TwitterEcho - A Distributed Focused Crawler to Support Open Research with Twitter Data

Matko Bošnjak
DEI-FEUP, University of Porto
Rua Dr. Roberto Frias, s/n
4200-465 Porto, Portugal
mbosnjak@fe.up.pt

Eduardo Oliveira
DEI-FEUP, University of Porto
Rua Dr. Roberto Frias, s/n
4200-465 Porto, Portugal
ei06023@fe.up.pt

José Martins
DEI-FEUP, University of Porto
Rua Dr. Roberto Frias, s/n
4200-465 Porto, Portugal
josemartins88@gmail.com

Eduarda Mendes Rodrigues
DEI-FEUP, University of Porto
Rua Dr. Roberto Frias, s/n
4200-465 Porto, Portugal
eduardamr@acm.org

Luís Sarmento
Sapo.pt - Portugal Telecom
Av. Fontes Pereira de Melo 40,
5th floor
1069-300 Lisbon, Portugal
las@co.sapo.pt

ABSTRACT

Modern social network analysis relies on vast quantities of data to infer new knowledge about human relations and communication. In this paper we describe TwitterEcho, an open source Twitter crawler for supporting this kind of research, which is characterized by a modular distributed architecture. Our crawler enables researchers to continuously collect data from particular user communities, while respecting Twitter's imposed limits. We present the core modules of the crawling server, some of which were specifically designed to focus the crawl on the Portuguese Twittosphere. Additional modules can be easily implemented, thus changing the focus to a different community. Our evaluation of the system shows high crawling performance and coverage.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information filtering; H.3.4 [**Systems and Software**]: Distributed systems; D.2.13 [**Reusable Software**]: Domain engineering

General Terms

Design, Performance, Documentation, Experimentation

Keywords

Twitter, micro-blogging, focused crawling, open source, distributed system, social network analysis

1. INTRODUCTION

The social media services that emerged over the past decade changed the way we communicate. They have, thus, become an attractive object of research in several fields, including, for instance, information extraction and social network analysis. Obtaining data from these social media services is usually a complex problem, since these services usually do not provide direct access to all the data being produced, even for research purposes. Therefore, researchers need to build systems for obtaining such data, either leveraging existing APIs exposed by the social media services or, alternatively through web crawling [14].

The Twitter[1] micro-blogging service, in particular, has prompted numerous studies, ranging from analysis of users' interactions [12, 16], to analysis of hashtag usage [8] and URL citations [1], and to specific content analysis [2, 6]. Many of these studies relied on crawling data independently, but there has been an effort to make the data available to the wider scientific community in order to promote further research. However, a change in Twitter's terms of service prohibits publicly sharing crawled data for any purposes, including research. A window of opportunity stays open though: it is possible to provide the community IDs of the users and tweets crawled, and let other researchers crawl Twitter by themselves using the available API. This approach has been used in the recent TREC 2011 Microblog Track[2]. However, there is still an additional restriction imposed by Twitter: access to API's is limited to a relatively reduced number of queries (details on the API limits are provided in Appendix). The alternative for obtaining unlimited access to data consists in acquiring a license to the "Gnip The Social Media API"[3], but such access is costly and, therefore, inaccessible to most academic researchers.

Our approach to solving this data access problem was to implement a Twitter crawler and making it available open-source to the community[4]. This allows other researchers to perform their own data collection. In this paper, we present the crawler which allows retrieval of Twitter data from a focused community of interest. The modular nature of our crawler allows focusing it on different segments of Twitter data, namely different communities of users described by geographic, demographic, linguistic or even topical characteristics.

The remainder of this paper is organized as follows. Sec-

WWW 2012 Companion, April 16–20, 2012, Lyon, France.
ACM 978-1-4503-1230-1/12/04.

[1] http://www.twitter.com/
[2] https://sites.google.com/site/microblogtrack/
[3] http://www.gnip.com/
[4] TwitterEcho is available open source at: http://labs.sapo.pt/twitterecho/

tion 2 presents a review of previous work that required obtaining large amounts of Twitter data, with the emphasis on the characteristics of the crawling system used. Section 3 identifies the key technical requirements that guided the crawler implementation. Section 4 describes the architecture of the crawler system. Section 5 presents the results of the system evaluation, and section 6 presents the conclusions and outlines future work.

2. RELATED WORK

This section reviews the crawling systems which have been used to support Twitter research in recent years. We divide these systems in two categories, depending on whether they make use of whitelisted accounts or not (see Appendix A.5).

