Framework for Timely and Accurate Ads on Mobile Devices

Alex Penev Raymond K. Wong

mContext Pty Ltd & NICTA, Sydney, Australia

{alex.penev,raymond.wong}@nicta.com.au

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*information filtering, retrieval models, selection process*; J.7 [**Computers in other systems**]: Consumer products

General Terms

Algorithms, Experimentation

Keywords

Advertising, mobile device, monetization, search

ABSTRACT

We propose a framework for mobile advertising covering Value-Added Services where an ad-database is maintained on the device and both selection and display are dictated by the device. Advantages over existing mobile marketing are that ads are more timely, viable on a variety of use cases, can be both location-sensitive and personalized with minimal privacy concerns, and provide an obvious means for subsidizing users' service costs. We construct a suitable selection algorithm and evaluate its execution, accuracy and scalability. We show that ad-serving can be done under the processing constraints imposed by mobiles, which may lead to improvements in mobile marketing effectiveness.

1. INTRODUCTION

Following the advent of large-scale wireless infrastructure some three decades ago, mobile phones have become the fastest growing technology in history and represent one of the largest known consumer markets. The devices have evolved from bulky brick-phones to slim flip-phones and now touch-sensitive smartphones. Today there are some 2.5bn mobile users with 3.3bn subscriptions among them, representing 40% of the population.

Use is widespread in developed countries with the US having 86% penetration and most of Western Europe boasting over 100% penetration due to multiple subscriptions. But the market has become saturated because those who want a phone already own a phone. This means that service providers compete for a share of existing users. They attract customers by offering a more attractive handset or more attractive price, and sustained revenue comes from subscriptions. A big money-maker is *Value-Added Services*, an industry term encompassing SMS, EMS, MMS, MIM, voicemail, email, web browsing, mobile TV, downloadable content (games, apps, video, music) and all services beyond end-to-end calls that 'add value' to use.

VAS make mobiles more functional, convenient and entertaining, but often incur a separate payment. Their price affects subscription rate and 3 in 4 users cite price as the critical deciding factor [5]. Thus, lowering VAS prices should be a priority for service providers who want to increase market share. One way to lower prices is to subsidize costs with advertising, but ads will need to be timely and accurate to be effective. Timeliness refers to ads being shown at opportune moments. Accuracy refers to ads being contextually relevant to the user activity, geographically relevant to all users in an area or personally relevant to the individual.

Existing mobile marketing approaches struggle with the timeliness and accuracy requirements, while our framework addresses both. Shown in Fig 2, it consists of an ad-database and ad-selector software on the device.

The ad-database stores ads of various formats, each labeled with keyword tags to facilitate efficient indexing, retrieval and scoring. The ad-database is localized (region-relevant) and synched to the ad-broker via periodic updates.

The ad-selector filters, scores and ranks ads for a query. This 'query' is a piece of arbitrary text understood to represent the user activity or displayed content and a text representation is possible for most VAS. The ad-selector can also track the users perceived interests and bias its selections, as well as make location-sensitive choices.

This paper is organized as follows. Section 2 gives some background on mobile advertising and the existing problem space. Section 3 constructs an ad-selector to address this problem, which is then evaluated in Section 4. Section 5 summarizes related work and Section 6 concludes.

2. BACKGROUND

The process described herein is an upcoming subsystem of mContext's existing product for indexing large data on mobiles. It complements the product by enabling monetization of packaged data. This paper generalizes the subsystem as a generic ad-serving agent for heterogeneous mobile content.

CIKM'09, November 2–6, 2009, Hong Kong, China.

Figure 1: Ads can now be timely and relevant, being shown in response to user interactions. Here an SMS exchange invokes the ad-selector, which finds a potentially useful ad of a movie trailer. A teaser is temporarily overlaid (e.g. as a banner along the top, in free space) to invite the user to click it and see the ad content.

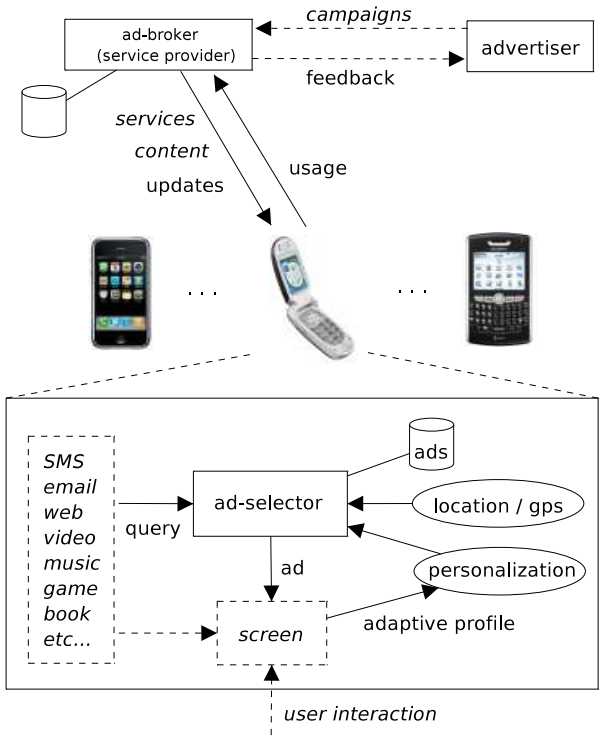

Figure 2: Overview of framework.

2.1 Mobile devices

For some a mobile is strictly a communication tool, but for many it is an important everyday accessory kept within reach at all times. The power of the devices is improving, with Apple's iPhone, Palm's Pre, Samsung's Omnia and Sony Ericsson's Xperia all boasting 400–624MHz processors. The hardware allows mobiles to perform computationally intensive tasks such as play video games. In terms of capacity, modern phones have multiple gigabytes of storage as built-in or via an SD card.

Value-added services (VAS) add value to mobile usage. Used by 3 out of 4 owners, SMS is the most popular with an estimated 2 trillion messages transmitted in 2008. Other VAS are growing their own customer bases, such as the 800m users who browsed the mobile web in 2007. The large customer base creates a lucrative market for advertisers.

2.2 Online advertising

The advertising model is largely responsible for the livelihood of traditional media such as broadcast television, newspapers, magazines and radio. Modern examples of its success are online targeted keyword advertising systems such as AdWords. Targeted advertising has a long media history in magazines and television, and keyword advertising itself predates search engines by a century—the first 'Yellow Pages' was published in the 1880s.

