SEVENTH FRAMEWORK PROGRAMME

Networked Media

Specific Targeted Research Project

SMART

(FP7-287583)

Search engine for MultimediA environment generated contenT

D5.3.b Query Scoring and Anticipation Subsystem

Due date of deliverable: 01-07-2013

Actual submission date: 22-07-2013

Revised submission date: 16-01-2014

Start date of project: 01-11-2011 Duration: 36 months

Summary of the document

Code:	**D5.3.b Query Scoring and Anticipation Subsystem-v2.06**
Last modification:	15/01/2014
State:	
Participant Partner(s):	GLA
Author(s):	Dyaa Albakour, Craig Macdonald, and Iadh Ounis
Fragment:	No
Audience:	☒ public ☐ restricted ☐ internal
Abstract:	This deliverable is the public report corresponding to D5.3a. It provides public information relating to this deliverable, treating confidential/proprietary information as a black-box
Keywords:	• Event retrieval • Filtering • Anticipation • Recommendation

1 Executive Summary

1.1 Scope

The SMART Search Layer indexes streams of physical sensor metadata and social posts such that real-world events can be retrieved in response to a user query. This document is a public cut-down version of the description of the retrieval models used in SMART to identify and rank events that match a user query as inferred from the sensor metadata and the social posts indexed in the search layer. Moreover, the document describes the anticipation and recommendation models we built to predict locations of interest to a user within a certain context identified by the time and the location of the user. The full version of this report may be obtained by contacting the authors at the University of Glasgow.

1.2 Audience

The content of this deliverable can be of interest to a wide range of individuals within these groups:

- **The Information Retrieval (IR) community**: The challenges addressed in this deliverable are relevant to IR researchers and practitioners. Our solutions advance that-state-of-the-art in proposing novel information retrieval problems and in modelling sensor and social streams for local event retrieval.
- **SMART software developers**: The developers of the SMART project, notably those dealing with the open source software implementation of SMART components in the search layer and SMART applications/visualisation libraries.
- **SMART project members:** The deliverable gives insight to all project members involved in delivering the SMART concept. Notably, members involved in the development of WP3, WP4 and WP6 can better understand how the Search layer models the information streamed from the social and sensors streams in order to retrieve and rank events in response to user queries.
- **The Open source community:** The open source community, notably the community that we are building around the SMART results, will use the present deliverable as a guide towards understanding the of the retrieval and recommendation capabilities of the search layer, as well as its effectiveness performance.

1.3 Summary

In this public deliverable, we give an overview of the retrieval, filtering and anticipation capabilities in SMART.. The search layer adds an added value to sensor and social feeds describing the environment as it combines the evidence from these feeds to locate, identify and rank events that match the user interests. This document identifies the challenges imposed by the nature of the streams of sensor and social observations on the retrieval and anticipation models. It also describes the models we have developed to tackle these challenges. The deliverable also summarises the results of the evaluation conducted to measure their effectiveness.

1.4 Structure

The deliverable is structured as follows:

- Section 2 provides an overview of the query scoring and anticipation techniques in SMART.
- In Section 3, we present our local event retrieval framework of our query scoring subsystem.
- In Section 4, we present the Twitter filtering functionality of our query scoring subsystem
- Section 5 introduces the time-series modeling and forecast we adapt for the anticipation subsystem.
- Finally, Section 6 summaries the conclusion and gives an outlook for the next version of the deliverable.

2 Overview

The Search Layer in the SMART framework is composed of the software components that provide services and applications with easy *real-time* access to information stemming from both the social and sensor media streams. In this deliverable we have developed two major subsystems within the Search Layer: (1) the query scoring and (2) the anticipation subsystems where we mainly focus on the effectiveness of the retrieval and recommendation from sensor streams in SMART.

The *query scoring* subsystem provides two main functionalities:

i. *Local Event Retrieval:* Identifying and ranking local events in response to an explicit user query from observations across different locations produced in real-time from multiple edge nodes. In Section 3, we describe the local event retrieval framework that we have developed. More details about the framework and the evaluation conducted using social media data can be found in a recently published paper [Albakour2013a].
ii. *Filtering*: Supporting "running queries", where the user is notified with new relevant events or social media updates as they happen (as they are received and indexed). In Section 4, we describe our model for tackling the challenges in real-time tweet filtering. More details about the filtering model and the evaluation conducted can be found in a recently published paper [Albakour2013b].