Kwak et al. [12] did one of the first large crawls using whitelisted accounts, to study the topology of the Twitter network and its power for information sharing. They crawled Twitter from July 6th to 31st 2009, using 20 whitelisted machines with a self-imposed limit of 10.000 tweets per hour. The performed a breadth-first crawl starting with the user "Perez Hilton", and continued over his followers, whose number at the given time exceeded one million. To collect data from users disconnected from the main component, additional searches over the Search API Topics were conducted. The crawl collected in total 41.7 million users, 1.47 billion social relations and 106 million tweets. Cha et al. [4] studied influence on Twitter. They built a crawler containing 58 whitelisted servers requesting the data for each user ID ranging from 0 to 80 million. In total they collected 55 million users, more than 1.9 billion social relations, and almost 1.8 billion tweets. While the crawling period was similar to that of Kwak et al., Cha et al. managed to get 12 million users more. This approach proved better in reaching users disconnected from the giant component.

We observe that whitelisting is of great importance when trying to collect vast amounts of data. However, since it is no longer possible to get a whitelisted account, due to Twitter's current policy, researchers have to suit themselves with regular authenticated or non-authenticated access.

One of the first studies that collected data without whitelisted accounts focused on the topological and geographical properties of Twitter [11]. Java et al. crawled Twitter from April 1st to May 30th 2007, obtaining approximately 1.3 million tweets from 76 thousand users, using REST API methods. This crawl coincides with the embryonic stage of Twitter, when it was far less popular than it is today. Currently, this method would be insufficient to collect a significant amount of data. Wang [17] crawled Twitter from January 3rd to 24th 2010. During this period he collected around 500 thousand tweets and 49 million relations from 25 thousand users. Wang used the same API methods as Java et al. to extract recent tweets and their authors, and then used standard web crawling to retrieve social relations, along with the first 20 tweets of each new user. He used a distributed system with self-imposed limit of 120 requests per hour.

Several other researchers crawled Twitter, e.g. to investigate sentiment analysis [15], to study the propagation of URLs [9] and epidemic events in several countries [13]. Most of them were general systems focusing on specific data. In contrast, our goal is to provide a modular system capable of focusing the crawl on a specific community of users. The modularity of the system enables plugging in custom components that define the desired properties of the focused community.

From the studies reviews, we can conclude that:

- Different approaches to crawling Twitter (for details see the Appendix) lead to different performances. Combination of approaches is possible [17].
- Thorough knowledge of the API methods is needed. Several studies could be improved with a different choice of API or crawling method [9, 13].
- Collecting large amounts of data requires a distributed system [12, 17].
- Special care has to be taken to avoid overstepping the limits set by Twitter.

3. TECHNICAL REQUIREMENTS

Problems with previous crawling approaches led us to defining the following set of requirements for our crawler:

Adhering to limitations We must abide to Twitter's restrictions (see A.5). This is why we chose to use *users/lookup* method, giving us the most data per query.

Continuous operation The crawler must be able to continue operating during long periods of time, measured in months.

Run-time expandability Introduction of changes and improvements to the system should be enabled in runtime, thus keeping the data collection continuous.

Data completeness The system has to keep the whole crawling data (tweets, relations, statistics, etc.) in a structured form for thousands of users.

Fault tolerance The system has to be reliable and robust on failures in order not to compromise or lose data.

Modularity Researchers should be able to plug in their own modules, thus setting the crawling focus to a specific community defined by any criteria.

4. ARCHITECTURE

The need for large amounts of data, coupled with Twitter's imposed limits demand for a distributed system. We chose a centralized distributed architecture for our system, as depicted in Figure 1. By using multiple thin clients, we are able to create a scalable system that can cope with many requests towards the Twitter API. We directly influence the rate of the crawl by setting the appropriate number of clients. Since all the work is managed and distributed by a central server, we ensure continuous data collection if a client drops out. By "server" we are not restricting ourselves to a single machine. In fact, the role of the "server" may be supported by a cluster of machines that manage this centralized role. The central server is also made up of easily replaceable or upgradeable modules which enable the configuration of the crawl focus. We ensure run-time expandability by creating the whole process as a trigger and having no saved state of the server. The whole process is triggered on an HTTP request, and the possible code alteration (i.e. enabling a new module, or changing the old one) comes in effect on a new client connection.

