Towards Twitter User Recommendation Based on User Relations and Taxonomical Analysis

Kristian SLABBEKOORN, Tomoya NORO and Takehiro TOKUDA
Department of Computer Science, Tokyo Institute of Technology, Japan

Abstract. Twitter is one of the largest social media platforms in the world. Although Twitter can be used as a tool for getting valuable information related to a topic of interest, it is a hard task for us to find users to follow for this purpose. In this paper, we present a method for Twitter user recommendation based on user relations and taxonomical analysis. This method first finds some users to follow related to the topic of interest by giving keywords representing the topic, then picks up users who continuously provide related tweets from the user list. In the first phase we rank users based on user relations obtained from tweet behaviour of each user such as retweet and mention (reply), and we create topic taxonomies of each user from tweets posted during different time periods in the second phase. Experimental results show that our method is very effective in recommending users who post tweets related to the topic of interest all the time rather than users who post related tweets just temporarily.

Keywords. Recommendation, Twitter, Microblog, Taxonomy, Wikipedia

Introduction

Social media enables us not only to communicate with others but also to deliver and share information in real-time. Twitter is one of the most popular social media platforms, with 200 million active users (Dec. 2012) [1] delivering 400 million tweets a day (Jun. 2012) [2]. Due to the large number of Twitter users, finding users [1] who often provide valuable information related to the topic of interest is difficult. In most cases, we find such users by someone's recommendation or just by chance. Twitter user recommendation systems will help us find good users to follow that post related to the topic of interest.

One simple way to find such users is to search for tweets by some keywords related to the topic of interest and see how many tweets each user posted. However, this method has two problems. First, counting the number of related tweets is not enough. Some users post many tweets for advertisement and so on; in most cases, recommending such users is meaningless since they are usually considered as spammers. To avoid such users, we need to look at user relations to measure user influence as well as look at the tweet count of each user. Second, it is difficult for us to choose appropriate keywords to search for related tweets, which means we cannot get enough tweets just by giving

[1] In this paper, we refer to "Twitter user" as "user".

some keywords to the Twitter search. Suppose that we would like to get tweets related to malware. Searching for tweets including a word “malware” is not enough for getting related tweets since each tweet is less than 140-characters long and not all of the related tweets include “malware”. Some tweets may include names of malware, Internet security companies, technology related to malware, and so on. Since it is almost impossible for us to list all keywords related to the topic of interest, we need to take the deeper semantics of tweets into consideration.

To solve these problems, we present a method for user recommendation based on user relations and taxonomical analysis. First, our method finds users to follow related to the topic of interest based on provided keywords, then picks up users who continuously post related tweets from the user list. In the first phase, we consider tweet behaviour of each user such as retweet and mention (reply) as user relations. Given some keywords related to the topic of interest, our method searches for tweets including the keywords, extracts user relations based on tweet behaviour among users from the obtained tweets, then ranks users, taking the discovered user relations into account. In the second phase, we create simple topic ontologies, or *taxonomies*, of each of the users ranked high after the first phase from tweets collected during different time periods, then determine whether each user continuously post tweets related to the topic of interest, which is not directly considered in the first phase. We adopt taxonomical topic analysis in this phase instead of keyword matching as in the first phase to cope with the second problem mentioned above; that is, we cannot encapsulate the entire topic of interest using a (manually created) list of keywords alone. Experimental results show that our method can recommend users who consistently post tweets related to the topic of interest, and exclude users who post related tweets just temporarily.

The organization of the rest of this paper is as follows. In section 1, we present some related works. The first phase of our method, user recommendation based on user relations, is described in section 2, followed by the second phase, user recommendation based on taxonomical analysis, in section 3. Section 4 shows some evaluation results and lastly we conclude this paper in section 5.

1. Related Work

Twitter provides its own user recommendation service. Basically, it recommends users who have mutual followers/friends and does not directly consider what we are interested in. We need to follow some users related to the topic of interest in advance to have the service give an appropriate recommendation result. We can also search for users by the Twitter keyword search service. However, it mainly returns users whose screen names or profiles match the input keywords and does not care whether they actually post tweets related to the keywords.

TwitterRank [4] finds influential users by taking topical similarity among users and link structure (follow relation) into account. Although it returns influential users for each topic cluster, we cannot control how topics are clustered. As a result, we cannot always find an appropriate cluster corresponding to the topic of interest.

Twittomender [5] uses lists of followers, friends, and terms in their tweets to find users related to a particular user or query. It calculates similarity between users by representing each user as a weighted term vector. However, it does not take the overall semantics of a user's tweets into consideration, which reduces chance of finding relevant users since each tweet is limited to 140 characters and has little information.

Syed et al. [6] proposed a scheme for using Wikipedia as an ontology for describing documents. They demonstrate several algorithms for finding named entities in a document and to leverage the Wikipedia category hierarchy in order to find a set of topics for one or more documents. We exploit the Wikipedia category hierarchy in a similar way, but we focus on finding topics of interest of a user from tweets rather than domain-specific articles, which makes our problem quite different in nature and significantly more difficult.

Michelson and Macskassy [7,8] described methods for entity-based topic detection focusing on tweets mined from Twitter. To deal with the noisy nature of tweets, they only consider capitalized words (i.e. proper nouns) as possible entities, and disambiguate them to Wikipedia pages by calculating overlap of contexts between the source tweet and target Wikipedia page candidates. For each page, categories are selected and ranked based on their frequency of occurrence and position in the category hierarchy (more specific categories are prioritized). Our approach is similar, but considering only proper nouns is not an option for us due to the sparsity of data, and because we want to be able to discover any kind of topic, which may not always be represented by proper nouns.