The era of web advertising began in 1993–94 with GNN and HotWired, both selling click banners. In 1996 OpenText improved the ROI by placing specific ads alongside specific searches on a search engine. GoTo was more successful with a similar idea in 1998. Google launched AdWords in 2000 to monetize its search and lated turned it into a self-service portal to give advertisers high levels of control.

The state-of-the-art in targeted advertising is now online. The three major search engines are also the three largest online ad agencies, reaching 80% of the world [1]. All three provide Sponsored Search and Contextual Advertising. In the former, ads are shown alongside search results and can be highly relevant because the user immediately reveals an information need with their query. In the latter, small ads are inserted into generic webpages based on multiple relevance factors (Section 5). Sponsored search is more effective with conversions [1], but contextual advertising helps monetize much of the internet today. Despite persistent challenges from ad-blocking and click-fraud, online advertising is a big business (e.g. it forms 97% of Google's revenue [2]).

2.3 Mobile advertising

Although a few years younger than its online counterpart, mobile advertising is growing rapidly. Strategy Analytics projects expenditure to boom 900% over the next 3 years. The two main types of mobile ads today are advertiser-to-user messages and content-embedded ads.

Message ads are prominently SMS, EMS and MMS. Each is a form of telemarketing that is more intrusive than embedded ads because messages can not be ignored—phones flag their arrival with an icon or pop-up and users must dismiss the notification to return the phone's prior state. Users must also manually delete the ad even if they do not open it. But message ads suffer from bigger problems: poor timing, poor accuracy and cost of repeated transmission. They are not timely because the usually arrive when the phone is on standby. Phones that beep to notify the arrival of messages may even interrupt the user from another task.

Message ads are generally targeted using demographics and focus groups. Individuals may also be targeted if the advertiser knows their interests. This is typically done using a questionnaire, but unfortunately the practice is invariably opt-in, thus suffers from low participation and fails to make optimal use of the customer base. Even when user-targeted, message ads do not address the problem of poor timing.

The embedded ads are more timely. They are typically injections, prepended clips or web-embedded ads. Injections occur when a message passes through an intercepting server that parses it, inserts an ad and forwards it to the recipient. Prepended clips are short audio-video segments played before a normal clip. Web-embedded ads are online ads viewed on the mobile browser.

While they primarily generate revenue for websites instead of the service provider, web ads are still highly interesting to both service providers and users. From the provider's view, a user click means more bandwidth and more revenue. From the user view, the ads relate to the page and are always up to date. One disadvantage of web ads is that they handle only browsing, which is only one type of VAS. Another is that they cost users money (for bandwidth) even if users ignore them. This somewhat limits them to text because users who are not interested in ads will not want to spend their bandwidth on images or video ads. In this sense, our framework complements web ads by showing ads of similar timeliness and relevance for other VAS content. Users who normally ignore web ads (i.e. links to websites) may be interested in our system since we can show more kinds of ads (not only links to websites) while even saving the user money rather than spending it without permission.

It is also worth highlighting the increased use of secondary communication channels such as Wi-Fi and Bluetooth, where users are not downloading content from the service provider or from the web. They may download it from their laptop, a kiosk, another phone, etc. Our framework may heighten user experience in these cases if we can show useful and relevant ads. In particular, users may enjoy our system because we can show entertaining ads such as images, movies and interactives [6] that are already on the phone.

The first MVNOs to substantially subsidize mobile costs are recent. In 2007, Blyk.co.uk launched a free service where users were required to view ads in exchange for mobile credit. Despite being UK-only, age restricted and invitation-based, the service achieved reasonable success by reaching its 100k customer target in half the expected timeframe. In 2008, ComTel.com.au announced a similar service where users receive 5 SMS ads per day. Both companies rely on SMS delivery. According to Empowered [10], 40% of surveyed users say they 'pay too much' for mobile use and 50% would consider receiving SMS ads if ads reduced costs and were relevant to their general interests. If the market accepts SMS ads tailored to general interest then it should be more welcoming of timelier ads tailored to immediate interest.

2.4 Improving mobile advertising?

The imbalance of reach and effectiveness between online and mobile methods may lead us to think that perhaps a better way to advertise on mobiles is to 'dial a search engine'. Indeed, many types of displayed content have a text representation and engines excel at analyzing text. But this notion raises many questions. What content does the phone send? Isn't it private? And how often should it contact the engine? For every user action? Who will pay for this bandwidth? And what about latency? Is money lost if the user has switched activities by the time the ad arrives?

In this paper we place a mini search engine within the phone, thus resolving these questions. In particular, we provide a filtering and selection method that works under the heavy size and processing constraints imposed by mobiles. We show that it can filter through many ads with fast response time, achieves good accuracy and is scalable.

The system has multiple advantages over current mobile marketing as it addresses timeliness, relevance, localization and personalization for multiple types of content:

Ads are relevant and shown based on viewed content.

Ads are timely because they relate to the current activity. For instance, if reading (or writing) a message '*want to see a movie?*' to a friend, the user is likely thinking about her friend and about movies. A very basic query would be simply 'movie'. Retrieved ads may vary in presentation and format—a box office trailer, a TV schedule, a link to IMDB, a cinema ticket voucher—but all would relate to her thoughts. As is it sometimes desirable not to show an ad [8], the ad-selector can refrain using both a back-off function and a minimum confidence threshold.

Ads can be localized because the ad-database is region specific, e.g. a Palo Alto pizzeria would not have its ad on phones in New York. On a location-aware phone the ad-selector can make location sensitive choices, e.g. a nearby cinema or local movie rental shop for the 'movie' query.

Ads can be personalized by adaptively building and using a profile of the user's interests, based on prior clicks.

Ads can be shown for heterogeneous activity, including SMS, music/video downloads, web browsing, etc.

Ads can save users money by subsidizing service costs.

The database is exchangeable and allows domain-specific ads to be packaged with mobile content such as digital magazines, e-books and portable encyclopedias. For example, a digital tourism guide that users download to their phones at airports may be packaged with tourism ads.

Intuitively these advantages may be able to provide a cheaper price and better user experience for mobile users, a better way to reach a market for advertisers, and more customers for the service provider.

3. APPROACH

3.1 Ad-database

The ad-database consists of ad records with an id, title and set of tags. The tags are used for indexing, retrieval, relevance scoring and personalization.