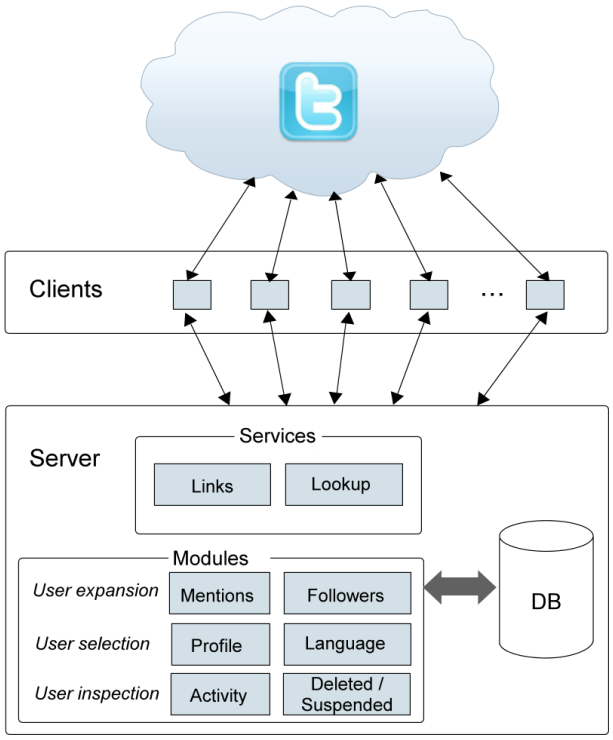

Figure 1: Architecture of the TwitterEcho crawler

For the tweet retrieval itself, we opted for the REST API and **users/lookup** method (see Appendix A.2). The method enables retrieval of the latest tweet and user information for 100 users in one API query. This gives us up to 30.000 possibly new tweets per client, per hour. We preferred REST API over Streaming API (see Appendix A.4) since the latter gives us only 1% of the whole Twitter stream. Since we are keen on crawling a focused community with a high coverage, coverage of 1% is not nearly enough.

By using the described architecture, we have successfully met our previously stated requirements.

4.1 Client

The client is implemented in two lightweight, non-obtrusive Perl scripts. The first script collects *tweets*, user profiles and simple statistics (number of tweets, followers and friends). It is executed every minute and consumes 300 API calls per hour. Since it can't guarantee higher coverage, we're using REST API with a prioritization scheme intended to maximize the coverage of Tweets. This script is unable to collect multiple tweets posted within the same minute by any given user, since the API call used returns only the last tweet per user.

The second client script collects *social network relations*, i.e. lists of friends and followers of a given set of users. Since these relations persist for longer periods of time and since each run consumes 50 API calls, there is no risk in loosing valuable information by running the script less often (e.g. every 10 minutes).

Both scripts ensure continuity of the crawling process within the rate limits, thus respecting Twitter's limitations. The frequency of the crawling process can be easily increased with the number of clients, assuming there is an adequate performance on the server side.

4.2 Server

The main tasks of the server are (i) coordinating the crawling process through allocation of user lists to each client, and (ii) maintaining the database of downloaded data. The modularity of the server enables user-control over both of these tasks.

The server is initially configured with a seed user list, e.g. a list of users belonging to the community of interest, and continuously expands the user list based on a particular expansion strategy. Such strategy is implemented through specific server modules, which need to be developed according to the research purpose. If the relevant community is, e.g. topic-driven, a new module would be implemented for detecting the presence of specific topics in tweets.

Our focus on the Portuguese Twittoshpere dictated building modules for checking if the users location lies in Portugal. Since some users do not state their location in the public profile, we implemented an additional module for automatic language identification.

Prioritizing

After a preliminary analysis of the data collected from the Portuguese Twittosphere, we observed that about 2.2% of users posted about 37% of the content, which highlighted the need to monitor active users' tweets more frequently than the inactive ones. This reduces the probability of tweet loss for the most active users and ensures scalability of the system. We solved this problem by introducing user priorities. The priority of a user, ranging from 1 to 100, where 100 denotes the highest priority, is increased upon retrieving a new tweet from the user, and decreased otherwise. Since our preliminary analysis also revealed lower activity on Twitter during the night, we freeze priorities for this period.

Using these priorities, the server assigns priority-based classes to users, and randomly selects users according to these classes (higher priority class, higher selection probability). The server then sends a list of these users, together with users from tentative lists of other modules (explained further in the text). The selected users are marked and disabled for reassignment for the next two minutes in order to prevent redundancy.