Nakatsuji et al. [9] used taxonomies to model user interests and how they change over time. Focusing on specific domains of interest, they apply hand-crafted taxonomies to classify entities. Item recommendations are made for an active user based on the similarity of (super-)topics with target users, where not only the current time frame is examined, but also tweets within certain time frames that target users made in the past. Our approach is not aimed at item recommendation but user recommendation. We model a user's entire recent tweet history to recommend users who consistently post tweets about a certain topic.

2. User Recommendation Using User Relations

Firstly, we rank users using user relations. We made some modifications to our previous work, TURKEYS [10]. In our approach, given an input query representing the topic of interest, we first get tweets matching the query using the Twitter keyword search, then extract user information and user relations from them. After we create a reference graph based on the user relations, we calculate the TURKEYS score of each user. We consider tweet behaviour of each user such as retweets and mentions (replies) as user relations.

We assume the following.

1. Users who post many valuable tweets and retweets about the topic are worth following.
2. Valuable tweets attract attention from many users.
3. Each user pays attention to tweets he/she retweets or replies to. Also, he/she watches the other tweets to some extent (not more than retweets and reply tweets).

Based on these assumptions, the TURKEYS score of each user u is defined as follows.

$$\mathrm{TURKEYS}(u) = \mathrm{TC}(u)^{w} \times \mathrm{UI}(u)^{1-w} \tag{1}$$

$\mathrm{TC}(u)$ and $\mathrm{UI}(u)$ are respectively tweet count score and user influence score of user u, ranging between 0 and 1 (details are described later). Weight w also ranges between 0 and 1. The tweet count score is based on tweet count of each user and reflects the first assumption; the user influence score is based on user relations among users and reflects the second and third assumptions.

2.1. Collection of User and Tweet Data

We collect users and tweets as follows.

1. Given an input query representing the topic of interest, get tweets matching the query posted in the last n days by the Twitter search API [2]. Let this tweet set be T_0.

   (a) Remove tweets where the query does not match the tweet text but matches link URLs or user names, since the Twitter search API returns such tweets.

   (b) Remove duplicate tweets (the same tweet text posted by the same user at different times) since some users post the same tweets repeatedly and in most cases they are not valuable (ads or spam).

   (c) If the tweet text starts with "RT @username:", get the original tweet information of the retweet by the Twitter API *statuses/show/id.*

2. Get tweets replying to tweets in T_0 posted in the same time period as the first step by the Twitter search API. Let this tweet set be T_r.

$$T_r = \{t | t' \in T_0 \wedge t'.id = t.in\_reply\_to\_status\_id\} \tag{2}$$

   where $t'.id$ indicates the ID of tweet t' and $t.in\_reply\_to\_status\_id$ indicates the tweet ID replied to by tweet t. We can get tweets replying to a user who posted some tweets in T_0 by giving the user name prefixed with "to:" to the Twitter search API as a query. Then we check the "in_reply_to_status_id" value of each of the tweets.

   Although the Twitter API *related_results/show/id* was used in [10], we use the Twitter search API instead for two reasons. The first reason is that *related_results/show/id* is not officially supported by Twitter and at most 8 tweets can be obtained, which means the API does not always return all reply tweets. As a result, some influential users who receive many reply tweets would be ranked low. The second reason is that we only need to call the search API as many times as the number of users who posted tweets in T_0, while we have to call *related_results/show/id* as many times as $|T_0|$, which is more than the number of calls to the search API [3]. We can give several user names to the search API at once by joining them via the "OR" operator, which may further reduce the number of API calls.

[2] The Twitter search API returns at most 1500 tweets. However, we can get more tweets by tweaking the value of the "max_id" attribute (ID of the latest tweets to be obtained). n should be less than 6 since the Twitter search database is different from a complete database of all tweets and only tweets posted in the last 6-9 days are stored in the Twitter search database.

[3] Actually, the number of search API calls would be more than the number of users since some popular users receive many reply tweets and we have to call the search API more than once to get all of them. However, usually, the number of API calls is still less than $|T_0|$.

3. For each tweet in $T_0 \cup T_r$, get the poster's name, the tweet ID, user ID and user name the tweet replies to (in the case of a reply tweet), tweet ID and poster name of the retweeted status (in the case of a retweet), and user names in the tweet text (strings preceded by "@"). Let the set of tweets obtained and the set of users obtained be T_{all} and U_{all} respectively.

2.2. *Tweet Count Score*

The tweet count score is calculated by counting not only original tweets but also retweets in T_0 as each user's own tweets. The score is dampened by a logarithm function and is normalized so that the largest possible value is 1.

$$\mathrm{TC}(u) = \frac{\log\left(1 + |\{t|t \in T_0 \wedge t.user.id = u.id\}|\right)}{\max_{u' \in U_{all}} \log\left(1 + |\{t|t \in T_0 \wedge t.user.id = u'.id\}|\right)} \tag{3}$$

where $t.user.id$ indicates poster's ID of the tweet t and $u.id$ indicates ID of the user u.

2.3. *User Influence Score*

In our approach, we apply the user-tweet version of the user influence score calculation in [10] [4]. In this version, we define not only the user influence score of each user but also tweet influence score of each tweet. The user influence score of each user is calculated using the tweet influence score of original tweets and retweets the user posted, and the tweet influence score of each tweet is calculated using the user influence score of users who pay attention to the tweet.

We create a tweet-user reference graph from the tweet set T_{all} consisting of user nodes, tweet nodes, and directed edges between a user node and a tweet node. We use A_t and A_r to denote adjacency matrices of the graph. A_t represents what (tweet) is posted/retweeted by whom (user) and is used for calculating the user influence score, and A_r represents who retweets/replies to what and is used for calculating the tweet influence score.

$$A_t(t_i, u_j) = \begin{cases} 1 & \text{if } u_j \text{ tweets/retweets } t_i \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

$$A_r(u_j, t_i) = \begin{cases} 1 & \text{if } u_j \text{ retweets or replies to } t_i \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where t_i and u_j indicate the i-th tweet and the j-th user respectively ($1 \leq i \leq |T_{all}|$ and $1 \leq j \leq |U_{all}|$).