The *anticipation* subsystem in the SMART Search Layer aims to anticipate events in the future and provide the user with recommendations on activities to do or locations to visit, based on the context of the user i.e. his/her location and time. In section 5, we describe the anticipation model we have developed to mines temporal patterns of sensor observations about the environment collected by SMART Edge Nodes in order to predict future observations in a certain location at a certain time in the future

3 Local Event Retrieval

Local Event Retrieval allows end users and applications to retrieve a ranked list of local events for explicit keyword queries. In this deliverable, we have devised a novel event retrieval framework that is capable of identifying and ranking local events in a response to a user query. In this section, we first give a formal description of the local event retrieval problem (Section 3.1) and then we present our event retrieval framework (Section 3.2). Finally, we provide a summary of the evaluation we conducted to measure the effectiveness of the retrieval framework (Section 3.3).

3.1 Problem Formulation

The major functionality of the SMART search layer is to identify and rank local events happening in the real world as a response to a user query. For a formal definition of a local event, we adopt a definition that has been previously used in the TDT new event detection task over broadcast news [Allan1998]. This definition states that an event is something that occurs in a certain place at a certain time. We also make the following assumptions:

- An event occurs at a location at which SMART sensors are present.
- The SMART sensor produces metadata stream which can indicate an occurrence of an event.
- An event has a starting time and an end time.
- Multiple events cannot occur at the same location at the same time.

Formally, we consider a set of locations $L = \{l_1, l_2, ..., l_n\}$ that are of interest to the user. The granularity of locations can vary from buildings and streets to entire cities. For example, we might consider each location to represent an area in a city in which the user is located. The city in this case is considered to be divided into equally sized areas specified by polygons of geographical coordinates, or we can use the divisions defined by the local authority such as postcodes or boroughs. Each location l_i at a certain time t_j is denoted by the tuple $\langle l_i, t_j \rangle$.

We define the problem of *local event retrieval* as follows. For a user interested in local events within locations L (explicitly defined or implicitly inferred from the current user's location), the event retrieval framework aims to score tuples $\langle l_i, t_j \rangle$ according to how likely t_j represents a starting time of an event within the location l_i that matches the user query. An event is considered relevant if it matches the explicit query of the user and/or the implicit context of the user (the time of the query, the location of the user and or her profile).

In other words, the event retrieval framework defines a ranking function that gives a score $R(q, \langle l_i, t_j \rangle)$ for each tuple $\langle l_i, t_j \rangle$ with regards to the user's query q. In SMART, we use the social and sensor streams collected from edge nodes for *local event retrieval*, targeting local events happening in a city. Examples include festivals, football matches or security incidents. When expressed explicitly by a user, a query is assumed to be in the form of a bag of words (e.g. “live music”, “conference”).

The observations from the social and sensor streams collected from a location l_i at a certain time t_j within a given time frame $t_j - t_{j-1}$, denoted by $O_{i,j}$, reflect what is actually happening in that location at that particular time. Note that the fixed time frame is defined using an arbitrary sampling rate θ; $\forall j: t_j - t_{j-1} = \theta$. An event happening in the real world is represented by a tuple $\langle l, t_s, t_f \rangle$; where l is the location of the event is taking place, t_s is the starting time and t_f is the finishing time.

Our aim is to use the social and sensor observations as the main source of evidence to define the ranking function $R(q,\langle l_i,t_j\rangle)$. More specifically and to define the ranking function, we use the set $O_{i,j}$ of sensor observations and social media posts originating from that location shared publicly within the given time frame $t_j - t_{j-1}$, and also a time series of social and sensor observations in the location l_i prior to the current time t_j: $\langle\ O_{i,j-k},..,O_{i,j-1},O_{i,j}\ \rangle$. This allows us to identify sudden increases in the sensor observations, which may have been triggered by an occurrence of an event (e.g. increasing crowd level, increasing tweeting activity).

3.2 Framework for Local Event Retrieval

In this section, we describe our event retrieval framework. The framework aims to define an effective ranking function that scores tuples of time and location according to how likely they represent the starting time and the location of a relevant event for a given query.