4.3 Modules

The client-server architecture tracks individuals from the pool of users marked as the community of interest (*links* and *lookup* services), by using modules executed in parallel with the primary crawl scheduling task. This is the key aspect that enables the crawler to gather focused data. Additionally, the system includes groups of modules responsible for controlling the expansion strategy.

This section describes the modules concerned with user expansion, selection, and inspection.

4.3.1 User expansion

The *user expansion* feature is supported by two modules whose task is to specify the strategy for growing the pool of users to be monitored. The first module analyses downloaded tweets and extracts users' *mentions* (explicit references to another user, e.g. *@9gag*). The module attempts to crawl the Twitter page of the mentioned user to check if the

profile exists, is valid, and is public. If so, the user is added to a list of tentative users. In a second phase, tentative users are further checked to decide if they should be added to the permanent list of users being crawled.

The second module adds users' followers to the list of tentative users. We are likely to find additional users from the targeted community among the followers of existing users. The same could be said for the list of friends. However, in our setting we use only the followers list, since a preliminary analysis revealed that Portuguese users tend to have a higher percentage of Portuguese followers than friends.

Other modules, such as the expansion by addition of friends, replied-to users, or users connected by the most popular topics, can also be easily implemented and integrated in our system to suit other research purposes.

4.3.2 User selection

The *user selection* feature expands the list of tentative users identified through the expansion modules, and its task is to identify accounts to be monitored by the system. Thus, the modules for user selection define the focus on the target community. For example, in the most basic case, a user selection module simply accepts all tentative users, leading to a general crawl of the Twittosphere. In practice, our system was conceived and optimized for focused crawling, and thus the modules should be designed for crawling specific communities. We implemented two modules supporting the data requirements of our research on the Portuguese Twittosphere: profile analysis and language identification.

Profile analysis

The profile analysis module uses a set of heuristics which consider information like the user's time zone, location and name to positively identify Portuguese microbloggers. Specifically, the module classifies the user as Portuguese the corresponding declared location and time zone fit to Portugal territory. If neither the time zone nor the location are consistent with Portugal [10], and if the user's name is not among the list of allowed Portuguese names [7], the user is classified as non-Portuguese. All other cases are deemed inconclusive. Preliminary tests revealed that this module classifies around 80% of users. For the classification of the remaining 20% we developed a language identification module.

Language identification

Language identification on Twitter is a non-trivial problem by itself due to the short length of the tweets, word misspellings, improper accentuation, informal writing style and novel word and term introduction. We implemented a language identification using a simple and fast unigram approach, as described by Cavnar et al. [3]. This approach is robust against spelling errors and is computationally inexpensive. The disadvantage is it requires a larger chunk of text for analysis.

We built a Portuguese unigram profile based on 15 thousand randomly selected tweets written in Portuguese, preprocessed to remove URLs, mentions, hashtags, accents, and common English words [18] and trigrams [5]. The language identification is performed by computing the distance between the built language profile, and the profile derived from 100 tweets from the given inconclusive user.

The main challenge in language identification in our case was to distinguish between European and Brazilian Portuguese. Since Brazil's population is roughly 18 times larger than Portugal's, special care needs to be taken to avoid crossing over with the Brazilian Twittosphere. The same problem exists with languages of former colonial empires, and of similar languages like South and East Slavic languages. We implemented several simple heuristic rules for elimination of Brazilian Portuguese. We relied on differences in written emotions ("kkk" and "rss" vs. "lol"), the use of gerund in Brazilian Portuguese, and a list of seventy words not used in European Portuguese (e.g. "ônibus").

4.3.3 User inspection

It is necessary to monitor events like deletion, suspension and the activity of users' accounts, while expanding the pool of users. Two modules perform these monitoring tasks. The first module checks deleted and suspended users, with the aim of adjusting the server's scheduling priorities. Inactive or suspended users are assigned the lowest priority, to avoid spending valuable API calls.

The second module detects users inactive for long periods of time with the aim of defining an appropriate schedule for crawling their data. For example, our crawler has collected data from around 100 thousand users up to this date, out of which approximately 60% did not publish for more than one month. This long period of inactivity indicates the need for sporadic checks. Given the reduced overall activity during the night period, the system is set to check inactive users between 3am to 7am.