The adjacency matrices A_t and A_r are transformed as follows.

[4] In [10], we used the 1000 latest tweets for the score calculation and result of the user-only version was a little better than result of the user-tweet version. However, we found that the user-tweet version outperforms the user-only version if we use tweets posted in the last 5 days (usually more than 1000 tweets).

```
$\mathbf{u}_0 = (\frac{1}{|U_{all}|}, \frac{1}{|U_{all}|}, \ldots, \frac{1}{|U_{all}|})$;
$\mathbf{t}_0 = (\frac{1}{|T_{all}|}, \frac{1}{|T_{all}|}, \ldots, \frac{1}{|T_{all}|})$;
$k = 1$;
Repeat
    $\mathbf{u}_k = B_t^T B_r^T \mathbf{u}_{k-1}$; $\mathbf{t}_k = B_r^T B_t^T \mathbf{t}_{k-1}$;
    $k = k + 1$;
until $|\mathbf{u}_k - \mathbf{u}_{k-1}| < \varepsilon_u$ and $|\mathbf{t}_k - \mathbf{t}_{k-1}| < \varepsilon_t$;
return $\mathbf{u}_k$ and $\mathbf{t}_k$;
```

Figure 1. Calculation Algorithm of the User Influence Score and the Tweet Influence Score

$$B_t(t_i, u_j) = \frac{A_t(t_i, u_j)}{\sum_k A_t(t_i, u_k)} \tag{6}$$

$$B_r(u_j, t_i) = \begin{cases} \frac{A_r(u_j, t_i)}{\sum_k A_r(u_j, t_k)}(1-d) + \frac{d}{|T_{all}|} & \text{if } \sum_k A_r(u_j, t_k) \neq 0 \\ \frac{1}{|T_{all}|} & \text{otherwise} \end{cases} \tag{7}$$

where d is a damping factor of $0 \leq d \leq 1$. A_t is transformed just by dividing each element by the corresponding row sum. Note that $\sum_k A_t(t_i, u_k) \neq 0$ since any tweet is posted by at least one user. Transformation of A_r reflects the third assumption described at the beginning of this section. Each user pays attention to tweets he/she retweeted or replied to. Also, he/she watches all tweets (regardless of his/her activity of retweet and reply) at a certain rate of "d".

The user influence score and the tweet influence score are calculated as follows.

$$\mathbf{u} = B_t^T \mathbf{t} \qquad \mathbf{t} = B_r^T \mathbf{u} \tag{8}$$

where $\mathbf{u}$ and $\mathbf{t}$ indicate the column vector of the user influence score of all users and the column vector of the tweet influence score of all tweets respectively. We can calculate the user influence score and the tweet influence score using the power iteration method. The iterative processes are as follows.

$$\mathbf{u}_k = B_t^T B_r^T \mathbf{u}_{k-1} \qquad \mathbf{t}_k = B_r^T B_t^T \mathbf{t}_{k-1} \tag{9}$$

where $\mathbf{u}_k$ and $\mathbf{t}_k$ indicate the user influence score and the tweet influence score at the k-th iteration respectively. The calculation algorithm of the user influence score and the tweet influence score is shown in Figure 1. ε_u and ε_t are error tolerance parameters. Lastly, the user influence score of each user is normalized so that the largest value should be 1.

$$\mathrm{UI}(u_j) = \frac{\mathbf{u}(j)}{\max_k \mathbf{u}(k)} \tag{10}$$

The TURKEYS score defined in Eq. (1) is calculated using Eqs. (3) and (10). We take the top-k ranked users as candidates for the next phase.

3. User Recommendation Using Taxonomical Analysis

Once we have obtained a ranking of users based on keywords and user influence, we want to select only those users that continuously post about topics closely related to the keyword. However, a user may not always be using the same keywords when talking about a topic. Consider, for example, the topic “space development”. There are many keywords that relate to this topic (e.g. “NASA”, “SpaceX”, “Curiosity”, etc.). Users will likely not be mentioning any one of these all the time, and it is difficult and time-consuming to come up with an exhaustive search query. Another example is a user searching by a keyword “whaling”. This user likely also be interested in discussions about the hunting of dolphins and orcas as well. Some of these posts may go undetected when relying solely on keywords. Therefore, we aim to encapsulate multiple keywords that can be traced back to the same topical domain by using taxonomical background knowledge. Only if a user consistently posts about the same topic do we consider this user to be appropriate to follow.

There are several difficulties that need to be overcome, which are listed below. The solutions to these difficulties comprise the sequential steps of our proposed approach.

1. *User tweet collection*: this time, we need tweets from specific users rather than tweets related to a certain topic.
2. *Named entity extraction and classification*: we need some way to extract named entities from tweets, then use a background ontology to obtain class information for extracted keywords.
3. *Topic representation*: we need a way to represent the topic(s) of interest. Here, it is important to encapsulate just the right amount of generality. For example, when a user is interested in baseball, “sports” would be too generic, but “major league baseball players” might be too specific to determine topic consistency.
4. *Topic consistency checking*: once we have a topic representation, we need to determine whether a user posts about this topic consistently.

We build on our earlier work on domain-aware entity matching described in [11] and modify the approach to work within the context of Twitter. We take the output of the user recommendation method described in the previous section; for the top users, we derive a topic representation for (1) the tweets made during the period of tweet mining for the user relation recommendation, and (2) a selection of tweets made outside this period. We then compare the two topic representations and determine their similarity; if this is above a certain similarity threshold, we can say that this user posts about a topic consistently.

Our method thus applies a binary selection to the user ranking: starting at the best ranked user output in the previous step (user recommendation based on user relations), we decide whether this user is valuable or not based on a threshold of topic similarity across his timeline. We continue down the ranking, until we have found the target k of top-k users.

