Resolving Name Conflicts for Mobile Apps in Twitter Posts

Sangaralingam Kajanan, Ahmed Shafeeq Bin Mohd Shariff, Kaushik Dutta, Anindya Datta

▶ **To cite this version:**

Sangaralingam Kajanan, Ahmed Shafeeq Bin Mohd Shariff, Kaushik Dutta, Anindya Datta. Resolving Name Conflicts for Mobile Apps in Twitter Posts. Working COnference on Shaping the Future of ICT Research , Dec 2012, Tampa, FL, United States. pp.3-17. hal-01495225

HAL Id: hal-01495225
https://hal.science/hal-01495225

Submitted on 28 Apr 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Resolving Name Conflicts for Mobile Apps in Twitter Posts

Sangaralingam Kajanan[1], Ahmed Shafeeq Bin Mohd Shariff[2], Kaushik Dutta[3], and Anindya Datta[4]

School of Computing,
National University of Singapore, Singapore
[1]skajanan@comp.nus.edu.sg
[2]ahmedshafeeqm@gmail.com
[3,4]{dutta,datta}@comp.nus.edu.sg

Abstract. The Twitter platform has emerged as a leading medium of conducting social commentary, where users remark upon all kinds of entities, events and occurrences. As a result, organizations are starting to mine twitter posts to unearth the knowledge encoded in such commentary. Mobile applications, commonly known as *mobile apps*, are the fastest growing consumer product segment in the history of human merchandizing, with over 600,000 apps on the Apple platform and over 350,000 on Android. A particularly interesting issue is to evaluate the popularity of specific mobile apps by analyzing the social conversation on them. Clearly, twitter posts related to apps are an important segment of this conversation and have been a main area of research for us. In this respect, one particularly important problem arises due to a name conflict of mobile app names and the names that are used to refer the mobile apps in twitter posts. In this paper, we present a strategy to reliably extract twitter posts that are related to specific apps, but discovering the contextual clues that enable effective filtering of irrelevant twitter posts is our concern. While our application is in the important space of mobile apps, our techniques are completely general and may be applied to any entity class. We have evaluated our approach against a popular Bayesian classifier and a commercial solution. We have demonstrated that our approach is significantly more accurate than both of these. These results as well as other theoretical and practical implications are discussed.

Keywords: Affinity, Microblogs, Twitter, Mobile Apps, Filter

1 Introduction

The Twitter platform has emerged as a leading medium of conducting social commentary, where users remark upon all kinds of entities, events and occurrences. As a result, organizations are starting to mine twitter posts to unearth the knowledge encoded in such commentary. Applications that can benefit from such knowledge discovery are many: trending topic discovery, sentiment analysis of consumer products and gauging public reaction to political campaigns to

name a few. A key requirement of a majority of such applications is the timely identification of twitter posts related to specific entities of interest, like products, persons or events. Such identification is well understood to be difficult due to a number of reasons, including (a) real-time discovery of relevant twitter posts given their massive rate of generation [13, 20], (b) handling multi-lingual posts and (c) interpreting highly cryptic tweets, driven by brevity constraints [7].

In this work, we will be exploring this problem, i.e., the real-time identification of microblog postings that contain references to pre-specified entities of interest. For example, we might wish to identify tweets that talk about the movie "Harry Potter and the Deathly Hallows: Part 2".

Two key problems that need to be addressed to perform such identification arise due to (a) the practice of *aliasing* entity names and (b) *naming conflicts* that arise between the entity of interest and other objects. *Aliasing*, driven by the need to conserve space, is the practice of using a subset of complete entity names (such as "Harry potter", for "Harry potter and the deathly hallows: Part 2") to refer to the entity. Clearly, if the identification system was unaware of such aliasing, it would perform poorly. The second problem, i.e., *naming conflicts* arise from semantic overloading of entity names, and is a common problem in the general search area. For instance, a film historian seeking information about the movie "ten commandments" (a phrase with wide connotations) will find that a simple search with just the movie title yields an enormous amount of information not related to the movie. However, adding contextual clues to the title (e.g., "ten commandments movie", "ten commandments de mille", "ten commandmentsheston") yield high quality results [11, 24, 6]. In most cases (such as in regular internet search), the user performing the search is aware of additional context clues (such as the fact Charlton Heston played the lead role in Ten Commandments) and can easily expand the search term.

In Twitter, the aliasing and entity name conflict problems assume special significance as the brevity of twitter posts precludes the usage of traditional context clues. While this problem arises while searching any entity type, we ran into this issue particularly often in the domain of mobile applications, as we explain below.

Mobile applications, commonly known as *mobile apps*, are the consumer software for smart-devices, such as those running on Google's Android [10] or Apple's iOS [3] platforms. They represent the fastest growing consumer product segment in the history of human merchandizing [30, 17, 16], with over 617,000 apps on the Apple platform and over 357,000 on Android. Their growth rate is astonishing, with nearly 3500 apps being added to the Android market and Apple app stores every day.

The importance of apps is underscored by the fact that the future of the mobile strategies of both Apple and Google are heavily dependent upon who wins the mobile app wars [34, 21, 32]. With this backdrop, there is tremendous academic as well as commercial interest in mobile apps.