4.3.4 Miscellaneous

Various other modules can be integrated in the crawling platform. For example, we deployed a module for tracking the growth and the health of the crawl database for statistical purposes and monitoring the server processes.

5. EVALUATION

Our evaluation of the crawler is two-fold. The first experiment tests the coverage of the crawler, i.e. if the tweet loss for the monitored users is not significant. The second experiment aims to evaluate if the crawler is keeping the focus on the target community, in this case the Portuguese Twittosphere.

5.1 Coverage

To evaluate the coverage of Twitterecho, we randomly sampled 2.608 users from our crawl database and fetched up to 200 of the latest tweets per user, yielding 148.228 tweets. A comparison with the tweets obtained with our crawler revealed an overlap of 111.485 tweets, thus attaining the coverage of 75.2%. We present the results, varied by the average activity of a user (average time between tweets) in order to show coverage over users of different activity. The results are presented in Table 1.

We observe higher tweet loss occurring for the most active users. However, the tweet coverage is reasonably high, steadily higher than 70% across different types of users. These results emphasize the importance of adopting a prioritization scheme for adapting the crawl frequency to the users activity level. If we checked all users with the same frequency, the coverage of highly active users would be very low, and vice versa for inactive users. Further equalization can be achieved by tweaking prioritization parameters.

Table 1: Tweet loss per average activity

Average activity	Users	Tweets fetched	Tweets crawled	Accuracy per user (%)
1h	127	21400	16073	74,0
3h	217	36463	26592	72,8
6h	216	32098	23153	71,7
12h	277	25287	19512	76,3
24h	378	16946	13078	76,8
3d	725	12197	9942	83,2
6d	358	2543	2046	84,3
12d	228	999	841	87,3
24d	82	297	248	88,7

5.2 Focus

The second evaluation aims to assess TwitterEcho's focus on the target community. The evaluation focuses on the effectiveness of the profile analysis and language identification modules.

5.2.1 *Profile analysis*

The crawler classifies users' nationality into three classes: Portuguese, non-Portuguese, and inconclusive (if there is insufficient data to classify it into the first two classes). At the time of evaluation, the crawl database contained 60 thousand users marked as Portuguese, around 500 thousand as non-Portuguese, and 170 thousand marked as inconclusive. We randomly selected 500 users per class, and performed manual labelling using user's profile, latest tweets and lists of friends and followers on users' Twitter web site. The selected samples did not contain any users from the seed list. Table 2 presents the results of the evaluation.

Table 2: Evaluation of user classification

User class	Correctly classified	Accuracy (%)
Portuguese	454	90.8
Non-Portuguese	497	99.4
User class	**Classified as Portuguese**	**Accuracy (%)**
Inconclusive	60	12.0

The results show that our crawler successfully focuses on Portuguese users. We observed that users falsely marked as Portuguese are mostly Brazilians living in Portugal. In order to reduce these cases, profile identification should work in parallel with the language identification module. Highest accuracy was achieved on users classified as non-Portuguese. Comparing this value with the estimated 55 thousand Portuguese users present in the database, this value would represent an increase of about 4.5% of users.

However, a significant number of Portuguese users was classified as inconclusive. The extrapolation of the results leads to circa 20 thousand users, which represents a significant increase of 36% of estimated true Portuguese users. Manual evaluation showed that lists of friends and followers played a significant role in the evaluation due to some users having little or no tweets. This highlights that additional modules (e.g. a module for friends and followers analysis) might be useful to increase recall of Portuguese users.

5.2.2 *Language identification*

To ensure presence of tweets in multiple languages, we sampled 550 users classified as Portuguese (75%) and inconclusive (25%), and randomly selected between 30 and 100 tweets per user. Prior to language identification, we preprocessed, concatenated and manually classified tweets, and additionally checked for the presence of multiple languages. The result of the language identification, encompassing precision and recall, is shown in Table 3.

Table 3: Evaluation of language identification

Tweets	Users	Precision	Recall
30-40	115	0,96	0,49
41-60	193	1	0,663
61-80	117	0,949	0,755
81-100	125	0,969	0,738
30-100	550	0,974	0,659

The results show high precision, regardless of the number of tweets in the sample. However, recall is lower due to the fact that profile comparison needs larger texts for classification. Further evaluation showed that the recall increased if we discarded all the mixed-language tweets, while the precision stayed unaffected.