In the following sections we will explain each step of the approach in detail.

3.1. User Tweet Collection

We take the top ranked users and their tweets that are output from the recommendation step based on user relations. We call this set of tweets belonging to one user T_{mine}.

Additionally, for each user we collect earlier tweets from their timeline using the Twitter API. Although the Twitter API allows one to retrieve a maximum of 3200 tweets from a user's timeline, the subsequent entity extraction step that we perform on the tweets is computationally intensive, so we decide to collect a maximum of 500 tweets per user profile. We call this set of tweets $T_{profile}$. Here, we exclude tweets that were made during the mining period for the previous step, i.e. $T_{mine} \cap T_{profile} = \varnothing$.

3.2. Named Entity Extraction and Classification

First, we need to find entity mentions in a user's tweets and link them to a background ontology to obtain class information for these entities. As in [11], we use the off-the-shelf named entity recognition tool DBpedia Spotlight [5] to find entity mentions and create links to DBpedia [6] resources. DBpedia is the Semantic Web representation of Wikipedia, where each resource corresponds to a page in Wikipedia.

Before feeding the tweets to Spotlight for entity recognition, we strip each tweet of all Twitter entities (user mentions, hashtags, links, media). Furthermore, we strip the tweets of common, uninformative words such as "tweet", and internet slang, such as "lol", as these can throw off the entity recognizer. We keep stopwords, however, as Spotlight internally already ignores single common words as possible entity mentions, and entity mentions can contain stopwords (e.g. "Sea of Japan"). We also include retweets that a user makes into the set of tweets processed.

As mentioned, entity mentions are linked to DBpedia resources. There are several advantages to using DBpedia over Wikipedia directly:

1. DBpedia contains more class information: in addition to Wikipedia categories, DBpedia classifies resources (1) by its own, hand-crafted ontology; (2) by the YAGO [12] ontology, which is derived from a combination of the Wikipedia categories and WordNet [13]; (3) by links to corresponding resources in Freebase [7]. Each of these four types are term hierarchies, or *taxonomies*, with predefined sub-class/super-class relations. Therefore, we are not limited to the class directly associated with an entity, but can move "up" the taxonomy to consider more generic classifications as well.
2. DBpedia and its class taxonomies are easily queried using the SPARQL [3] query language. This circumvents the need to mine Wikipedia for the information that we need.

For this early work, we focus on using just the YAGO ontology, as the DBpedia and Freebase class hierarchies are rather sparse and generic (containing 359 classes and around 1000 classes respectively), while the Wikipedia category hierarchy is very large (nearly a million categories) and not strictly hierarchical, i.e. classes usually have more than one super-class, making it difficult to process properly. The YAGO ontology contains over 100,000 classes to a considerable degree of specialization, so that even very specific topics can be modeled. Furthermore, with a few select exceptions, each class has only one super-class, making it easier to process and navigate.

[5] http://spotlight.dbpedia.org

[6] http://dbpedia.org

[7] http://www.freebase.com

Each tweet of a user is fed to DBpedia Spotlight's *annotate* function, which searches the text for surface forms and returns the best matching DBpedia resources where they are found. Spotlight has two parameters that can be set: *confidence*, which is a threshold (between 0 and 1) for minimum similarity between the tweet text and candidate Wikipedia pages (based mainly on word co-occurrence) and *support*, which defines the minimum level of prominence for a given page, measured by the number of inlinks, i.e. hyperlinks pointing to this page from other pages in Wikipedia.

For each resource found, we collect the YAGO classes assigned to them, as well as the super-classes of these classes, and so on, up until the root of the hierarchy. For each individual resource, we only count unique class occurrences. For example, some person resource may be assigned both class "LivingPeople" and class "Artist", which both have class "Person" as a common super-class – in this case, we only count class "Person" once. Once all tweets of a given user have been processed, we combine all sets of classes that we gathered for each tweet, and count the total number of occurrence for each class, to obtain a weighted class taxonomy C.

$$C = \{(c, o) |\ c \text{ is a YAGO class, } o \text{ is its number of occurrence}\} \tag{11}$$

We build taxonomies for both tweet sets T_{mine} and $T_{profile}$, to obtain (YAGO) class taxonomies C_{mine} and $C_{profile}$.

3.3. Topic Representation

Next, we need to trim the sets of classes C_{mine} and $C_{profile}$ to obtain an accurate representation of a user's topic(s) of interest. Since entity recognition on informal tweets is far from perfect, we need to discover and remove any classes that have been linked to mismatched entities. Second, since we collect classes up to the root of the hierarchy, we need to filter out classes that are too generic and cover too broad of a topic.

3.3.1. Mismatch Removal

We rely on the law of large numbers to remove mismatched entities: as long as the accuracy rate of our entity recognizer is not so low as to be equivalent to a random selection of DBpedia resources for each possible entity mention in the text, then classes related to the topic being talked about by the user will statistically have the highest number of occurrence in the long run. Since DBpedia Spotlight has been shown to be able to attain an accuracy rate that is far better than a random selection [14], we assume that, given enough tweets of a user to process, classes related to his/her interests will appear the most in the collection.

We can therefore choose to remove classes with a particularly low number of occurrence, as these are the most likely to be mismatches. Since we cannot be sure exactly how many classes we collect (for example, since we collect classes up to the root of the hierarchy, more classes are collected for specific topics than for generic ones), we define a threshold for removal as a percentage $t\%$ of the total number of occurrence for all classes gathered for a user. Any class with a number of occurrence lower than this percentage will be removed.