An interesting feature about mobile apps is their virality - most successful apps (e.g., Angry Bird, Talking Tom, Flashlight etc.) gain popularity not by

explicit outbound marketing, but rather, through viral word-of-mouth spread. Consequently, social media plays a significant role in the success of mobile apps. Given this context, we have been trying to to evaluate the popularity spread of mobile apps by analyzing the social conversation on them. Twitter posts related to apps are an important segment of this conversation. However, when we tried to extract twitter posts related to specific apps we discovered that it was a difficult task, due, to the aliasing and name conflict problems. For instance when searching for tweets discussing the popular iPhone app titled "Movies by Flixster with Rotten Tomatoes - Free", we found that tweeters typically aliased this app simply as "Flixster". We then tried to simply search for tweets containing the term "Flixster". However, even this proved to be challenging as we discovered that "Flixster" is overloaded – it could refer to both the app or the website (`http://www.flixster.com/`) – its was not easy to discard the tweets referring to the website and retain those referring to the app. We found these issues to be common across many apps. Clearly, unless these issues are addressed meaningfully, it would be impossible to perform the core task, i.e., extracting tweets referring to apps.

In this paper, we present a strategy to reliably extract twitter posts that are related to specific apps, overcoming the aliasing and name conflict issues discussed above. While we were motivated by mobile apps, our techniques are completely general and may be applied to any entity class.

In the next section, we review related literature. In Section 3, we describe our solution approach. Section 4 experimentally demonstrates the efficacy of our techniques and in Section 6 we conclude the paper.

2 Related Work

The work related to this research may be classified as commercial or academic, and we discuss each in turn.

First let us address commercial solutions. *Tweet filter* [31] is a browser plugin that runs on top of `twitter.com`. Using *Tweet filter* we can filter tweets by matching usernames, keywords, phrases or source. *Filter Tweets* [9] is a browser based script for filtering tweets by a specific topic and it works only with the new Twitter version. One of the features in *Filter Tweets* is filtering tweets that contain a set of terms. *Social Mention* [26] is a social media search and analysis platform that aggregates user generated content from more than 100 social media web sites including: Twitter, Facebook, FriendFeed, YouTube, Digg, Google+ etc. It allows users to easily track and measure what people are saying about a person, company, product, or any topic across the web's social media landscape in real-time. Social Mention provides an API [27] to filter the user generated contents based on the given keywords from the popular social medias mentioned above.

All of the above-mentioned commercial solutions have similar characteristics. First, all of them work based on exact keyword match, however as described in the Section 1, mobile apps are seldom referred with the full name in the twitter

posts, so it will be difficult, if not impossible, to find twitter posts related to mobile apps using any of three. In other words, these solutions do not address the aliasing or name conflict problems. We will demonstrate this experimentally in Section 4.

Let us now look at some academic research of relevance to our problem. Inherently, at the end, our aim is to classify each twitter post as whether it is related to a mobile app or not. Thus, at a high level our problem resembles a classification problem. In this respect the Bayesian classification technique is worth mentioning. The study titled "An Evaluation of Statistical Spam Filtering Techniques" [33] evaluates five supervised learning methods such as "Naive Bayes","Maximum Entropy model","Memory based learning", "Support vector machine"(SVM) and "Boosting" in the context of statistical spam filtering. They have studied the impact of different feature pruning methods and feature set sizes on each learner's performance using cost-sensitive measures. This study has observed that the significance of feature selection varies greatly from classifier to classifier. In particular,they found SVM, AdaBoost, and Maximum entropy model to be the top performers in this evaluation, sharing similar characteristics: not sensitive to feature selection strategy, easily scalable to very high feature dimension, and good performances across different data sets. In contrast, Naive Bayes [14, 19], a commonly used classifier in spam filtering, is found to be sensitive to feature selection methods on small feature sets, and fails to function well in scenarios where false positives are penalized heavily. Many other studies [1, 22, 25] have studied the popularity of "Naive Bayes" [14, 19] in anti spam research and found that it outperforms keyword based filtering, even with very small training corpora.

The paper "Short Text Classification in Twitter to Improve Information Filtering" by Sriram et al. [29] has proposed an intuitive approach to determine the class labels and set of features with a focus on *user intentions* on Twitter. Their work classifies incoming tweets into categories such as News (N), Events (E), Opinions (O), Deals (D), and Private Messages (PM) based on the author information and features within the tweets. Their work is based on sets of features which are selected using a greedy strategy. Sriram et al.'s work experimentally shows that their classification out-performs the traditional "Bag-Of-Words" strategy. Unlike this resarch, our approach does not rely on supervised learning , thus we do not have the overhead of feature selection and manual labeling. In addition, we can classify a tweet as referring to any mobile app out of an arbitarily sized set of apps, unlike Sriram et al, who need a predefined exact number of categories into which they perform the classification.

In addition to classification of short text messages, integrating messages with meta-information from other information sources such as Wikipedia and WordNet [4, 12] are also relevant. Sankaranarayanan et al. [23] introduce TweetStand to classify tweets as news and non-news. Automatic text classification and hidden topic extraction [29, 4] approaches perform well, when there is meta-information or the context of the short text is extended with knowledge extracted using large collections. This does not apply in our case.