6. CONCLUSION AND FUTURE WORK

TwitterEcho allows obtaining data from large Twitter communities, in our case comprising only Portuguese users. It does so through the use of a modular distributed system design, easily adaptable to target other communities. The distributed nature of the system enables crawling high volumes of data, while respecting the limits imposed by Twitter. The modularity of the system makes versatile, allowing the incremental addition of new functionalities. The system is currently operating continuously for more than nine months, which shows its robustness and reliability. Our evaluation showed high tweet coverage, and high focusing capabilities.

To this date, the crawler has already been successfully used to support two research projects. Firstly, it provided the data for an opinion mining system Twitómetro[5] [2]. This system monitored the sentiment of the Portuguese Twittosphere about the prime-minister candidates in 2011 parliamentary elections in Portugal. It provided Portuguese media a valuable insight into the sentiment of the electoral body and was widely covered by media. Secondly, the crawler is currently being used for social media analytics applications in the scope of an international research project REACTION (Retrieval, Extraction and Aggregation Computing Technology for Integrating and Organizing News). Through the scope of the project, TwitterEcho is used to obtain near-to-real-time Twitter data from the Portuguese community for further analysis as a part of the computational journalism platform for Portuguese. In general, we expect TwitterEcho to be of great use for any academic researcher or analyst who wishes to carry out research with focused Twitter data (e.g., for social network analysis, opinion mining, influence detection, trend analysis, etc.).

The current TwitterEcho setup has been running for 11 months now, and since the original evaluation, the user

[5] `http://legislativas.sapo.pt/2011/twitometro/`

database has grown at a steady pace to 81 thousand where it slowed its pace and started plateauing. This coincides with our initial estimates of the size of the Portuguese Twittosphere of around 100 thousand users. The current number of collected tweets is 14.5 million. TwitterEcho has been continuously operating with a varying number of clients, ranging from 1 to 18 concurrently, with up to 40 registered, periodically active clients. We assess that a single client could support around 30 clients concurrently.

Future work on TwitterEcho is oriented on improving scalability for tracking very large communities by distributing the server tasks. Scaling on the server side will be achieved by using distributed database technology such as HBase or Cassandra. Parts of the back-end processing will be moved to more specialized systems, for example graph databases for querying user graphs. Greater focus will be given to verification of highly active users using new scheduling methods. One of our priorities is increasing the precision of TwitterEcho's focus by improving the modules for user selection as well as the addition of modules for user's friends and followers analysis.

7. ACKNOWLEDGMENTS

This work is supported by Fundação para a Ciência e a Tecnologia (FCT) funded project REACTION: Retrieval, Extraction and Aggregation Computing Technology for Integrating and Organizing News (UTA-Est/MAI/0006/2009).

The authors would like to thank Jorge Texeira, Gustavo Laboreiro, and Arian Pasquali for valuable discussions and technical support.

8. REFERENCES

[1] D. Antoniades, I. Polakis, G. Kontaxis, E. Athanasopoulos, S. Ioannidis, E. P. Markatos, and T. Karagiannis. we.b: the web of short urls. In *Proceedings of the 20th International Conference on World Wide Web (WWW)*, pages 715–724, 2011.

[2] P. Carvalho, L. Sarmento, J. Teixeira, and M. J. Silva. Liars and saviors in a sentiment annotated corpus of comments to political debates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers - Volume 2*, pages 564–568, 2011.

[3] W. B. Cavnar and J. M. Trenkle. N-gram-based text categorization. In *In Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, 1994.

[4] M. Cha, H. Haddadi, F. Benevenuto, and K. P. Gummadi. Measuring user influence in twitter: The million follower fallacy. In *Proceedings of the 4th International AAAI Conference on Weblogs and Social Media (ICWSM)*, May 2010.

[5] J. Cowan. English trigram frequency table. `http://home.ccil.org/~cowan/trigrams`. Accessed: 31/08/2011.

[6] N. A. Diakopoulos and D. A. Shamma. Characterizing debate performance via aggregated twitter sentiment. In *Proceedings of the 28th international Conference on Human Factors in Computing Systems*, pages 1195–1198, 2010.

[7] P. do Cidadão. Atribuição do nome a um recém-nascido. `http://www.portaldocidadao.pt/PORTAL/entidades/MJ/IRN/pt/SER_atribuicao+do+nome+a+um+recem+nascido.htm`. Accessed: 31/08/2011.

[8] M. Efron. Hashtag retrieval in a microblogging environment. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 787–788, 2010.