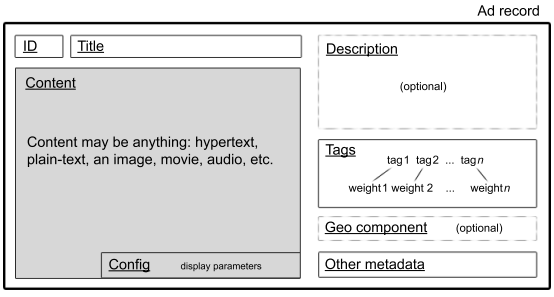

Figure 3: Abstract ad representation.

Fig 3 shows an ad entry. `ID` is an ad's machine reference, delegated by the ad-broker, that allows simple internal referencing and updates. `Title` is how humans reference the ad and is included for two reasons. One is that the small teaser banners, such as Fig 1, need a text label. The other is that, as shown in [18], titles serve as an additional supporting feature for matching. The `Description` is an optional text forming a second such feature.

`Content` is the ad itself and `Config` are its parameters such as compression and encoding. The **Metadata** are extra fields tracked for updating and bookkeeping purposes. Useful fields include checksums, base ratings, expiry, timestamps (latest update, view, click) and total views and clicks.

`Geo` is an optional component defining the ad's relevance area, which will be used to boost or penalize candidate ads that are near or far from the user. A basic representation is a set of ⟨latitude, longitude, radius⟩.

The `Tags` are text annotations that help index and retrieve ads efficiently. They also allow fast relevance approximation, scoring and personalization (Sections 3.2.2, 3.2.3, 3.2.6). Each tag has a weight attached that represents its *strength*. Advertisers may want high levels of control over tag choice and billing, therefore the weight will likely incorporate the tag's relevance as well as its bid and budget health. This paper focuses on timing and accuracy instead of billing, so we treat the tags as content descriptors and the weight as content relevance. It should be noted that this is not a limitation in practice because meaningless tags can be useful; suppose a record label like EMI offers free mobile songs provided EMI ads are shown during play. Using a very high weight, broker-approved special syntax tag on both EMI's songs and ads will match only them.

Updates and administration

The ad-broker's tasks involve contracting with advertisers and provision of updates. The updates should be divided into region-specific patches so that the ad-databases are reasonably localized. For convenience and privileged network access, the broker may be the MNO. In terms of size, the broker's master database of ads will be much larger than the phones' ad-databases. This is because the broker may want to track new ads, old ads, and ads for multiple regions, whereas individual users are only interested in ads relevant to them. Ideally we would like the ad-database to support as many ads as possible provided the phone can search them within reasonable time. Giuffrida et al. [12] report a mobile marketing company inducting 20–30 ads per day, so supporting 30×365=11k ads would be a start.

The phone will need to return some feedback to the broker so that advertisers can receive statistics on their campaigns. Advertisers will at least want to see CTR and individual tag performance, meaning that clicks and their triggering tags should be part of bookkeeping. Other personal data can stay on the device.

The remainder of this section constructs a prototype ad-selector to be evaluated in Section 4.

3.2 Ad-selector example

The ad-selector filters suitable ads for some input text query. In a commercial system the ad-selector would need to obtain its input automatically. For instance, to receive the SMS exchange in Fig 1 it needs to be invoked by the phone's OS, be part of the OS, or be a registered listener to message events. Even after an ad is selected the agent must decide where and how to display it. These are out of scope because we are concerned with a selection, and this paper assumes that the ad-selector's task is to convert a piece of English text into a list of ads with selection probabilities.

As mentioned in Section 3.1, we will treat the tag weight as a content relevance although in practice they can encode extra information that influences selection. Thus, without loss of generality the selection steps are:

1. **Transforming** the input text into weighted tags.
2. **Query expansion** to add some extra tags.
3. **Retrieval** of candidate ads.
4. **Prefiltering** to discard weak candidates based on their approximated relevance.
5. **Scoring** the passed candidates with a more in-depth (i.e. more expensive) measure.
6. **Postfiltering** to discard weak candidates after scoring.

The remaining candidates enter a final draw ('lottery') where selection is randomized based on the probabilities as defined by the ads' scores. If no ads scores highly enough then the ad-selector may decide not to return an ad. The remainder of the section describes the above steps.

3.2.1 Transformation and expansion

While not strictly part of the selection process, allowing arbitrary input makes the framework more versatile. Perhaps the ideal kind of input query would be weighted, human-assigned tags that are assigned in a similar way as how advertisers assign the ad tags (viz. an *ESP Game* between users and advertisers). Certainly the matching would then be almost direct and thus very fast. Unfortunately, most VAS content does not have human tags (or any tags) so we approximate it mechanically.

Our transformation is relatively simple, extracting salient TFIDF keywords and growing the top ten terms. This process uses case-folding, tokenizing, stopword removal, stemming, TFIDF and query expansion. The calculations require a stemmer, stoplist, IDF table and relationship table to be kept on the phone. In Section 4 we use a Porter stemmer, 600-token stoplist, 90k-token IDF table and 60k-token relationship table, all of which total 0.1% of an iPhone's capacity. In particular, the relationship table maps tokens to a mix of weighted synonyms and weighted co-occurrent terms, as in [16].

The top ten TFIDF input terms are expanded by adding at most 50 related terms each from the relationship table. As we will see in Section 4, the expansion typically results in 12 new words per term. The expansion terms' weights are set to 25% of the product of their generating term's TF and the relationship strength (from the relationship table) with that term. Consequently they are usually a low weight and low importance.

For example, for the input message '*want to see a movie tonight?*' the tags are 'movie' and 'tonight'. They are expanded to create a tag-form query of 34 terms that includes (the stems for) film, television, actor, night, evening, etc.

3.2.2 Retrieval and prefiltering

Query expansion substantially increases recall but many of the matched candidates are weak. For performance reasons, they are now be discarded so that only promising ads pass to the scoring step. Differentiating the promising from the weak requires approximating their relevance.

Retrieval: candidates are ads which match at least one tag in the expanded tag-form query. The index maps tags to (id,weight) tuples. To approximate scores we sum the product of the tag's IDF, query weight and ad weight.

Prefilter: weak candidates are discarded to keep only the top-k. Determining k can use a score-based or size-based cutoff, but the prefilter is only needed for performance reasons and therefore we use size-based filter where we take the top-k ads as ordered by their approximation. In general k should be a function of the database size. In the experiments we use relatively small datasets and set $k = 50$. Note that the ad-selector is concerned with selecting a *a single ad* for display, so it does not need to maximize its recall.