Currently, there are about 1 millions mobile apps in the market [18]. To classify each twitter post as related to one or more of these apps, or not at all related to any of the mobile apps, will require equivalent number of classes, i.e. 1000,000 classes in the classification approach. Such a large number of classes are impossible to handle using existing machine learning and classification techniques such as SVM [5] and Artifical Neural Networks(ANN) [8]. Therefore, instead of applying a classification approach, in this paper, we address the problem at hand using corpus based data driven approach. In the next section, we first describe the intuition behind our approach and then explain the algorithm in detail.

3 Solution Approach

We will first provide the intuition behind our approach and then delve into details. A precise statement of our problem is as follows: **given app *A*, find twitter posts that refer to *A***. Our approach has two steps namely "Alias Identification" and "Conflict Resolution".
1. First we need to discover what alias is commonly used by users to refer to app A as names are often abbreviated in the length-restricted twitter posts (140 characters). For instance, the popular iTunes app "Doodle Jump - BE WARNED: Insanely Addictive", is commonly referred to in twitter posts as "Doodle Jump". We call this step the *Alias Identification* step.
2. After alias identification, we need to resolve name conflicts, i.e.,make sure that the twitter posts we find refer to the app and not to other objects with the same name. One particularly ripe area for conflicts is between mobile apps and a regular web applications. To see this consider the popular iPhone app titled "Movies by Flixster with Rotten Tomatoes - Free". It turns out that this app is commonly referred to as "Flixster". However, a twitter post containing the term "Flixster" might be referring to the app, *or, to the highly popular sister website*. We are of course interested in the popularity of the app. Similar issues arise in the case of the Facebook app, or the Google Translate app. We refer to this phase as *Conflict Resolution*.

3.1 Intuition behind the Alias Identification Phase

To identify the appropriate alias of an app with name A, we find the sub phrase contained in A that is the most *meaningful* and *unique*. Such meaningfulness and uniqueness (described below) is judged in the context of a *Social Media Corpus* (SMC) we have constructed by lexical analysis of a vast amount of data gathered from Social Media Avenues such as Twitter, Facebook and the user comments awarded to apps in the native app stores.

Meaningfulness: Intuitively, meaningfulness refers to the semantic content of a phrase. For instance, in the context of the app title "Doodle Jump - BE WARNED: Insanely Addictive", the reader can easily see that the sub phrase

"Doodle Jump" is more meaningful than, say "Be Warned", or "Insanely Addictive". From an information theoretic perspective, meaningful n-grams will exhibit higher collocation frequencies relative to individual occurrence frequencies of the constituent 1-grams. We describe this ratio as *Affinity*. Formally, we define the *Affinity* of a word phrase P as, $Affinity(P) = \frac{f(P)}{min_{\forall w_i \in P}(f(w_i))}$, where $f(P)$ is the frequency of phrase P in a corpus and $min(f(w_i))$ is the minimum frequency across the words in a phrase P in the SMC. For the app name "Doodle Jump - BE WARNED: Insanely Addictive!", Table 1 shows the frequencies and affinity measurement of word phrases, which formally identifies the word phrase "Doodle Jump" as more meaningful than others.Note that the table does not show all phrases whose affinities are measured for comparison. For a particular n ($n = 1 \ldots N$, where N is the number of words in the name of the application as the respective mobile app store), we take all n-grams from left to right beginning with the first word and stopping at the $(\mathrm{N} - \mathrm{n} + 1)^{\mathrm{th}}$ word.

Phrase	$f(P)$	**Affinity**
Doodle Jump	99	$99/1456 = 0.07$
Be Warned	8231	$8231/138408 = 0.06$
Insanely Addictive	18	$18/5315 = 0.003$

Table 1. Affinity Measure

Uniqueness: The meaningfulness property, while useful, is by itself not adequate for our purposes. To see this consider the following. Let us hypothetically assume (perhaps due to sampling biases while corpus creation) that the sub phrase "insanely addictive" is as (or more) meaningful than "Doodle Jump". Our system, using meaningfulness alone, would then judge "insanely addictive" as the best alias for the app "Doodle Jump - BE WARNED: Insanely Addictive" – a patently bad choice (as "insanely addictive" might be used in the context of many other apps). The *uniqueness* property (used in tandem with meaningfulness) prevents this mis-judgment, by *ensuring that the selected alias is used often in the correct context, but rarely in alternate contexts*. Furthermore, affinity does not apply to 1-grams and we cannot compare affinity directly to the uniqueness property we shall define. As such, this step will help to choose between the most meaningful n-gram phrase and all other 1-grams such that the end result is both highly meaningful and unique. Thus, to quantify uniqueness, we make a slight modification to the well-known IR notion of *inverse document frequency (idf)* [28] for a word or word phrase. The traditional idf is defined as: $idf(P) = log_2 \frac{|D|}{1+df(P)}$, where $|D|$ is the total number of documents in the corpus and $df(P)$ is the document frequency of phrase P, namely the number of documents that contain phrase P in corpus. We have modified this expression to: $idf(P) = log_2 \frac{1}{1+tcount(P)}$, where $tcount(P)$ is the frequency of P as recorded by Twitter in the target time interval T and we have done away with $|D|$ because for all phrases, the number of documents in the corpus (in this case, number of tweets in Twitter's database) within the target time interval T will be the same. Since we're looking for the highest $idf(P)$ it does not matter what $|D|$ actually

is. We retrieve phrase level $tcount(P)$ directly from Twitter. For instance, the *idf* of the phrase "Doodle Jump" in our corpus is 18.28 but the *idf* values of "Doodle" and "Jump" are 14.2 and 7.6 respectively. Therefore, "Doodle Jump" has more uniqueness and rarity than the individual terms "Doodle" and "Jump".

