLs are shortened. It would help if we made a mode advanced URL match:

- expand the shortened URLs at its maximum (it may be shortened a few times)
- compute approximation of equal expanded URLs, as we've encountered some URLs like `http://yoursportman.mobi/football_news_details.php?news_id=....` and these are very similar and easy to infer that it's a new feed you don't want to subscribe to; this user is actually detect as a spammer.

5.2.4 Conclusions

Conclusions per feature to decide on weighted experiment:

Mention Impact highly useful in identifying authorities, it gets the Good percentage upwards if used

Retweet Impact highly useful in identifying authorities, it gets the Good percentage upwards if used

Signal Strength, non-Chat Signal together they are not useful, they can't find many authorities (only 23%).

URL As we can see, URL feature is not bringing any influencers, but rather attracting spammers. See Section 5.2.3 on why the URL feature may have been wrongly computed for users and deserves another another try.

Chapter 6

Conclusions

6.1 Contributions

In a brief manner, we enumerate our main contributions that we will go into detail below:

- combining features from Pal et. al.[10] and a new original feature (URL) but the feature did not turn to actually be useful in getting authoritative users.
- using different weights for features to see how this influences the resulting influencers and got **100% authoritative users** from users returned with equal weights for **Mention Impact, Retweet Impact** and weight zero for the other features.
- a small accepted bug fix to an open source library used, Tweepy[1]

.

The results we have shown in Section 5 have been inspired both from reading papers present in Section 2 and others not present there, helping in offering an overview of the current research results and further experiments in the area of Microblogging and outside Microblogging (like Q&A communities and blogs).

The work in this thesis is mainly based on the contributions brought by Pal. et. al.[10] from where we used all the metrics but the graph based ones (which were hard to compute due to Twitter's API rate limiting and other challenges, see Section 4.1.1 for more details). It should be noted that we had some own interpretations of the work proposed by Pal. et. al.[10]:

[1]https://github.com/tweepy/tweepy/pull/446

- we gathered the **topical signal** related tweets by using `api.search` endpoint, not using a simple `$regex` in the database. This may in a way give better results due to Twitter's API search being more complex

- mentions related metrics `M1, M2, M3, M4` have been computed by database lookup, and not by using the Twitter's API search (which could be inexact, as we didn't use Firehose API, see Section 4.1.1 for details).

- **self-similarity** in our work is done only on tweets in the searched topic (returned by Twitter Search API); it is not clear whether Pal. et. al. in [10] computed the self-similarity metric on all a user's tweets. It may be a good idea to do so, as Weng et. al. did on a user's timeline by using Latent Dirichlet Allocation (LDA) model[3] in their work, TwitterRank[13]. This offered an idea of what topics does a user post about on a daily basis.

- **hashtag ratio**, **links ratio** features are not really the best to try and maximize as we've currently done, because abusing hashtags or links in a Twitter post may rather signal spam/bot behavior. Although not specified, Pal. et. al. may have used some more sophisticated ratios for the two features.

- **non-similarity feature** is computed through our own understanding of the fact that the goal is to maximize the features. It is not explicitly presented in the work of Pal et. al.[10]

- **mention impact** is yet again considered to be at least 0, we don't leave the feature go below 0 to not skew the data interval too much but rather keep it above 0. This is not specified in the work of Pal et. al.[10].

Another feature not present in the work of Pal et. al.[10] is inspired from Bakshy et. al.[1], where the authors do a full trace of how URLs spread and influence the networks of authors to be able to tell the influence it has. We simplify the idea and abstract it into a simple to compute feature, **original contribution**, representing the number of URLs a user posts first (before any other author about that specific topic). For more details see Section 3.3.2.

6.2 Future work

Although our approach is based on simplicity and quickness, it captures important users (authoritative users) for topics in the datasets from Section 3.2.

The best results are obtained from using only features **Mention Impact, Retweet Impact**, with a 100% authoritative users returned for all topics in the dataset from Section 3.2. In order to obtain even better results for broader topics that have not been tested, an external API **BotOrNot**[2] can help filter out bot/spam users.
Currently no API exists, as the project is still in testing and development phase. If not for exclusion purposes, at least flagging those potential spam users in the final authorities results would be useful.

Another future work would be to gather more data sets from more general topics (like `IT, TV`) and see how results appear there. This might result in discovering new features that could increase the chances of finding authorities over all topics.

[2]http://truthy.indiana.edu/botornot/

Bibliography

[1] E. Bakshy, J. M. Hofman, W. A. Mason, D. J. Watts. Everyone's an Influencer: Quantifying Influence on Twitter. In WSDM, pages 65-74, 2011.

[2] F. M. Bass. A new product growth for model consumer durables. Management Science, 15(5):215–227, 1969.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. Journal of Machine Learning Research, 3:993–1022, 2003.

[4] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. Computer Network and ISDN Systems, 30(1-7):107–117, 1998.

[5] A. Farahat, G. Nunberg, and F. Chen. Augeas: authoritativeness grading, estimation, and sorting. In CIKM, pages 194-202, 2002.

[6] P. Jurczyk and E. Agichtein. Discovering authorities in question answer communities by using link analysis. In CIKM, pages 919-922, 2007.

[7] E. Katz and P. F. Lazarsfeld. Personal influence; the part played by people in the flow of mass communications. Free Press, Glencoe, Ill." 1955.

[8] E. Keller and J. Berry. The Influentials: One American in Ten Tells the Other Nine How to Vote, Where to Eat, and What to Buy. Free Press, New York, NY, 2003.

[9] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. In SIAM symposium on Discrete algorithms (SODA), pages 668-677, 1998.

[10] A. Pal, S. Counts. Finding topical authorities in microblogs. In WSDM, pages 45-54, 2011.

[11] A. Pal, R. Farzan, J. A. Konstan, R. E. Kraut. Early Detection of Potential Experts in Question Answering Communities. In UMAP, pages 231-242, 2011.

[12] E. M. Rogers. Diffusion of innovations. Free Press, New York, 4th edition, 1995.

[13] J. Weng, E.-P. Lim, J. Jiang, and Q. He. TwitterRank: finding topic-sensitive influential twitterers. In WSDM, pages 261-270. 2010.