3.2.3 Scoring

The previous two steps provided a weighted tag-form query and a set of promising candidates. These candidates can now be scored for their suitability. We use the tags, location and personalization to score ad A against query Q as:

$$s_{A,Q} = base(A) + rel_{A,Q}\,(1 + \alpha\ g(A))\,(1 + \beta\ p(A))$$

where $base(\cdot)$ is a base rating, rel is a relevance score (Section 3.2.4), $g(\cdot)$ is a function returning a penalty or boost in [-1,1] for A's location (Section 3.2.5) and $p(\cdot)$ is a personalization factor in $[0, 1]$ (Section 3.2.6).

The ad will receive a low score if its rel is low or if it specifies a location and the location is inappropriate. It will receive a high score if its rel is high and even higher if it is aligned to the user's interests or has an appropriate location. The parameters α and β are tunable but from preliminary testing $\alpha = 1$ and $\beta = 4$ are suitable. Note that $g(\cdot)$ has a positive, neutral or negative effect while $p(\cdot)$ is only positive or neutral. This is because we can adaptively learn a person's likes but not so easily their dislikes.

The base score is useful for ads that always have a positive relevance and qualify as candidates for any query. This may include service provider bulletins, but it would be more suitable for temporarily boosting ads that the user has historically shown strong interest in and has not yet clicked since their most recent update. The magnitude of the rating affects the increase in selection probability. A typical ad that may suit this is the movie session times at the cinema, which are usually updated weekly.

3.2.4 Relevance

The relevance between the ad tags and query tags can be performed with greedy sum to maximize the number of overlapping tags and their strength:

$$rel_{A,Q} = \sum_{t \in (A.tags \cap Q.tags)} w(t, A.tags) \cdot w(t, Q.tags) \cdot IDF_t$$

where $w(t, \cdot)$ is a weight in [0,1] and IDF measures rarity. An obvious extension to this calculation is to use the Fig 3 title and description as a supporting match, but we do not implement this in our prototype because we wish to show experimental results for using tags exclusively.

3.2.5 Location

The ad-selector can make location-sensitive choices if it knows its location. Aside from GPS, a mobile can obtain its location by other means, such as by letting the user place a marker on a digital map, prompting the user for a zipcode or triangulating a position using signal strength from known call towers.

Section 3.1 specified location as optional because many ads are location-independent. For example, the Coca-Cola logo and many websites are globally relevant. But business such as restaurants and shops have fixed addresses and can benefit from location sensitivity. Location sensitivity is somewhat out of scope because the ads in our experiments do not have locations (they are web ads), but the aim of this section is to show that it is possible to add location functionality to the ad-selector in a way that is fast to compute and produces useful values.

The geography function $g(\cdot)$ acts as a boost or penalty. It does not need to be symmetric or continuous but we would like to control its effect and clamp it to $[-1, 1]$. Desiderata for $g(\cdot)$ are that ads with no location are unaffected, being close to a focused ad with small radius is better than being close to an unfocused ad, being within radius should never incur a penalty, and that we need some leeway at the radius edge because it is not always intuitive for advertisers to specify. Below is a piecewise example that satisfies these:

$$g|_r^d = \begin{cases} 0 & \text{if unspecified} \\ 1/\sqrt{\log d \cdot \log r} & \text{if } d \le r \\ (g|_r^r + 1)^{\frac{3}{4}^{\log((\frac{1+d}{1+r})-1)}} - 1 & \text{if } d > r \end{cases}$$

where r is radius, d is user distance and log is a non-negative $\log_2(2 + x)$. In the third case $g(\cdot)$ takes the value at the edge and reduces it by a fixed amount each time the user's distance effectively doubles. The maximum boost is when the ad is focused and the user is there while the maximum penalty is when it is focused and the user is infinitely far.

Suppose a small pizzeria targets customers within a 3 mile radius. By not giving its location the pizzeria's ad will compete against other ads, such as a national pizza chain, based on tag calculations only. By giving a location it can gain a boost when the situation is economically favorable, i.e. when the user is close. If a user is 1 mile from the pizzeria, its ad score will be boosted by $g|_3^1 = 52\%$. A user 10 miles away will not have it boosted and any users farther away will have it penalized. At 50 miles the ad would be penalized by 40%, which favors the national chain and may result in the pizzeria's ad being discarded by the postfilter.

3.2.6 Personalization

People naturally have both persistent and transient interests, which we refer to as long-term (LT) and short-term (ST). The ad-selector can model these using tags, build a simple user profile and align its selections with the profile.

Passive observation has several advantages over the 'questionnaire approach': it requires no user effort, it is self-updating, it can glean fine-grained interests and it can track ST interest. We model LT interest as repeated confirmations of ST interests and ST interests as ad clicks. For example, the transient interests for a computer engineer may be {*movie, wolverine, jackman*}, perhaps after viewing the Wolverine trailer ad of actor Hugh Jackman. Her persistent interests may be broader tags such as *technology*, *internet*, *movie* and *software*. Meanwhile, a novelist's persistent interests may include *book*, *movie*, *news*, *music*, etc. Both users share a movie interest, but their other unique tastes can influence the ad-selector into choosing different ads for these

users given the same database and input. We model the two interests using tags as follows:

long term: the most frequent tags from historical ad clicks. Their interest score is assigned by normalizing the frequencies.

short term: tags from recently clicked ads are added to the top of the ST list and gradually pushed down. Their interest score is initially 1 and the whole list is decayed by a factor $0 < d < 1$ whenever a new ad is clicked, when n ads in a row are shown but ignored, or every h hours. A tag is dropped from the list when its score falls below a s_{min} threshold.

From preliminary testing suitable values are $d = 0.8$, $n = 8$ and $s_{min} < 0.1$. Suppose the lists are labeled LT and ST, each mapping tags to interest scores in (0,1]. A basic personalization function for ad A with tags $t_{1..n}$ would be to return 1 if any t_i is in either list and 0 otherwise. Using the interest scores and t_i's weight would give a more fine-grain scoring and we can also stipulate that it is better for A to match both lists. This gives a simple $p(\cdot)$:

$$p(A) = \frac{1}{2} \max_{t_1,t_2} (w_{t_1} ST[t_1] + w_{t_2} LT[t_2])$$

3.2.7 Postfilter and final selection ('lottery')

Aside from location and personalization factors, the selection process so far produces the same result each time. If we were to always choose the top candidate then there would be little variety in selections, especially if the same query is run moments later. It is therefore desirable to randomize the final selection to provide variety. We have already determined the ads' scores and can use them to set their lottery probabilities. But before making the choice, a postfilter is needed to again discard weak ads. Despite passing the prefilter, these ads may have been weakened by underperforming in the scoring comparative to the other candidates (e.g. penalized for location). Unlike the prefilter, which is used for performance, the postfilter decides the final candidates and is critical to the final accuracy.