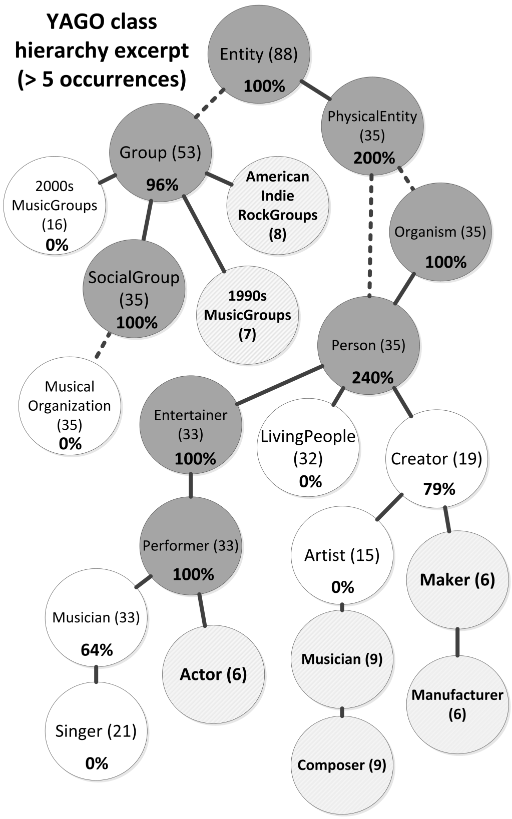

Figure 2. Example excerpt of a YAGO class taxonomy after filtering, with a mismatch removal cutoff of $t = 1\%$ (14 occurrences), and $p = 80\%$.

3.3.2. Filter Generic Classes

We also need a way to remove classes that are too broad and cover too many classes. We apply the algorithm described in our previous work [11]. That is, we filter out all super-classes where the sum of the numbers of occurrence of their *direct* sub-classes is greater than $p\%$ of the number of occurrence of the super-class. In other words, for every YAGO class $Super$, if

$$\sum_{i=1}^{n} occs(Sub_i) > \frac{p}{100} occs(Super), \tag{12}$$

where n is the number of direct sub-classes Sub_i of $Super$, and function $occs$ counts the number of occurrence of each class, then $Super$ is removed from the collection.

See figure 2 for an illustrative example of pruning a taxonomy with a dominant topic of “musical artists”. In this figure, class occurrence numbers are shown in parentheses. Dotted lines abstract some unimportant intermediate classes, and classes occurring 5 times or less are also not shown. The percentages are the proportions of direct sub-class occurrences to this class’ occurrences. Classes in light gray are discarded during mismatch removal; classes marked dark gray are discarded during generic class filtering. The white classes represent the final topic. As can be seen, most generic and unrelated classes are successfully removed. However, there is one particular YAGO class, *LivingPeople*, that is often assigned directly to a person entity, making it not possible to filter out using this algorithm. Since this is an exceptional class, and is too generic for our purposes, we exclude this class from the result manually.

3.4. Topic Consistency Checking

Finally, we need to compare the topic representations of the tweets collected in the user relation mining period and remaining tweets collected from a user's timeline, in order to determine whether or not a user posts about the same topic consistently. We have two sets C_{mine} and $C_{profile}$ of classes c_i, and each class has a number of occurrence $occs(c_i)$. We devise a simple way to determine the similarity between the two sets.

First, we take the smallest of the two sets, in terms of the total number of class occurrences. For simplicity, we will assume this to be C_{mine} as this is true in virtually every case. We then count for each class c_i in C_{mine} how many occurrences of this same class c_i are also covered in $C_{profile}$. Finally, we normalize by dividing the total number of common occurrences found by the total number of class occurrences in C_{mine}. Formally:

$$s = \frac{\sum\limits_{c_i \in C_{mine} \cap C_{profile}} \min\{occs(c_i)|c_i \in C_{mine}, occs(c_i)|c_i \in C_{profile}\}}{\sum\limits_{c_i \in C_{mine}} occs(c_i)}. \qquad (13)$$

The *taxonomical similarity score* s that is obtained is a number between 0 and 1. We can subsequently define a threshold parameter θ_s between 0 and 1 to determine whether two taxonomies are similar enough to consider the user to be posting about the same topic consistently. We apply a binary selection on the user ranking, starting with the highest ranked user. For each user, if s is above θ_s, the user is kept in the ranking. If s is below θ_s, the user is removed from the ranking. We continue until the desired top-k ranked users are obtained, or until we reach a stop condition (e.g. process only the top 50 users).

4. Evaluation

In this section we report on results obtained with the devised approach. First, we explain the methodology employed for our experiments in section 4.1. That is, we describe the keywords used for the experiments, the parameter settings used for each of the steps in our approach, how we evaluate results and which other methods we compare the results to. Section 4.2 details the results that were obtained, and we finish the evaluation with a discussion of the results, in section 4.3.

4.1. Methodology

We build on the results from our previous work in [10] and evaluate results with the additional step of user selection based on taxonomical analysis. For this work, however, we have focused on English-language tweets rather than Japanese-language tweets, so we use different keywords than before.

4.1.1. Keywords Mined

We have mined the Twitter search API over the course of 5 days, from Dec. 15th, 2012 to Dec. 19th, 2012. We gathered all tweets that contained any of the following keywords: *mars rover* $\cup$ *curiosity rover*, *malware*, *nuclear power*, and *genetically modified*. For *mars rover*, we consider tweets concerning Mars or space probes in general relevant to the topic as well. For *malware*, we also consider viruses, software vulnerabilities, etc. to be related. We feel that this reflects a real-world scenario, as a user searching for "malware" will likely be using this term as a catch-all for cyber security related topics, and not just the fairly narrow subject of "malicious software". For *nuclear power*, we aim to cover topics related to nuclear energy. Keyword *genetically modified* is used to catch topics related to genetically modified organisms (GMOs) and food. The evaluation will be performed on these four keywords separately.

4.1.2. Parameter Settings

There are a variety of parameters to set for the methods and tools that we use. For calculating the TURKEYS score in the first step of the approach, we need to define how to weigh tweet count score TC vs. user influence score UI (equation 1). We will fix the weight w to 0.5, so that both scores are weighed equally, as this weight was shown to perform the best on average in previous experiments [10].