3.2 Intuition behind the Conflict Resolution Phase

The alias identification step ensures that the best alias is selected, but does not guarantee that this alias will not have conflicts with other object names, as the "Flixster" example above illustrated. In this phase we attempt to minimize this error. The core idea is as follows: Assume an alias, say S, is context-overloaded. Our objective is to identify the overloaded aliases and then rerun the core tweet search by using a new search term that consists of the alias and a few contextual terms that disambiguate the search(e.g., "flixster + iPhone"). The additional context raises the probability that the retrieved tweet is talking about the mobile app domain.

3.3 Details of Alias Identification Phase

As discussed in section 3.1, in this step we discover the alias A' of an app A, based on its meaningfulness and uniqueness values. This procedure is shown in Table 2 from steps 1-6. Here, step 1 extracts all sub phrases from A (using a parser [2]), and computes affinities of each sub phrase in step 2. Subsequently, in step 3 we extract the most meaningful (highest affinity) phrase. This phrase is then subjected to a uniqueness test in step 4 by comparing its *idf* to the *idf*s of all 1-grams derived from A. Based on this test, the selected alias A' is returned.

After alias identification, the tweets containing this alias are considered *Legitimate*, while disqualified posts are marked as *Spam*. The legitimate tweets are then subjected to the conflict resolution phase, described below, to ensure that these refer to the app, and not to other objects with similar labels.

3.4 Details of Conflict Resolution Phase

To ensure that legitimate tweets refer to mobile apps and not to alternate objects, we design a classification mechanism where we first identify dual purpose aliases (e.g., Flixster, Facebook) and then incorporate additional context. More specifically, we run the k-means clustering algorithm [15] on all the *idf* values of the aliases A' with $k = 2$, i.e. two clusters. The two initial mean points for each cluster are the lowest and the highest *idf* values across all aliases. This is shown in Table 2 in step 7. The result of the k-means classification will be two sets of aliases, a *high-idf* cluster and a *low-idf* cluster. An example follows.

After partitioning the top ranked Android apps based on the *idf* values of their aliases, we found "paper toss","pocket god", "words with friends","ebay

mobile","pandora radio" and "espn scorecenter" belonged to the high-*idf* cluster, indicating they exist only in mobile app domain. Conversely, "flixster", "google earth","skype" "facebook","kindle", "bible", "flashlight", "netflix", "backgrounds" and "translator" are aliases with low *idf* values, indicating these names are used both in mobile apps and in other domains, such as web applications.

For the aliases with higher *idf*, we accept their associated twitter posts, as there is a very high probability that the post is referring to the mobile app.

For the aliases with the low *idf* values, we incorporate additional filtering mechanisms, by adding additional keywords like "app","Android","iPhone", "iPod", "Apple" and "iPad". Tweets containing any of these additional keywords are considered relevant(*Legitimate*), otherwise it is categorized as *Spam*.

1. Generate set of all word phrases $\mathcal{C}$ of length 2, 3 or 4 of the app name A. For example, for the app name "Doodle Jump - BE WARNED: Insanely Addictive!", some of the collocates will be "Doodle Jump", "Be Warned" and "Insanely Addictive".
2. Compute $Affinity(C_i)$ for each word phrase $C_i \in \mathcal{C}$ as derived in Step 1. For example, $Affinity$("Doodle Jump") = 0.07, $Affinity$("Doodle Jump Be") = 0.00068 and $Affinity$("Be Warned") = 0.06.
3. Identify the word phrase C_i^{max} that has the highest value of $Affinity(C_i)$. In our example, the highest value is for $Affinity$("Doodle Jump") = 0.07, thus C_i^{max} = "Doodle Jump".
4. Compute the *idf* for C_i^{max} and all one gram word of the name A. In our example, *idf*("Doodle Jump") = 18.28, *idf*("Doodle") = 14.2, *idf*("Jump") = 7.6, *idf*("Warned") = 7.79 and so on.
5. Identify the word phrase that has the highest *idf* as computed in step 4. In example, "Doodle Jump" has the highest *idf*.
6. Return the word phrase identified in Step 5 as the alternate app name A' of the app A.
7. After running steps 1-6 for all app names, we run k-means clustering on the *idf* values of the word phrases returned in step 6 with a k value of 2 and the initial means to be the highest *idf* and lowest *idf* values in the corpus respectively. This will yield two clusters, one that is high-idf and one that is low-idf.
8. For all word phrases that are part of the low-idf cluster, append extra context keywords before querying the tweet database. For all words phrases that are part of the high-idf cluster, we can use the word phrases "as is".

