Focused Crawling of Online Networked Data

Baris Can Esmer, Arman Boyaci and A. Taylan Cemgil

January 28, 2019

Abstract

- Define focused crawling Focused crawling of web is well-studied in the literature. In this paper we propose a focused crawler architecture for social network data.

1 Introduction

The initial motivation for this work is the task of identifying a community of people that share a common interest and are geographically co-located. A slow and somewhat restricted approach for identifying a community is using a referral system. However, such an approach is very labor intensive and hard to automate as it requires personal human to human interactions. In recent years, with the availability of large amounts of publicly available social media data, there is an increasing trend of automatically detecting such communities on social networks. However, several technical challenges still appear in this automation.

Social networking sites, such as Twitter provide a rich source of dynamic information that is potentially useful for identifying communities of people and organizations related to a specific subject. However, even accessing a tiny fraction of this massive data is difficult for third parties due to bandwidth restrictions or cost barriers imposed by commercial or privacy concerns.

A typical analysis on Twitter from the perspective of a third party that does not have a direct access to the entire data and needs to respect bandwidth and query restrictions proceeds often as follows: First, one formulates a task such as identifying a group of users potentially interested in some topic. Then, using certain heuristics, such as retrieving the followers of manually selected nodes (such as an influencer) a large set of users is identified. Alternatively, it is possible to listen to the streaming API and storing users related to twit s that contain specific keywords. The hope is that this set contains the desired community as a subset. Often, it is difficult to retrieve the individual profile data of the entire set, but even more difficult is retrieving the connections due to large number of edges.

As a concrete technical problem, we will focus on identifying the geographical locations of individual accounts, hence refining the overall information about a community. Here, the term community refers to a group of Twitter accounts that are related to a particular topic such as 'Artificial Intelligence, Maker Movement, Finance'. A simple keyword or link based approach can easily identify a large set of candidate accounts that are interlinked. However, obtaining an accurate location information in the desired granularity (such as a city, country or a continent) becomes more difficult because only half of the accounts provide any location information, if at all.

Our observation is that the location information can often be inferred from other profile data, such as the name, language, time zone, or profile image. Even a more informative indicator is the location information of neighbors (other accounts followed or friended). However, it is difficult to obtain the latter information due to bandwidth restrictions.

In many social data analysis tasks, the common workflow is to retrieve an initial chunk of data, then making a batch analysis and then possibly refine the research question, model or algorithm and reiterate this procedure. Here, we will develop a more efficient approach that is aware of the data access restrictions.

In machine learning and statistics, such problems are investigated under the terms of active learning and experiment design...

2 Related Work

The goal of a *focused crawler* is to selectively seek out pages that are relevant to a pre-defined set of topics. ([CvdBD99] [VSG14])

- In [HWS18] the authors propose a focused Web crawler that is designed to collect Web pages relevant to a predefined topic. The crawlers main goal is to retrieve as many relevant pages while avoiding irrelevant ones. Relevancy to target topic is defined as cosine similarity between word vector of Web page and word vector of topic. The Web page itself is enough for calculating relevancy. (Unlike Twitter, where a user profile is an abstraction of the user and may not always have enough information for relevancy) They model the crawling environment as an MDP, where Web pages are the states and hyperlinks in a Web page are the actions. State-action value for each hyperlink is computed (called priority in the paper) and all links are added to a priority queue, sorted by priority value.

  Database dump of Simple English Wikipedia is used to measure performance of the crawler. They choose 6 target topics, where 3 of them have a lot of relevant pages and 3 of them do not have much. In order to compare 3 different algorithms (1 without learning and 2 that uses reinforcement learning), for each topic they run the crawlers until they collect 10000 pages. Then they use the number of relevant pages as a performance metric and compare the algorithms.

- Anthelion [MMB14] is also a focused Web crawler and its goal is to find web pages that contain structured data embedded in them (e.g. schema.org markup). They use online learning to predict which links may contain structured data, this is possible because after downloading a link it can be seen whether the page contains any structured data. To cope with exploration/exploitation problem, they model the hosts as arms in a multi armed bandit setting and evaluate different bandit functions using λ-Greedy selection method.

  The datasets used for the experiments are extracted from the publicly accessible dataset provided by the Common Crawl Foundation. In the first part of the experiments, they compare online and batch learning(percentage of relevant fetched pages during crawling) and the results show that online learning performs better at the end of the crawl. They also compare different bandit functions and lambda values (including decaying λ).

- TwitterEcho ([BOM+12]) is an open-source Twitter crawler that helps researchers collect large amount of data while respecting Twitter's rate limits. Unfortunately the code is not available. It has a modular and distributed architecture. They used TwitterEcho to find people from Portuguese Twittosphere, however the modules can be changed depending on the research interest. The paper lacks mathematical background and the aim of the crawler is to collect huge amount of data with the help of a distributed system rather than focusing on a community with same features.