The filter can be either size-based (quantity) or score-based (quality). The advantage of size-based is it provides guarantee that there will be some variety in selections, whereas with score-based only the most promising ads would be considered even if there are very few of them. In either case, a filter that is too strong will pass too few ads and a filter that is too soft will pass too many weak ads. We conduct some experiments in Section 4.2.2 to find a suitable postfilter.

The selection process for the sample ad-selector constructed in this section combines the steps from Sections 3.2.1-3.2.7:

ad-selector : set ad probabilities for transformed input Q.
1. $E := Q.tags \cup \bigcup_{t \in Q.tags}$ queryexpand(t)
2. $C := \{cand \in$ ad-db $\mid cand.tags \cap E \neq \emptyset\}$
3. apply prefilter to take k candidates
4. add ads with positive base rating
   //now score candidates using more features
5. for each $c \in C$:
6. $\quad rel_{c,Q} := \sum_{t \in c.tags \cap E} w(t, c.tags) \cdot w(t, E) \cdot IDF_t$
7. $\quad s_{c,Q} := base(c) + rel_{c,Q}(1 + g(c))(1 + 4p(c))$
8. apply postfilter to discard weak candidates
9. set probabilities using L_1-norm

The following section provides time and accuracy experiments for this process. Due to page limits we do not give experiments for the location and personalization steps individually. Nevertheless, we include them as part of the construction to show that they can both be implemented with simple calculations that befit the problem domain. Their calculations will be included in the timing experiments but disabled for the accuracy experiments.

4. EVALUATION

The implementation of the Section 3 ad-selector being evaluated runs as a CLDC prototype on a mobile simulator (Java Wireless Toolkit 2.5.2). Two ad-databases were compiled for testing:

G-db: 430 real web ads from Google's search engine. The 114 generative queries used to fetch these ads came from Google's top searches[1]. We require ads to be tagged with weighted tags, so we generate keyphrase recommendations using an automatic labeler [3]. From the recommendations, the 10 most recommended keywords (unigrams) are used as the tags (in some cases fewer than ten were recommended). The tag weights were set as the normalized TFs among the recommendations such that 'best' tag received a unit weight. For the cases where the initial query's keywords were in the top ten or were not recommended, we added the query keywords as tags with a medium weight. (The query was added because it represented the only label that was virtually guaranteed to have been used by the real advertiser for the ad to have been retrieved.)

D-db: 4300 random Delicious entries. The objects are web bookmarks instead of ads, but we are only interested in their tagged nature and not their content. The tags were assigned by Delicious users and naturally have a weight represented by their frequency. The more popular the tag with users, the closer its weight to 1.

Note that D-db contains non-ads tagged by humans while G-db contains real ads tagged and rated mechanically.

4.1 Dataset independence

A first experiment is to show that timing is largely dataset independent. We constructed three test ad-databases of equal sizes but different natures:

- 430 real ads from G-db; avg. 9.9 tags per ad, σ=1.5.
- 430 random ads from D-db; avg. 6.1 tags, σ=5.9.
- 430 carefully chosen D-db ads that resulted in the same tag distribution as G-db; avg. 9.9 tags, σ=1.5.

Fifty queries were executed on each of these ad-databases. The queries consisted of tags randomly chosen from the tagspace intersection such that each query would produce a result for each database and trigger all selection steps. Execution time is shown in Fig 4. Compared to G-db, the closeness of the middle column indicates that changing the distribution's mean and variance has no noticeable effect while the third column indicates that there is little timing difference between human-assigned and machine-assigned tags.

[1]http://www.google.com/intl/en/press/zeitgeist2008/

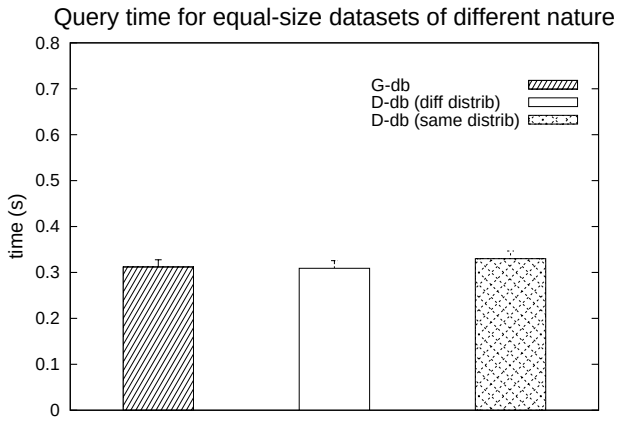

Figure 4: Mean processing time for ad-databases of differing tag sources and tag distributions.

4.2 Accuracy

Showing ads in response to user interactions makes them timely and Fig 4 shows that retrieval can be sufficiently fast, but to be effective ads also need to be accurate. Assessing accuracy requires a set of ads, queries and corresponding relevance judgments. The G-db collection of real ads was used for the following experiments.

4.2.1 Queries and relevances

We formed 3 variants of a subset of the 114 generative queries used to compile the G-db:

Direct queries: 35 of the 114, randomly chosen. Examples: *software, coldplay, paint, java*.

Near queries: 35 topically aligned queries, one for each direct query, that used different keywords. The near query was created from the direct query via refinement: a human judge choose three new keywords from Google's related queries shown below the search results. The near queries for the above examples were: *computer hardware ipod, band tour lyrics, picture color interior, plugin flash microsoft*.

Mix queries: the direct and near query as one.

Ads in G-db were then judged for relevance relative to the 35 direct queries only (i.e. not to the near query). Both 'strict' and 'kind' judgments were taken. Under strict judging ads needed to be "very relevant" while under kind judging ads needed to be "somewhat relevant". For example, for the query *travel*, strict judgments included ads for flights and accommodation, kind judgments included ads for Visas and Google Maps, but ads for outdoor camping or gift shopping were not considered relevant.

Since the direct queries were popular search queries, they generally consisted of 1–2 words and some were quite broad, therefore even strict judging accepted many ads: a median of 10 per query ranging from 4 for *ice cream* to 35 for *shopping*. Kind judging shows a median of 21 and range of 8–74.

4.2.2 Finding a postfilter cutoff

The ad-selector's final step before making a choice is a postfilter (Section 3.2.7) to remove weak ads. The candidates which pass the postfilter enter the lottery, but a question remained of how many ads to discard in a way that ensured some variety in selections but did not allow too many weak ads. The filter could use either a quality (score-based) or quantity (size-based) threshold.