Table 2. Algorithm for retrieving exact query phrase to use on the tweet database to ensure high relevance

4 Experimental Results

In this section we demonstrate the efficacy of our approach, which we will refer to as TApp. The idea is to evaluate the quality of the *legitimate tweets* produced – if a tweet refers to the appropriate mobile app, the result is correct, otherwise, for that particular tweet, our procedure has failed. Specifically, we need to

test for both Type 1 and Type 2 errors, i.e., how well we retain and how well we avoid the rejection of good tweets. First, we do a comparison with Naïve Bayesian approach. Next, we compare with one of the commercial platform, Socialmention [26].

4.1 Comparison with Bayesian Approach

For a baseline comparison, we have used Naïve Bayes classifier [14, 19], a popular method for document classification in anti-spam research [1, 22, 25]. It is widely used in text categorization task [19] and often serves as baseline method for comparison with other approaches [33]. In our implementation of Naïve Bayes (using the Laplacean prior to smooth the Bayesian estimation, as suggested in Nigam [19]) classification we extracted a set of keywords from every twitter post and used those as the feature set. Based on the key word occurrences in the twitter posts in the training data, probabilities are calculated. These probability values are used to classify the twitter posts.

Both the TApp and the Bayesian classification technique have been implemented using Java 1.6. We ran all the experiments using a Windows 7 machine with quad core processor of 2.33 GHz.

To compare TApp and the Bayesian classifier, we first selected a set of "apps of interest" – for this experiment, we chose the top 50 "hot" android apps using a popular mobile app search engine platform (`http://www.appbrain.com/apps/hot/`). To create our test bed for these 50 apps, we randomly selected tweets from our database of 14 million tweets and manually verified whether they contained references to these apps (*legitimate* tweets) or not (*spam*). In this fashion we manually identified 1000 posts from our database, consisting of 500 posts that refer to one of these 50 apps (*legitimate posts*) and 500 tweets that refer to mobile apps or internet web sites, but not any of the selected 50 mobile apps. We apply both the Bayesian classifier and the TApp approach on this test bed to classify these 1000 posts into *Legitimate* and *Spam*. In Figures 1 and 2, we plot the histogram distributions of accuracy of the two approach - Bayesian and TApp. As can be seen from Figure 1, the Bayesian classifier identifies 337 out of the 500 *Legitimate* posts (a recall rate of 67%), whereas the TApp approach demonstrates a recall of 97.2% by correctly classifying 486 of the 500 *Legitimate* posts. Similarly, as portrayed in Figure 2, the Bayesian classifier wrongly identified 174 of the 500 *Spam* posts as Legitimate, whereas TApp mis-identifies only 23 of 500. In Table 3, we have presented classical IR metrics such as precision, recall, true negative, accuracy and F-measure in both the cases. In all cases TApp significantly outperforms the Bayesian classifier (TApp scores above 90% in every case).

4.2 Comparison with SocialMention

SocialMention(SM) [26] is the leading social media search engine. To demonstrate the effectiveness of our approach, we decided to compare the accuracy

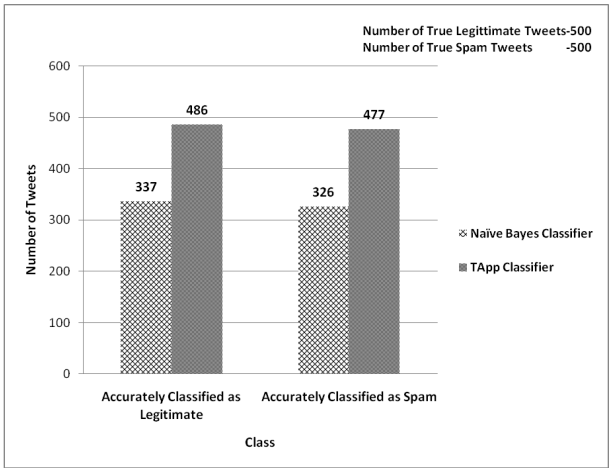

Fig. 1. Comparison of Accurate Classification

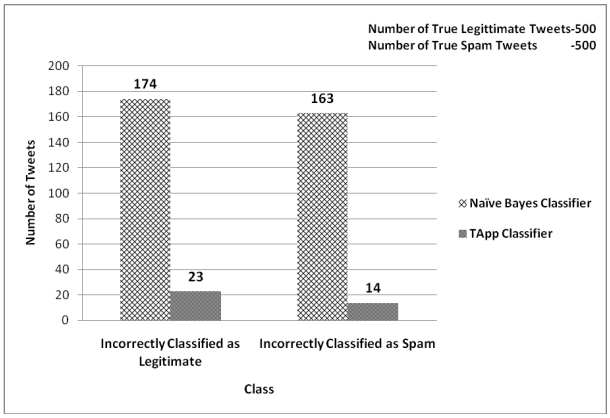

Fig. 2. Comparison of Incorrect Classification

Matrix	**Naïve Bayes classifier**	**TApp classifier**
Precision	$100 * 337/(511) = 66\%$	$100 * 486/(509) = 95.6\%$
Recall	$100 * 337/(500) = 67\%$	$100 * 486/(500) = 97.2\%$
True Negative Rate	$100 * 326/(500) = 65.2\%$	$100 * 477/(500) = 95.4\%$
Accuracy	$100 * 663/(1000) = 66.3\%$	$100 * (963)/(1000) = 96.3\%$
F-measure	$(2 * 65.9 * 67.4)/(66 + 67) = 66.7\%$	$2 * 95.6 * 97.2/(95.6 + 97.2) = 96.4\%$