[9] W. Galuba, K. Aberer, D. Chakraborty, Z. Despotovic, and W. Kellerer. Outtweeting the twitterers - predicting information cascades in microblogs. In *Proceedings of the 3rd conference on Online Social Networks (WOSN)*, pages 3–3, 2010.

[10] T. Grader. Top twitter cities. `http://tweet.grader.com/top/cities`. Accessed: 31/08/2011.

[11] A. Java, X. Song, T. Finin, and B. Tseng. Why we twitter: understanding microblogging usage and communities. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, pages 56–65, 2007.

[12] H. Kwak, C. Lee, H. Park, and S. Moon. What is twitter, a social network or a news media? In *Proceedings of the 19th International Conference on World Wide Web (WWW)*, pages 591–600, 2010.

[13] L. Lopes, J. Zamite, B. Tavares, F. Couto, F. Silva, and M. Silva. Automated social network epidemic data collector. In *INForum - Simpósio de Informática September*, 2009.

[14] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[15] A. Pak and P. Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of the 7th conference on International Language Resources and Evaluation (LREC)*, May 2010.

[16] D. Sousa, L. Sarmento, and E. Mendes Rodrigues. Characterization of the twitter @replies network: are user ties social or topical? In *Proceedings of the 2nd international workshop on Search and Mining User-generated Contents (SMUC)*, pages 63–70, 2010.

[17] A. H. Wang. Don't follow me: Spam detection in twitter. In *Proceedings of the International Conference on Security and Cryptography (SECRYPT)*, July 2010.

[18] Word-English. The 500 most commonly used words in the english language. `http://www.world-english.org/english500.htm`. Accessed: 31/08/2011.

APPENDIX

A. CRAWLING TWITTER

In this appendix we provide an outline of the different ways of crawling Twitter, including their advantages and disadvantages, as a short reference point to researchers.

A.1 Web crawling

Twitter provides access to users' information through a public profile over a web interface, obtainable by web crawling. The advantage of this approach is limitless crawling. Disadvantages are higher response time, indirect data need-

ing parsing, and less obtainable information (no numeric IDs). Luckily, Twitter offers three distinct APIs.

A.2 REST API

REST API is a very comprehensive interface that allows performing virtually all actions available on Twitter. Its disadvantage is the need to conform to the imposed limits. Out of the available methods, we used:

statuses/public_timeline Returns 20 most recent tweets

statuses/user_timeline Returns up to 20 of the most recent tweets from a user, capable of returning up to 3.200 tweets in total

users/lookup Returns profile information of up to 100 users including the most recent tweet, if not protected

friends/ids, followers/ids Returns the numeric IDs of friends and followers, respectively

statuses/followers Returns profile information of 100 followers of a more recent user (including the last tweet)

A.3 Search API

Search API is an interface for information search on Twitter. It allows searching for specific users and tweets content. The advantage of this API is that its imposed limit is higher and independent of the limit of the REST API. The disadvantages are the mismatch with REST API identifiers, and the returning list of tweets ordered by relevance, thus leaving out tweets deemed irrelevant by the retrieval engine.

A.4 Streaming API

The Streaming API allows obtaining tweets in near real-time from a subset of Twitter data, depending on the access level. Three distinct access levels: Spritzer, Gardenhose and Firehose, enable fetching 1%, 10% and 100% of the Twitter data stream, respectively. Spritzer access level is the only one available to the general public. The main advantages of Streaming API are the high-throughput access to Twitter data, and a support for a limited set of filters. However, disadvantages are the limited default access level (1% coverage of all the tweets when not using filtering capabilities), and the inability to fetch data prior to establishing a connection. Streaming API is very useful for a general Twitter crawling, however it is not so useful for a focused crawling due to low coverage which gets even lower when using filters.

A.5 Twitter limits

All Twitter APIs impose a limit to the number of queries a user can issue hourly. The REST API allows 150 queries per hour per IP for anonymous users, 350 queries per hour per user for authenticated users, and 20.000 queries per hour per user for whitelisted users. Requests for whitelisting an account are no longer possible.

The Search API does not require authentication. However, like the REST API it limits the number of queries. This imposed limit is per IP and the exact quantity of allowed queries is not disclosed, but is higher than the REST API limit. Our tests reveal that the limit is up to about 20 thousand queries per hour per IP.

The Streaming API requires authentication and imposes a limit of one connection per account. Many of the methods of the API are available only to users given a higher access level.