Intuitively, using too strong a cutoff will increase precision but decrease recall and may lower the user experience by not having enough variety in selections for the same query. Too weak a cutoff would improve the experience but decrease precision. To find a compromise we calculated the F-score under both types of postfilters for each of the three query variants. The results in Fig 5 consolidate to maximum means at 3.0% for size-based and 18% for score-based. The F-score for the direct query at these points is 0.57 and 0.62, respectively. The difference is small but it appears that score-based is better. The next experiment confirms this.

4.2.3 Precision and MAP

The previous experiment found suitable cutoffs for the two choices of postfilter. Their Precision and MAP is shown in Fig 6. At the optimized cutoffs, MAP is ≈0.5 for both but precision is much higher for the score-based filter.

MAP is widely-used and gives some insight into both Precision and Recall, but Recall is misleading here because the ad-selector applies two filters to purposely remove retrieved ads for performance and accuracy reasons (Sections 3.2.2, 3.2.7). The cutoffs naturally reduce recall and render its final value meaningless. Instead, we maximize the *relative* values of Recall as part of tuning the F-score, and now focus only on precision.

Table 1 shows the P@(1,5,10,|*Lottery*|) at the two F-score maximizing cutoffs for the 3 query variants, using both postfilters, and under both strict and kind relevance judgments. The results confirm that score-based is the better postfilter because it has far better lottery precision.

Judg.	Query/postf	P@1	P@5	P@10	P@|L|	Acc%	G
Strict	Direct score	.91	.75	.57	**.80**	**85**	(89)
Strict	Direct size	.91	.78	.62	.55	79	
Strict	Near score	.49	.46	.36	**.44**	**47**	(-)
Strict	Near size	.49	.47	.39	.35	45	
Strict	Mix score	.83	.74	.55	**.70**	**76**	(-)
Strict	Mix size	.83	.76	.58	.52	70	
Kind	Direct score	.97	.85	.69	**.94**	**96**	(99)
Kind	Direct size	.97	.91	.78	.73	92	
Kind	Near score	.74	.63	.50	**.63**	**67**	(-)
Kind	Near size	.74	.66	.56	.52	64	
Kind	Mix score	.94	.87	.70	**.87**	**91**	(-)
Kind	Mix size	.94	.90	.78	.67	84	

Table 1: Precision using both strict and kind judgments for three query variants under two possible postfilters. The ad-selector's final accuracy, as well as Google's for the same dataset, is on the right.

While P@n metrics reveal the goodness of a ranking, the ad-selector's choice is based not on the ranking but on the scores that defined the ranking order, i.e. the probabilities of the lottery candidates. Keeping in mind that the lottery has already been optimized for F-score, the more important measure of accuracy is the percentage of occasions where the final choice would be one of the query-relevant ads. This figure is given in the Acc% column in Table 1. The accuracy of our sample ad-selector was 85% under strict judgments for the direct queries. For near queries it was nearly 50%.

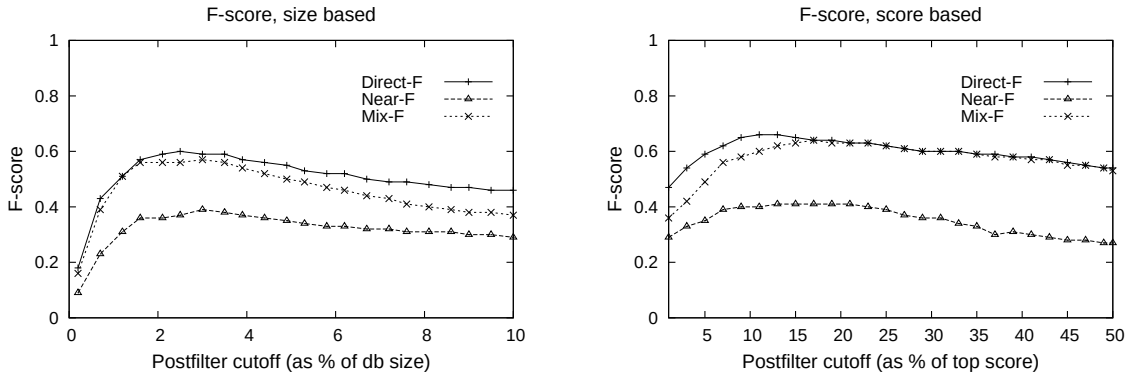

Figure 5: F_1-score for a size-based vs. score-based postfilter, using strict relevance judgments. The three query variations consolidate to peaks at 3.0% for size-based and 18% for score-based.

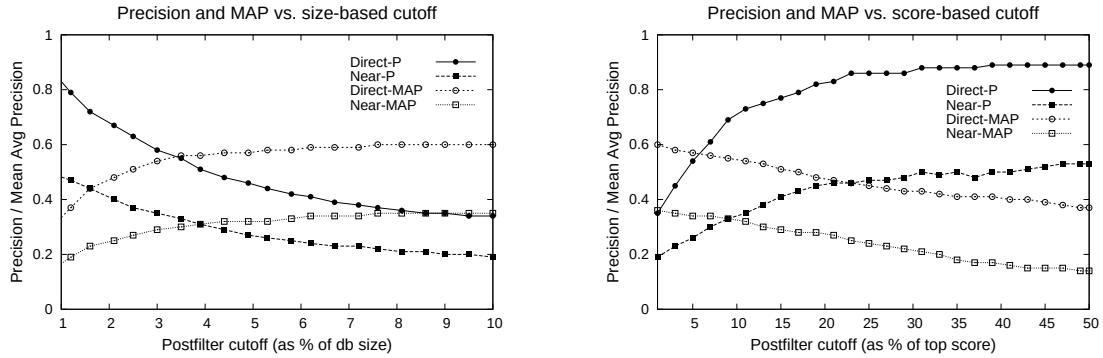

Figure 6: Precision and MAP for the lottery, using strict relevance judgments. The score-based cutoff (at its optimal 18%) outperformed size-based (at 3.0%) and was the better postfilter.

We believe that search engines represent the state-of-the-art in targeted advertising and, because the G-db was compiled from an engine's results, the same ads, queries and relevances could be used (retrospectively) to assess the engine's accuracy. This value is given in the last column of Table 1. The accuracy is very high for both systems, which can be explained. In the case of the engine, the queries were the engine's own top searches, which provide much historical CTR data for choosing ads that the users are most likely to click on. In our case it is likely a dual factor of the G-db being small and that the ads were labeled using the engine's label assistant [3] (which can make intelligent tag suggestions using various collected data). A comparison with the engine is useful because the ad-selector is concerned with filtering ads that are labeled, not with what means were used to label and weigh the labels.