Table 3. Comparison of IR metrics in Bayesian classifier vs. TApp

of our results with those acquired from Socialmention. As discussed in the Section 2, the exact algorithms of Socialmention implementation is not known. However, by observing different search results we concluded Socialmention uses an exact keyword matching approach to identify the twitter posts that contains the given keywords. In this experiment, we used the same set of 50 apps used in the previous experiment in Section 4.1. For each app, we retrieved the tweeter posts related to that app in the previous one month using both Socialmention API [27] and the TApp approach. The objective of our approach is to automate the Twitter post retrieval for large number of mobile apps. So, the input to both Socialmention and the TApp approach is app names as found in native app stores. The Socialmention uses these original app names to find the twitter posts. TApp approach applies name aliasing and name conflict resolution to retrieve the relevant tweets. However, the app names are chosen to be such that 22 out of 50 require either no aliasing and/or no name conflict resolution. This was done to access the effectiveness of the TApp technique in individually performing those 2 tasks.

To constrain the experimental data size, for each of the approach if the number of posts for an app is more than 50, we considered only the most recent 50 posts. Next, we passed the posts identified by both Socialmention and TApp along with the app names to two professional lexicographers. Each of the lexicographers has more than 5 years of experience of internet search optimization. They both worked together to arrive at an unanimous decision of which of these posts are "Valid" (i.e. the post is related to the respective app) and which of these are "invalid" (i.e. the post is not related to the respective app). We present the result in Table 4.

As can be seen from Table 4, for many apps, the Socialmention platform has retrieved tweets that are not related to that app. In total only 43.44% of the total tweets retrieved by Socialmention has been identified as "Valid" post by lexicographers. Whereas, for TApp approach, the absolute number of invalid posts for each app is much smaller compared to the Socialmention. Overall 95.45% of the twitter posts retrieved by TApp has been identified as "Valid" by lexicograpghers. The total number of valid tweets retrieved by TApp is 1584 compared to 769 by Socialmention. So both in terms of *accuracy* and the *coverage* of retrieval, TApp significantly outperformed Socialmention.

Additionally, we observe that Socialmention works well in cases when there no aliasing of the app names and when there is no naming conflicts between the entity of interest and other objects. In these cases, Socialmention achieved 82.93% accuracy. For example, the extracted tweets for the apps "Live Holdem Poker Pro","Google Sky Map","Handcent SMS" and "Lookout Mobile Security" in both Socialmention and TApp are highly relevant because these names are only used in mobile app domain and there is no aliasing by users. One should observe that, even in these simple cases, where there is no name conflict and aliasing, the accuracy in TApp case is higher than that of Socialmention. The exact approach followed in Socialmention is unknown, so we are not sure of the reason behind this improvement, however we anticipate that this is due to the