- In [Lud14] they try to infer a users gender based on features obtained from her tweets and also from celebrities that she follows (age of celebrity, gender of celebrity ...). A list of *celebrity users* is obtained for this task. They use SVM-based classifiers for gender classification. The network structure is initially known and is used to create features.

  They use a previously collected database which contains 131 male and 134 female labeled users. Using 10-fold cross validation, they measure the performance of the SVM classifier in terms of accuracy and obtain 85.67%.

- In [TLLS17] the main goal is to identify profession of social media users. They divide the information from Twitter into several different sources and and then use a cascaded two level classifier for prediction. They also use network structure for refining the profession identification but they do not use crawling.

  For experiments they use a database with more than 60,000 *manually annotated* microblog users from a microblog service in China. 14 different professions are selected such as art, government, etc ... The dataset is divided into test and training sets, other than that, macro-averaging precision/recall/F-Measure is used for evaluation metrics. They claim that all metrics used for evaluation are larger than 80%.

3 Similarities and Differences of Crawling Web and Twitter

There are some differences between web crawling and crawling in Twitter, such as

Focused Crawler (WEB)	Twitter Network
1. Object : Document/Webpage	1. Object : Profile
2. Relevancy : Deterministic methods such as calculating cosine similarity between word vector of the topic and word vector of the web page	2. Relevancy : Calculated by pretrained classifiers (depends on network structure or only local information)
3. (Crawling) Action : The crawler chooses next URL to follow.	3. (Crawling) Action : Crawler chooses the source to fetch its followers (5000)
4. (Guessing) Crawling Decision : The crawler makes use of the information about the URL in order to make a guess, such as anchor text, url text etc ...	4. (Guessing) Crawling Decision : ?? From source ??
	5. Network Info : Used to improve confidence of predictions, RMF, message passing ?
5. Network Info : Does not use	6. Seed : Profiles that we know are relevant
6. Seed : Wikipedia page of the topic	

4 Objective

Each user has a profile, of which some fields are observable (age, location but the value can be null or noisy) and some are not (gender). Our objective is to find users that satisfy the given criteria imposed on these fields. However we have access to limited number of users at the beginning and we need to crawl the network of users to enlarge our set of relevant users.

For observable fields, there are missing values for some users. But even when the value is not missing, there are cases where there is uncertainty. Location field is an example for such a case, there are a lot of places in different countries whose names are the same. (Boston in UK and USA)

For unobservable fields the values are unknown and supervised learning is not possible.

We assume that for each feature a classifier is already is trained and these classifiers use only observable features.

More formally, let $u \in U$ be a user and O and F be sets of observable and unobservable features, respectively. For $o \in O$, u_o is the value of the feature o for user u, where u_o can be NULL or need some kind of processing. u_f is unknown for $f \in F$, but we still want to find $U' = \{u \in U \mid u_f = \text{value we want}\} \subseteq U$. Generally, our objective is to find users that satisfy the criteria imposed on the features. Let Ω be the set of the criteria, i.e. $\Omega = \{(X_i, Y_i) \mid X_i \in O \cup F\}$ where Y_i are the respective values for features X_i. What we want is $U_\Omega = \{u \in U \mid u_{X_i} = Y_i \quad \forall X_i \in \Omega\}$. We have a set of seed users S to start crawling and our aim is to find users that we think may belong to U_Ω.

5 Relevancy in Twitter

In the most general sense, our aim is to get answers to our questions from an online networked data service. More formally, each question is a query, consisting of several features and values. What we want is a set of users that satisfy the given conditions. For example, we may want to find people living in *Turkey* and interested in *Artificial Intelligence*, the query for this question would be

$$\{country : \text{“Turkey”}, interest : \text{“Artificial Intelligence”}\}$$

However, it is not easy to identify users that satisfy these conditions because of

1. Uncertainty
   A feature in the query might not be in the user profile (e.g. interest), or it can be in the profile but its value can be empty or ambiguous (e.g. country information, in a twitter profile location field may be empty or country must be inferred indirectly from data).

2. Restrictions
   Most online networked data are huge and this alone is a restriction in terms of time and resources. Other than that, there are additional restrictions imposed by the data provider itself (e.g. *Twitter*), such as being able to retrieve a limited number of profile data in a pre-determined time interval. To cope with these problems, a clever optimization scheme must be used.

6 Giving Answers

Giving an answer to the query requires solving two other problems that are interlinked.

6.1 Crawling

In real life, the data from huge networks are only available in terms of a graph structure and direct list of users in this graph is usually not obtainable. However, in order to identify users satisfying some criteria, we need to have a list of users at hand.

Many times, one has an initial set of users to begin with, called *seed users*. These are sometimes obtainable from the data provider (e.g. Twitter gives a set of influencer users) or already present because of the nature of the problem. However, it is very unlikely that they are sufficient for the application so one needs to enlarge this set. This is the discovery problem, i.e. the problem of deciding how to enlarge this set and add new possible users.