When applying DBpedia Spotlight for entity recognition in tweets, we experimented with different values for its parameters. Recall that we can set *confidence* (similarity between the surrounding text of an entity mention and candidate Wikipedia page) and *support* (minimum number of inlinks to the candidate Wikipedia page). By trial and error we found that a confidence threshold of 0.3 gave a good balance between precision and recall, although a more in-depth analysis is needed to find an optimal value. We keep support at 0, as we want to be able to detect more obscure entities as well.

For the pruning of classes, described in section 3.3, we need to set a parameter t as a cutoff threshold for mismatch removal, and a parameter p for the filtering of generic classes. We set t to 0.1% of the total number of class occurrences, and p to 100%, meaning there should be at least as many direct sub-class occurrences as the number of occurrence of their super-class as a condition to remove this super-class. Again, experimentation showed that these values resulted in reasonable topic representations (not too many or too few classes, not too broad or too specific classes), but a more detailed investigation is needed in order to find optimal values.

Lastly, we need to set the similarity threshold θ_s to decide tweeting consistency of users for the final taxonomical analysis step. Given that the evaluation of this step is the focus of this work, we will experiment with different values to determine which provides the best result on average.

4.1.3. Evaluation Metric

As a measure of evaluation for the results, we will use the Discounted Cumulative Gain (DCG) [15]. First, we check the ranked Twitter users obtained by the user relation analysis step manually and assign a *relevance score* to each user. The relevance score can be either 0, 1 or 2. These numbers are defined as follows:

0: This user is irrelevant; either the user does not have tweets related to the topic outside of T_{mine}, or the user is a spammer or advertisement bot.

1: This user is somewhere in between; in this category we include users that post some tweets related to the topic, but also a considerable number of unrelated tweets, or users that post topic-related tweets with a specific goal in mind that may only be relevant to a select few users (e.g. a bot posting job offers for nuclear-related work when searching for keyword “nuclear power”).

2: This user is very relevant; most of the user’s tweets are directly related to the keyword we searched for.

Subsequently, the DCG is calculated according to the formula

$$\mathrm{DCG_k} = rel_1 + \sum_{i=2}^{k} \frac{rel_i}{\log_2(i)}, \tag{14}$$

where k means the top-k ranked users that we consider in the evaluation, and rel_i is the relevance score assigned to the user at rank i.

We want to evaluate performance based on several different ranking sizes k; we check performance for $k = 5$, $k = 10$ and $k = 20$. Since we will obtain different $\mathrm{DCG_k}$ scores for different k, we need to use the normalized version of the DCG, nDCG:

$$\mathrm{nDCG_k} = \frac{DCG_k}{IDCG_k}. \tag{15}$$

Here IDCG refers to the *Ideal* DCG, i.e. the maximum possible DCG until position k:

$$\mathrm{IDCG_k} = 2 + \sum_{i=2}^{k} \frac{2}{\log_2(i)}. \tag{16}$$

4.1.4. Comparison Methods

As mentioned above, we will compare the taxonomical analysis approach with different values for θ_s to each other, as well as to several baseline approaches to showcase the merit of the proposed method over other methods. The following baselines are applied:

Tweet count score (TC): Rank users simply based on the amount of tweets that contained the keyword they posted.

User influence score (UI): Rank users based on the user influence score described in section 2.

TURKEYS score: Rank users based on the TURKEYS score from equation 1.

Taxonomical analysis is performed on top of the user ranking based on TURKEYS scores. We try $\theta_s = \{0.1, 0.2, 0.3, 0.4, 0.5\}$. Note that the higher the similarity threshold θ_s, the higher the number of users that we discard from the ranking, meaning we need to process more than just the top 20 users even in the case of $k = 20$. We decide to process at most the top 50 users. Usually, this does not lead to a sufficient amount of resulting users for higher thresholds, so we stop at 0.5. Note, however, that even for $k = 5$, we found no performance improvements beyond $\theta_s = 0.5$ given the keywords that we experimented with.

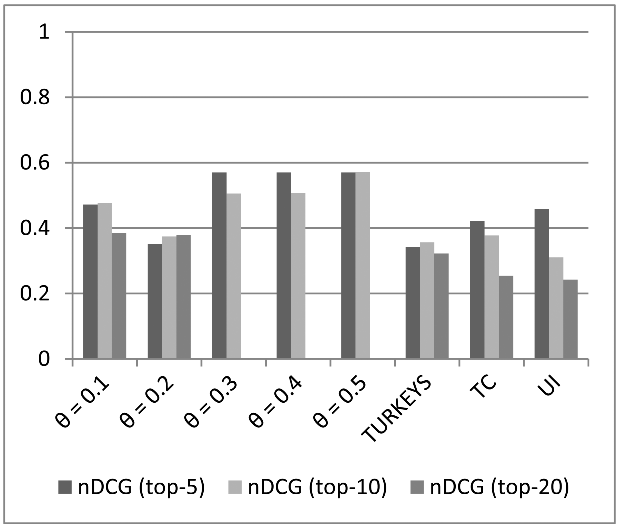

Figure 3. Results for keyword *mars rover* ∪ *curiosity rover*.

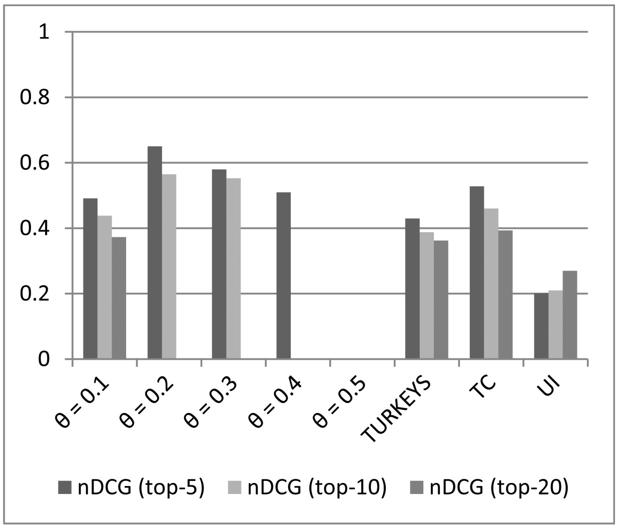

Figure 4. Results for keyword *genetically modified*.