	Store App Name	Alias Name	Using SM		Using TApp	
			Valid	In-Valid	Valid	In-Valid
No aliasing & name conflict	CardioTrainer	CardioTrainer	45	5	50	0
	Endomondo Sports Tracker	Endomondo Sports Tracker	20	8	27	0
	Flash Player 10.2	Flash Player 10.2	3	2	0	0
	Google Sky Map	Google Sky Map	38	11	16	0
	Handcent SMS	Handcent SMS	37	13	45	5
	Instant Heart Rate	Instant Heart Rate	47	3	50	0
	Live Holdem Poker Pro	Live Holdem Poker Pro	43	7	48	2
	Lookout Mobile Security	Lookout Mobile Security	41	9	49	1
	Stardunk	Stardunk	39	11	27	0
	Total		313	69	312	8
	Accuracy		81.93%		97.50%	
Aliasing required, but no name conflict	Calorie Counter by FatSecret	Calorie Counter	3	4	50	0
	Documents To Go 3.0 Main App	Documents To Go	5	7	11	1
	Funny Facts Free 8000+	Funny Facts	1	1	48	2
	Bubble Blast 2	Bubble Blast	32	10	39	0
	Kid Mode: Play + Learn	Kid Mode	4	24	40	10
	Kids Connect the Dots Lite	Kids Connect The Dots	1	0	27	0
	PicSay - Photo Editor	Picsay	4	0	50	0
	Mango (manga reader) Free	Mango manga reader	10	3	43	6
	Pandora internet radio	Pandora	5	1	17	5
	SpeechSynthesis Data Installer	SpeechSynthesis	2	22	4	0
	Talking Tom Cat Free	Talking Tom Cat	28	14	48	2
	Vaulty Free Hides Pictures	Vaulty	1	0	26	0
	Waze: Community GPS navigation	Waze	2	1	50	0
	Total		98	87	453	26
	Accuracy		52.97%		94.57%	
Both aliasing and name conflict required	Adao File Manager	Adao File Manager	1	0	1	0
	Advanced Task Killer	Advanced Task Killer	38	12	31	3
	Angry Birds	Angry Birds	30	20	46	4
	Backgrounds	Backgrounds	3	47	38	0
	Barcode Scanner	Barcode Scanner	6	44	48	2
	Bible	Bible	0	50	13	5
	Craigslist	Craigslist	0	50	2	2
	Drag Racing	Drag Racing	11	39	49	1
	Epocrates	Epocrates	33	17	50	0
	ES Task Manager	ES Task Manager	2	5	8	0
	ESPN ScoreCenter	ESPN Scorecenter	34	16	43	6
	Facebook for Android	Facebook	18	32	42	0
	FxCamera	FxCamera	18	5	17	0
	Google Maps	Google Maps	2	48	26	6
	Horoscope	Horoscope	3	47	15	0
	KakaoTalk	KakaoTalk	9	41	48	2
	LauncherPro	LauncherPro	31	19	50	0
	Mobile Banking	Mobile Banking	6	44	47	3
	Mouse Trap	Mouse Trap	2	48	9	0
	My Tracks	My Tracks	0	49	4	0
	NFL Mobile	NFL Mobile	19	31	50	0
	Ringdroid	Ringdroid	26	17	18	0
	Tap Fish	Tap Fish	22	28	47	3
	The Weather Channel	Weather Channel	3	47	45	5
	Tiny Flashlight + LED	Tiny Flashlight	24	26	47	3
	Total		358	845	819	41
	Accuracy		29.75%		95.23%	
Total			769	1001	1584	75
Accuracy			43.44%		95.45%	

Table 4. Comparison of Valid Tweets in Socialmention vs. TApp

generic keyword matching algorithms followed in Socialmention, vs. the phrase search using tweeter API followed in TApp.

In the second scenario, when the app names required aliasing, but no name conflict resolution, the Socialmention's accuracy in retrieving relevant tweeter posts was 52.97% compared to 94.57% in TApp approach. For example, the tweets extracted for the apps "SpeechSynthesis Data Installer", " Kid Mode: Play + Learn" and "Vaulty Free Hides" are mostly irrelevant or unfound because of aliasing practice of users when they post their tweets. These apps are typically referred to as "SpeechSynthesis" ,"Kid Mode" and "Vaulty" in most of the tweets.

To show the effectiveness of TApp's entity name conflict handling, we focus on the third category of app names, where both aliasing and name conflict resolution are required. If we look at the valid tweet count for the apps "Drag Racing","Mouse Trap","Mobile Banking" and "My Tracks" in case of Socialmention, they are very low compared to the invalid tweet count. These app names are used outside the mobile application domain as well and so required name conflict resolution in TApp approach, which is clearly not done in Socialmention. For these type of app names, Socialmention had a pretty low accuracy of just 29.75% in retrieving relevant tweets compared to 95.23% accuracy in TApp case.

This demonstrates the importance and effectiveness of both the aliasing and name conflict resolution steps in TApp.

5 Discussion

In this section we discuss the broader implications of our TApp approach. Our research falls in the design science research paradigm of Information Systems [35]. We have developed an artifact that can successfully resolve name conflicts of app names in twitter posts. We have demonstrated the artifact through experimental study and a comparison with a manual method. Our two step approach out performs the benchmark Naïve Bayes classifier and a commercial implementation (Socialmention [26]) both on true negative and false positive errors.

Identifying social media mentions related to most popular products in general and mobile apps in particular has important implications for marketers as well as for product owners. Being able to predict the social media popularity of items have tremendous value to not only service providers but also marketers who would bid for ad-space on items with high potential popularity in order to maximize the exposure. TApp approach can be used to identify user generated contents across social media, which in turn can be used to measure product's popularity. Our approach can be utilized in many ICT research domains such as, identifying twitter posts related to brand monitoring in e-commerce, identifying the public opinions of e-participation, e-services and general e-government implementations by using the social media mentions, identifying students opinions of e-learning systems and analyzing the public views on digitizing the medical records of patients(Electronic Medical Records: EMR). Thus our approach is

generalizable and broadly applicable across wide range of ICT research domains in general.

6 Conclusion

In this paper we have addressed the problem of reliably identifying tweets related to mobile apps. In the process we have addressed the *aliasing* and *name conflict* problems inherent in the task. We have compared the accuracy of our approach to Naïve Baysian approach and a commercial implementation (socialmention). Our approach outperformed in all measures of accuracy compared to Baysian approach and the socialmention. While our application is in the area of mobile apps, our techniques are generally applicable.