Note that the near and mix queries in Table 1 are included as reference, but the search engine's accuracy is not assessed since it shows completely different ads for those queries.

4.3 Scale

Determining scalability requires knowing the cost of each step. As this experiment concerns timing instead of relevance, we can use the larger D-db. Both the D-db and a 50% random sample of it were used to show the effects of doubling the number of ads.

Three sets of realistic queries were constructed (with 30 instances) and grouped according to their difficulty. Here difficulty refers to workload and selectivity instead of semantics and disambiguation. The query sets are described in Table 2. **Easy** queries were single tags randomly chosen from D-db. Being indexed meant they would produce a match and trigger all selection steps. Being randomly chosen meant they would generally lie in the long-tail of rare tags, hence match few items. **Medium** queries were popular Delicious tags, which are very common in the D-db and produced many candidates for every query. **Complex** queries were the plain text of same-day news stories from an online news site. This time the input consisted of long passages and required more substantial use of the transformation step. The larger number of input terms resulted in bigger query expansion and even more candidates (sometimes over 1/3 of the database) that required prefiltering.

The step cost breakdown is in Fig 7 and shown as stacks. The bottom stack, the transformation step, used 0.05s per 100 input words and was invisible for short queries. Query expansion averaged 0.13s for the single tag queries and just 0.2s for complex. The time was kept low because only ten terms are expanded (Section 3.2.1). The retrieval and prefilter, a dual-purpose step that repeatedly probes the index,

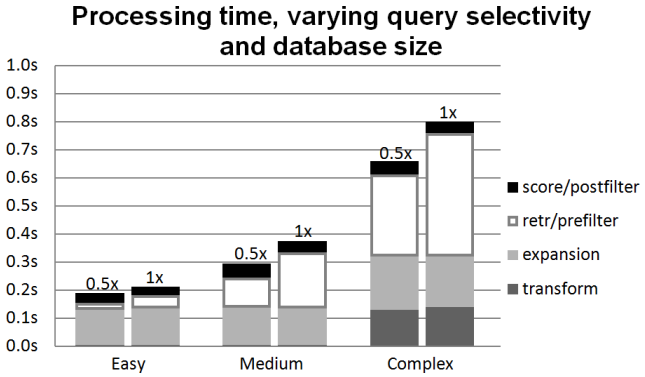

Figure 7: Time breakdown for Table 1 queries showing step cost. Also shows change in cost when doubling the database size.

was the slowest part. However, it is also the step that can be most improved because our prototype uses a $O(\log n)$ index rather than $O(1)$. Finally the top stack, representing the scoring, location, personalization and postfiltering together, was equally fast because the prefilter controlled how many candidates reach the scoring step.

Note that some user tasks on a mobile, such as typing messages, take longer than 1s and therefore the phone may sometimes have more time available to retrieve a result. Comparing the 50% and 100% stacks indicates that doubling the size does not double execution time, and we can extrapolate to handling Complex queries within 1s on a dataset of 9k. In particular, only the retrieval and prefilter step cost changes and this is the step for which performance can most improve when we complete the development on an iPhone using mContext's indexing platform.

5. RELATED WORK

Mobile advertising. To our knowledge, in the domain of mobile marketing the most similar systems are proposals in patent applications.

Western Digital [21] specified that image ads should be stored on a mobile and shown when browsing websites. Initially ads are saved on the phone and later the cached copy is displayed instead of downloading the ad again. The goal was bandwidth preservation. The selection process was entirely predetermined: ads were shown in 'reserved space' on participating websites and the site had to specify which type of image should be shown.

SanDisk [19] proposed for ads to be stored on a SIM card and be selected by the phone for various activities. This idea is similar to ours, but different in the selection process. In their case, ads were selected based on predetermined call events (incoming, outgoing, alarm clock) and specific values being read from the phone's sensors (time, temperature, perspiration level, accelerometer, etc.). Exemplary uses were a flower shop ad shown for an incoming call from a known address book entry and a pizza ad shown for an outgoing call at night. Such triggers are non-intuitive to specify and their correlation to an ad's relevance is unclear. Apart from accuracy, scaling to more ads may also be problematic and worsen user experience due to skewed popularity of triggers, e.g. a 'late at night' time trigger may be excessively popular and show ads that appear unrelated to each other, while a perspiration trigger may be unpopular and always show the same ad and frustrate the user. In our system ads are shown in direct response to viewed content instead of sensor readings, which makes them more relevant. We also use a dynamic selection process such that any one ad may be shown for multiple input scenarios rather than a single predetermined, physical trigger. Physical triggers were earlier proposed by Outland Research [15], where ads were downloaded from the web to be shown alongside mobile web search and their display was triggered by weather conditions, temperature, time, position, heart rate, blood pressure, etc. The selection processes in these proposals are rigid and non-intuitive, which may lead to low advertiser participation and a weak user experience. In contrast we use tags, which millions of 'advertisers' already use online.

Online advertising. Some ideas in our framework relate to online advertising, in particular that ads are labeled with words and both indexing and selection relies on these words. However, there are several domain differences that make it difficult to compare against work on online advertising. In sponsored search, multiple data sources are used to determine ad relevance: by typing a query the user immediately reveals highly useful information to the search engine, which then uses the query and various historical data associated with it [4]. But for VAS there is no user query and historical data is not available unless the phone continuously contacts outside sources. Contextual advertising is closer to our work because there is no user query, but the online systems use numerous web-specific features that do not apply to our case, such as positional effects, anti-fraud measures, search query logs, URL analysis, anchor text, link analysis, historical searches and their CTRs, and content merging of related websites [20, 22, 13]. A much more lightweight approach is needed for mobiles, but light methods are generally ignored by the IR community because of the assumption that servers will do the work (a valid assumption for online advertising).

Ribeiro-Neto et al. [18] provide something closer. Several of their match strategies use only ads and webpage content. They find that both query expansion and additional supporting text (e.g. ad's title and abstract) are helpful. We do not directly compare with their work because their domain and application is different. While they also match ads to content, their content is webpages, which are long documents where more expensive calculations such as cosine produce results. In our case, the typical 'documents' are very short (e.g. SMS, a song title, description of a clip) and we often have only a handful of keywords to use. This means that we need to use query expansion straight away yet keep the 'document' lengths short to satisfy the processing constraints. Additionally, their work is for the specific case of websites and they use specific data that does not apply in the generic case, e.g. their best strategy used hyperlinks and expanded the webpages to even larger sizes with content from related pages. Finally, their strategies relate to the expansion (and partially to the scoring) step in our selection process, but not to the other steps.