4.2. Experimental results

In this section we list the results for the four keywords that we investigated. In case not enough users could be gathered for the taxonomical analysis approach, the result is left empty in the charts.

Figure 3 shows the result for keyword(s) *mars rover* ∪ *curiosity rover*. We can see a significant improvement in nDCG scores for the taxonomical analysis approach, especially for higher similarity thresholds. Although for a top-20 ranking, we can no longer

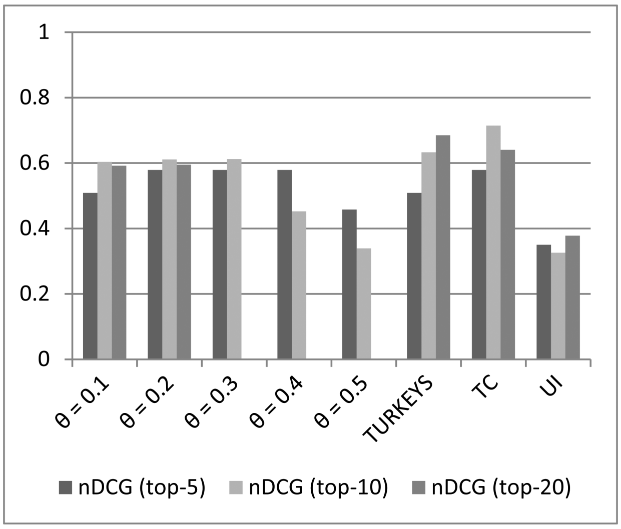

Figure 5. Results for keyword *malware*.

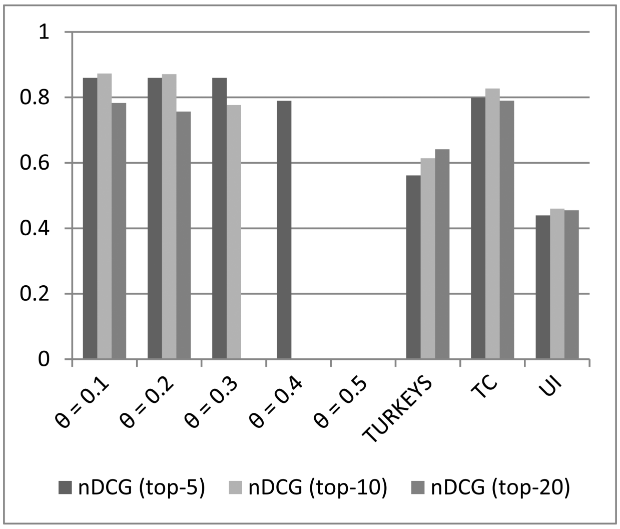

Figure 6. Results for keyword *nuclear power*.

obtain a sufficient amount of users past $\theta_s = 0.2$, even $\theta_s = 0.1$ and $\theta_s = 0.2$ already outperform the baselines.

For the results of keyword *genetically modified* in figure 4, we see a peak in performance at a θ_s value of 0.2. For this keyword, it is more difficult to obtain a sufficient amount of users, as the topic of genetically modified foods and animals is much less focused than the mars rover, making it harder to concretely define a topic covering and determine whether two coverings are similar. Again, there are significant improvements to be seen for $k = 5$ and $k = 10$. For $k = 20$, however, tweet count performs best.

Results for keyword *malware* are shown in figure 5. Performance of the taxonomical approach is equal to the tweet count score for $k = 5$, and inferior to tweet count score and TURKEYS for $k = 10$ and $k = 20$. We found that the topic of malware is consistently captured into the same classes (most often occurring classes are *Software*, *ComputerSecuritySoftwareCompanies* and *ComputerCrimes*), making it easy to end up with enough users even at high thresholds. We discuss an explanation for the poor performance for this keyword in the next section.

Lastly, figure 6 shows the result for keyword *nuclear power*. We can see that all methods perform well in terms of absolute nDCG score. For $k = 5$ and $k = 10$, the proposed approach with θ_s at 0.1 or 0.2 is the best performer. For $k = 20$, tweet count score performs best again, although $\theta_s = 0.1$ comes very close. For this keyword, it is again difficult to obtain enough users for $k = 10$ and $k = 20$ at higher thresholds, for the same reasons as keyword *genetically modified*.

4.3. Discussion

For most of the keywords analyzed and top-k values evaluated, our results show an improvement over the baseline approaches. Keyword *mars rover*, focusing on a very specific topic shows the most significant improvement. There are two main reasons for this. First, a specific topic is easier to model consistently, as the same specific set of classes tend to be associated to related entities (most commonly occurring classes are *NASAProbes* and *TerrestrialPlanets*, which would never appear in a more general context). Second, due to the popularity of the rover among the general public, many users not truly interested in Mars may mention the rover in passing when news concerning it is released, but are not consistently talking about the rover or Mars-related topics, making approaches that do not take tweeting consistency into account select the wrong users.

For “genetically modified” and “nuclear power”, improvements are less significant but still visible, as these topics are more generic and thus more difficult to concretely define in terms of Wikipedia classes.