References

1. I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, K. V. Ch, G. Paliouras, and C. D. Spyropoulos. An evaluation of naive bayesian anti-spam filtering. pages 9–17, 2000.
2. Apache. Open nlp. `http://opennlp.sourceforge.net/projects.html`, last accessed July, 2012.
3. AppleiOS. Apple-ios. `http://www.apple.com/ios/`, last accessed on July 14, 2011.
4. S. Banerjee. Clustering short texts using wikipedia. In *Proceedings of the 30th Annual International ACM SIGIR Conference*, 2007.
5. C. C. Chang and C. J. Lin. *LIBSVM: a library for support vector machines*, 2001.
6. H. Cui, J.-R. Wen, J.-Y. Nie, and W.-Y. Ma. Query expansion by mining user logs. *Knowledge and Data Engineering, IEEE Transactions on*, 15(4):829–839, 2003.
7. K. Dent and S. Paul. Through the twitter glass: Detecting questions in micro-text. In *Proceedings of AAAI-11 Workshop on Analyzing Microtext*.
8. L. Fausett, editor. *Fundamentals of neural networks: architectures, algorithms, and applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1994.
9. Filtertweets. Filter tweets for greasemonkey. `http://userscripts.org/scripts/show/87289`, last accessed June 27, 2012.
10. Google Inc. Android developers. `http://developer.android.com/index.html`, last accessed on July 14, 2011.
11. Google Inc. Google search appliance help center. `http://code.google.com/apis/searchappliance/documentation/46/help_gsa/serve_synonym.html`, last accessed on May 13, 2011.
12. X. Hu, N. Sun, C. Zhang, and T.-S. Chua. Exploiting internal and external semantics for the clustering of short texts using world knowledge. In *CIKM '09: Proceeding of the 18th ACM conference on Information and knowledge management*, pages 919–928, New York, NY, USA, 2009. ACM.
13. B. J. Jansen, Z. Liu, C. Weaver, G. Campbell, and M. Gregg. Real time search on the web: Queries, topics, and economic value. *Inf. Process. Manage.*, 47:491–506, July 2011.
14. D. D. Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. pages 4–15. Springer Verlag, 1998.

15. J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. L. Cam and J. Neyman, editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.
16. Markets & Markets. World mobile applications market - advanced technologies, global forecast (2010-2015). `http://www.marketsandmarkets.com/Market-Reports/mobile-applications-228.html`, last accessed on May 13, 2011.
17. Mashable. Mobile app market to surge to $17.5 billion by 2012. `http://mashable.com/2010/03/17/mobile-app-market-17-5-billion/`, last accessed on May 13, 2011.
18. Mobilewalla. Mobilewalla-an app search engine. `http://mobilewalla.com/`, last accessed on May 13, 2012.
19. K. Nigam. Using maximum entropy for text classification. In *In IJCAI-99 Workshop on Machine Learning for Information Filtering*, pages 61–67, 1999.
20. One Riot. The inner workings of a realtime search engine. 2009.
21. PC World. It's android vs. apple: Will you switch sides? `http://www.pcworld.com/article/199109/its\_android\_vs\_apple\_will\_you\_switch\_sides.html`, last accessed on May 13, 2011.
22. M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A bayesian approach to filtering junk E-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*, Madison, Wisconsin, 1998. AAAI Technical Report WS-98-05.
23. J. Sankaranarayanan, H. Samet, B. E. Teitler, M. D. Lieberman, and J. Sperling. Twitterstand: news in tweets. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '09, pages 42–51, New York, NY, USA, 2009. ACM.
24. N. Sarkas, N. Bansal, G. Das, and N. Koudas. Measure-driven keyword-query expansion. *Proceedings of The Vldb Endowment*, 2:121–132, 2009.
25. K.-M. Schneider. A comparison of event models for naive bayes anti-spam e-mail filtering. In *In Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL?03*, pages 307–314, 2003.
26. SocialMention. Social mention. `http://socialmention.com`, last accessed on Nov 5, 2011.
27. SocialMention. Social mention api. `http://socialmention.com/api/`, last accessed on Nov 5, 2011.
28. K. Spärck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):pp. 11 – 21, March 1972.
29. B. Sriram, D. Fuhry, E. Demir, H. Ferhatosmanoglu, and M. Demirbas. Short text classification in twitter to improve information filtering. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 841–842, New York, NY, USA, 2010. ACM.
30. Techcrunch. Report: Mobile app market will be worth $25 billion by 2015 apple ios share: 20%. `http://techcrunch.com/2011/01/18/report-mobile-app-market-will-be-worth-25-billion-by-2015-apples-share-20`, last accessed on May 13, 2011.
31. TweetFilter. Tweetfilter for greasemonkey. `http://userscripts.org/scripts/show/49905`, last accessed June 26, 2012.
32. Venture Beat. Why apple can not beat android. `http://venturebeat.com/2010/11/05/why-apple-cant-beat-android`, last accessed on May 13, 2011.
33. L. Zhang, J. Zhu, and T. Yao. An evaluation of statistical spam filtering techniques. *ACM Transactions on Asian Language Information Processing (TALIP*, 3:2004, 2004.

34. Znet. Android vs. apple: The 2011 cage match. http://www.zdnet.com/blog/btl/android-vs-apple-the-2011-cage-match/43682, last accessed on May 13, 2011.
35. A.R. Hevner, S.T. March, J. Park, and S. Ram. Design science in information systems research. *MIS Quarterly*, 28(1):75–105, March 2004.