Aside from relevance, much of online advertising literature focuses on the complex game theory of bid optimization (e.g. [14, 11, 7] for some recent results). In this paper we focus on timing and accuracy but otherwise largely ignore any details of billing. We recognize bid optimization as a complex problem and a distributed system, where clicks happen on individual mobiles and are reported to the ad-broker with

Set	Description of query set	|Input|	|Qry exp|	|Candidates|	Net time
Easy	Random tags from ad-db index, e.g. *nederlands*	1 word	12.3 tags	172 (σ=198)	0.26s (σ=.07)
Medium	Delicious top-30 popular tags, e.g. *science*	1 word	37.3 tags	878 (σ=330)	0.48s (σ=.07)
Complex	Online news stories, e.g. 'Sampras tips Federer to win Wimbledon. *Tennis great Pete Sa...* [320 words]'	280 (σ=72)	119 tags	1386 (σ=165)	0.85s (σ=.10)

Table 2: Summary of three sets of queries on D-db that increase the number of candidates prior to prefilter. The step costs for these queries is shown in Fig 7.

hours or days of delay, may not make it any easier. We do expect, however, that a billing process can work at least sufficiently well in practice by encoding the bid and unspent budget into the tag weights. The weights directly affect selection, so reducing these values as the budget decreases should somewhat address the delay problem. Web advertisers generally specify daily budgets but in our case a weekly or monthly budget should be more suitable.

Another related work from the web domain is Penev and Wong [16], which looked at using tags to approximate pairwise bookmark similarity in a pool of Delicious bookmarks. Using information extracted from bookmarks to accompany the users' tags, several strategies were tested to improve the correlation of the tags-tags comparison to the full document-document comparison.

We are currently implementing two additions to the prototype, the first of which is handling keyphrases and the second is a classification step after the query expansion. Classification can help guess the likely topic of the input and consequently provide an additional supporting feature. Classification has been shown to be useful for web advertising (e.g., Broder et al. [9]) and we are experimenting with a mobile-suitable, entropy-based method using the Wikipedia topics extracted by Phan et al. [17].

6. CONCLUSION

We describe a framework for mobile advertising that addresses the timing and accuracy problems of current mobile marketing approaches. Our framework borrows ideas from the state-of-the-art in targeted advertising—that of web search engines—to provide an efficient filtering and scoring method for generic mobile content such as Value-Added Services. We provide experiments to show that selection is fast, scalable and that given well-specified ad data has good accuracy. Our contributions are the introduction of the phone-as-a-search-engine problem and our selection method, which runs under the heavy constraints imposed by mobiles and which can serve as a baseline for future research.

Although this paper talk about mobile phone service providers and content, the concepts have wider applications. Consider that some devices, such as a Wi-Fi Netbook and iPod Touch, do not communicate with service providers but still display content. An ad agent similar to ours can enable advertising on such devices. It is also easy to draw comparisons to web prefetch technologies like Google Gears and AJAX, whose aim is to store content on the client in order to make it more responsive and improve user experience. For these reasons this paper may be of interest to both advertisers and manufacturers of small devices.

Ultimately, we hope to share ideas on the future of mobile-enabled marketing and discuss an approach for improving the advertising effectiveness for all parties involved.

7. REFERENCES

[1] http://www.google.com/ads/research/gcnwhitepaper.
[2] http://investor.google.com/fin_data.html.
[3] https://adwords.google.com/select/KeywordToolExternal.
[4] http://adwords.google.com/support/bin/answer.py?answer=10215.
[5] *The Netsize Guide 2008*. http://www.netsize.com.
[6] Bauer, H., Barnes, S., Reichardt, T., and Neumann, M. Driving Consumer Acceptance of Mobile Marketing. *JECR 6*, 3 (2005), 181–192.
[7] Borgs, C., Chayes, J., Immorlica, N., Jain, K., Etesami, O., and Mahdian, M. Dynamics of bid optimization in online advertisement auctions. In *WWW* (2007).
[8] Broder, A., Ciaramita, M., Fontoura, M., Gabrilovich, E., Josifovski, V., Metzler, D., Murdock, V., and Plachouras, V. To swing or not to swing: learning when (not) to advertise. In *CIKM* (2008).
[9] Broder, A., Fontoura, M., Josifovski, V., and Riedel, L. A semantic approach to contextual advertising. In *SIGIR* (2007).
[10] ComTel/Empowered Communications. Making mobile advertising a rewarding reality. Media briefing and market release, 25 Aug 2008.
[11] Even Dar, E., Mirrokni, V. S., Muthukrishnan, S., Mansour, Y., and Nadav, U. Bid optimization for broad match ad auctions. In *WWW* (2009).
[12] Giuffrida, G., Sismeiro, C., and Tribulato, G. Automatic content targeting on mobile phones. In *EDBT* (2008).
[13] Google Inc. Methods and apparatus for serving relevant advertisements. US2004/0059708, 2002.
[14] Mehta, A., Saberi, A., Vazirani, U., and Vazirani, V. Adwords and generalized online matching. *JACM 54*, 5 (2007).
[15] Outland Research. Method and apparatus for improving the matching of relevant advertisements with particular users over the internet. US2006/0206379, 2006.
[16] Penev, A., and Wong, R. K. TagScore: approximate similarity using tag synopses. In *Web Intelligence* (2008).
[17] Phan, X.-H., Nguyen, L.-M., and Horiguchi, S. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *WWW* (2008).
[18] Ribeiro-Neto, B., Cristo, M., Golgher, P. B., and Silva de Moura, E. Impedance coupling in content-targeted advertising. In *SIGIR* (2005).
[19] SanDisk. Method for advertising on mobile devices. US2008/0108337, 2007.
[20] Wang, X., Broder, A., Fontoura, M., and Josifovski, V. A search-based method for forecasting ad impression in contextual advertising. In *WWW* (2009).
[21] Western Digital. Caching advertising information in a mobile terminal to enhance remote synchronization and wireless internet browsing. US6826614, 2001.
[22] Yih, W.-t., Goodman, J., and Carvalho, V. R. Finding advertising keywords on web pages. In *WWW* (2006).