For “malware”, performance is equal to the best baseline for $k = 5$, but worse for $k = 10$ and $k = 20$, where tweet count and TURKEYS outperform it. This can be explained by the fact that a large portion of the top users for “malware” seem to be either news aggregation bots that automatically post malware related topics (such as discoveries of new viruses, exploits or security holes), or users that automatically retweet these bots. These users typically have very high tweet counts. Since the taxonomical analysis approach is not perfect, it is incorrectly excluding some valuable users from the ranking.

Compared to the experiments performed in our previous work, on average the results obtained with the TURKEYS ranking are somewhat worse. This is because the TURKEYS mechanism relies on the existence of a tightly knit user community and the mentions and retweets among them. This works particularly well when processing Japanese twitter users who are concentrated in a single location, use the same language and share the same culture, but it appears to be less effective on a global scale.

5. Conclusion

In this paper, we demonstrated the benefits of applying semantic analysis of timelines to user recommendation on Twitter. We first created an initial ranking of potentially valuable users by analyzing user relations, and ranking users according to a score based on a combination of tweet count and user influence (TURKEYS). We then took the top users from this ranking and analyzed their timelines to create topic taxonomies by matching terms in their posts to a background knowledge base (Wikipedia). In case topics were not steady across the user's timeline, we judged this user not to post consistently about these topics and excluded them from the ranking.

Our evaluation showed that, depending on the keyword, applying an extra binary selection to the ranking of users allows us to obtain significantly more accurate user recommendations. We found that our method is particularly useful for specific topics that are easily modeled in terms of YAGO classes. For more generic keywords, improvements are less significant, but still visible.

5.1. Future work

As this is still an early version of our work, there are many future additions and modifications that we have planned in order to improve the effectiveness of the approach.

A limitation of our approach is that, while we can tell whether a user posts about a topic consistently or not, we cannot tell if this is the correct topic that relates to the keyword we searched for. One possible future direction would be to focus on an active user looking for recommendations rather than a keyword, assuming the that the more similar a user is to the active user, the more interesting this user is for recommendation. We can build topic definitions for two users and compare their similarity, and recommend a target user if it is similar enough. This way, we do not need to know about the topic itself in order to make a good recommendation.

For entity recognition, we have used DBpedia Spotlight for convenience, as it is an entity recognizer that links entity mentions to Wikipedia pages. However, Spotlight has been shown to perform poorly compared to other entity recognizers in one study [16]. An investigation of alternative entity recognizers is needed, as more accurate results may be obtained.

There are many parameters to set for the various tools and algorithms that we use in our approach. In this work, we experimentally chose values for these parameters that individually provided decent results. However, in reality these parameters are not independent of each other, and also dependent on the topic. Hence, we need to investigate how changing the parameters will influence the result, and how we can tune the parameters to obtain the best results.

References

[1] TechCrunch. Twitter passes 200m monthly active users, a 42% increase over 9 months. http://techcrunch.com/2012/12/18/twitter-passes-200m-monthly-active-users-a-42-increase-over-9-months/.

[2] CNET News. Twitter hits 400 million tweets per day, mostly mobile. http://news.cnet.com/8301-1023_3-57448388-93/twitter-hits-400-million-tweets-per-day-mostly-mobile/.

[3] W3C. SPARQL Query Language for RDF. http://www.w3.org/TR/rdf-sparql-query/.

[4] Jianshu Weng, Ee-Peng Lim, Jing Jiang, and Qi He. TwitterRank: Finding topic-sensitive influential Twitterers. In *3rd ACM International Conference on Web Search and Data Mining*, pages 261–270, 2010.

[5] John Hannon, Mike Bennett, and Barry Smyth. Recommending Twitter users to follow using content and collaborative filtering approaches. In *4th ACM Conference on Recommender Systems (RecSys '10)*, pages 199–206, 2010.

[6] Zareen Saba Syed, Tim Finin, and Anupam Joshi. Wikipedia as an ontology for describing documents. In *2nd International Conference on Weblogs and Social Media*, pages 136–144, 2008.

[7] Matthew Michelson and Sofus A. Macskassy. Discovering users' topics of interest on Twitter: a first look. In *4th Workshop on Analytics for Noisy Unstructured Text Data*, pages 73–80, 2010.

[8] Sofus A. Macskassy and Matthew Michelson. Why do people retweet? anti-homophily wins the day! In *5th International AAAI Conference on Weblogs and Social Media*, pages 209–216, 2011.

[9] Makoto Nakatsuji, Yasuhiro Fujiwara, Toshio Uchiyama, and Hiroyuki Toda. Collaborative filtering by analyzing dynamic user interests modeled by taxonomy. In *11th International Semantic Web Conference*, pages 361–377, 2012.

[10] Tomoya Noro, Fei Ru, Feng Xiao, and Takehiro Tokuda. Twitter user rank using keyword search. In *22nd European-Japanese Conference on Information Modelling and Knowledge Bases*, pages 48–65, 2012.

[11] Kristian Slabbekoorn, Laura Hollink and Geert-Jan Houben. Domain-Aware Ontology Matching. In *11th International Semantic Web Conference*, pages 542–558, 2012.

[12] Fabian Suchanek, Gjergji Kasneci and Gerhard Weikum. Yago: A large ontology from wikipedia and wordnet. In *Web Semantics: Science, Services and Agents on the World Wide Web*, pages 203–217, 2008.

[13] Christiane Fellbaum. WordNet. In *Theory and Applications of Ontology: Computer Applications*, pages 231–243, 2010.

[14] Pablo N. Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. DBpedia Spotlight: Shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems*, pages 1–8, 2011.

[15] Kalervo Jarvelin and Jaana Kekalainen. Cumulated Gain-Based Evaluation of IR Techniques. In *ACM Transactions on Information Systems*, pages 422–446, 2002.

[16] Guiseppe Rizzo and Raphaël Troncy. Nerd: evaluating named entity recognition tools in the web of data. In *Workshop on Web Scale Knowledge Extraction (WEKEX'11), Bonn, Germany*, pages 1–16, 2011.