One naive solution to this problem is to simply add all followers of the seed users. The depth can be changed to add more users, e.g. followers of the followers etc. This does not however take the query into consideration and is not very effective. Consider the query in the previous section. Followers of some seed users can be very likely to satisfy a condition in the query (e.g. an influencer from Turkey) and for others this might be false (e.g. an influencer from Switzerland with moderate number of followers). If we increase the depth for these *great* influencers, the performance is likely to increase.

The naive solution above first handles discovery problem and then goes on, however a more sophisticated method would dynamically increase the set depending on which users satisfy the conditions given in the query.

6.2 Validation

As discussed before, most of the features in a query need an estimation. For each user, an estimation of the feature can be made with her own *local* data, however this does not take network account into account. Using network information can improve the accuracy of the estimation greatly.

For example, assume that we want to find programmers in a network. Local data of a user does not give enough information but the followers of a user can imply if a user is a programmer or not. If most of the followers of a user are predicted as programmers, it is very likely that the user itself is also a programmer.

These two problems can be approached in a sequential manner or one can come up with a solution that tries to solve both of them (which would increase performance (?)).

7 Model

On Twitter, any user can specify their geographical location in a given granularity, such as city, country. The GPS location, i.e., exact coordinates being present when sending a twit can also be provided if the user allows the so-called geolocation feature; however only a tiny fraction of all users prefer to allow access to such detailed information.

On the other extreme, about 40 percent of all users on Twitter do not provide any location information, even not a country. However, a human observer can easily guess a likely location from profile information. In this section, our aim is developing a technique for automating this process. First, we will formulate a model relate the location of the user to relate to other side information available from her profile and timeline.

The Twitter API enables users to share other information on their profile page that can be used to determine indirectly the location. Such information include

- Name, surname
- Timezone
- Language

Additionally, since Twitter is a *n*etworked micro-blogging service, meaning that each user is connected to other users in terms of friend, follower relations, or retwitting one users twits, the locations of the neighbors of a user also provide an indication about her location.

We formulate the location estimation problem as a classification problem. To each user u, we want to assign a location $c \in C$ where user u has the features described above; name, surname, timezone, language and locations of the followers.

We can choose Naive Bayes Classifier as our method and if we denote (name,timezone, language, location information of followers) as a vector, $\vec{x}$, then for each location c, we can write

$$p(c \mid \vec{x}) = \frac{p(\vec{x} \mid c)p(c)}{p(\vec{x})} \tag{1}$$

and then

$$\arg\max_c \; p(c \mid \vec{x}) \tag{2}$$

becomes

$$\arg\max_c \; p(\vec{x} \mid c)p(c) \tag{3}$$

and finally, using the independence assumptions, we arrive at

$$\arg\max_c \; p(\vec{x} \mid c)p(c) = \arg\max_c \; p(\text{name} \mid c)p(\text{timezone} \mid c)p(\textit{language} \mid c)p(\text{locations of followers} \mid c)p(c) \tag{4}$$

We can find approximations for each term in the product and then classification of a user is determined by the formula above.

References

[BOM+12] Matko Bošnjak, Eduardo Oliveira, José Martins, Eduarda Mendes Rodrigues, and Luís Sarmento. Twitterecho: A distributed focused crawler to support open research with twitter data. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12 Companion, pages 1233–1240, New York, NY, USA, 2012. ACM.

[CvdBD99] Soumen Chakrabarti, Martin van den Berg, and Byron Dom. Focused crawling: a new approach to topic-specific web resource discovery. *Computer Networks*, 31(11):1623 – 1640, 1999.

[HWS18] Miyoung Han, Pierre-Henri Wuillemin, and Pierre Senellart. Focused crawling through reinforcement learning. In Tommi Mikkonen, Ralf Klamma, and Juan Hernández, editors, *Web Engineering*, pages 261–278, Cham, 2018. Springer International Publishing.

[Lud14] Puneet Singh Ludu. Inferring gender of a twitter user using celebrities it follows. *CoRR*, abs/1405.6667, 2014.

[MMB14] Robert Meusel, Peter Mika, and Roi Blanco. Focused crawling for structured data. In *Proceedings of ACM International Conference on Information and Knowledge Management*, 2014.

[TLLS17] Cunchao Tu, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. Prism: Profession identification in social media. *ACM TIST*, 8:81:1–81:16, 2017.

[VSG14] George Valkanas, Antonia Saravanou, and Dimitrios Gunopulos. A faceted crawler for the twitter service. In Boualem Benatallah, Azer Bestavros, Yannis Manolopoulos, Athena Vakali, and Yanchun Zhang, editors, *Web Information Systems Engineering – WISE 2014*, pages 178–188, Cham, 2014. Springer International Publishing.