Note that with regards to the previous definition of the *local event retrieval problem* in Section 3.1, as a first step, we are not aiming to determine the finishing time of an event. We recognise that for some applications the finishing time of an event may be important, e.g. surveillance applications, however we leave this for the second version of the deliverable.

As discussed in Section 3.1, we aim to use the social and sensor streams as the main source of evidence to score the tuples. In particular, we define two components built on this evidence:

1. The Topical Component: The first component is based on the intuition that social media may reflect real world events, hence when an event occurs somewhere we expect to find topically related social posts about it originating from the location where it occurs. To instantiate this component, for each location at a given time, i.e. for each tuple $\langle l_i,t_j\rangle$, we measure how much the social media posts (e.g. tweets $T_{i,j} \subseteq O_{i,j}$) corresponding to the tuple are topically related to the query q.

   Moreover, this component could also be instantiated by measuring how much the topics of the metadata describing the sensor observations are related to the user query. Extracting metadata describing sensor feeds are discussed in the other deliverable of WP5 (D5.2). In this version of the deliverable, we do not look into this aspect and we only instantiate the topical component from the social media feeds.
2. The Change Component: The second component is based on the intuition that events trigger an *increasing* activity measured by the various sensors (both physical and social sensors) that is also *unusual*. For example, a sudden increase in the crowd level observed in a certain area that is not anticipated or an increasing tweeting activity causing peaks of tweeting rates during the event (bursts) [Zubiaga2011]. For this component, we aim to quantify the *unusual change* in the sensor observations. For instance, the volume of tweets over time, observed at $\langle l_i,t_j\rangle$ when compared to previous observations over time at the same location. In other words, we aim to measure the unusual behaviour that may indicate an occurrence of an event.

Following this, and using tweets for example as the source of social media posts, the ranking function can be defined as a linear combination of the previous two components as follows:

$$R(q,\langle l_i,t_j\rangle) \propto (1-\lambda)\cdot S(q,T_{i,j}) + \lambda \cdot E(q,\langle l_i,t_j\rangle) \quad (1)$$

where $S(q,T_{i,j})$ is the score of the tweet set $T_{i,j}$ that quantifies how much they are topically related to the query q; $E(q,\langle l_i,t_j\rangle)$ is a score proportionate to the probability that an event is about to start in the location l_i at time t_j as can be indicated from the sensor observations, and $0 \leq \lambda \leq 1$ is a parameter to control the contribution for each component in the linear combination in Equation (1). We experimented with different approaches to instantiate both components of the framework, for example we have presented how these components can be effectively implemented on tweets streams in [Albakour13a].

3.3 Summary of the Evaluation

Using our evaluation methodology for the local event retrieval task [Albakour2013a], we thoroughly evaluate our event retrieval framework using two different types of sensor feeds:

- The first evaluation is on large-scale geo-located tweets from four different boroughs as well as from the entire metropolitan area of London over a period of twelve days. London is chosen because it is a big vibrant city and it is among the top cities in the world in terms of the Twitter user population. We use two different sets of queries, one collected using crowdsourcing and another one collected from local news feeds in each of the given four boroughs. The results are promising, showing the effectiveness of the framework in using tweets as social sensors to correctly identify the time and the location of events. Our empirical results suggest that detecting local events using geo-located tweets is feasible but difficult. In particular, the results show that our event retrieval framework is capable of identifying and ranking events within a city. However, when applied on multiple fine-grained areas within the city, the retrieval effectiveness of the framework degrades, possibly because of the nature of the events considered in our experiments, i.e. their low coverage on Twitter.
- In the second evaluation, we aim to evaluate different approaches for quantifying the change component when using crowd analysis data from video feeds. The dataset used in this experiment is the crowd analysis data provided by AIT performed on indoors videos taken within a lab in their premises. One person manually annotated the data by defining a starting point of a potential event and the duration of each event. This has resulted in 21 events in total varying in duration between 3 minutes and 71 minutes over a period of 2 weeks. The results suggest that modeling background information from previous observations (what is usually observed in a certain location at certain time in a day or a week) can help to better estimate the change component. It has resulted in better precision of the retrieval model, which shows its power to distinguish unusual events from expected ones.

4 Real-time Tweet Filtering in SMART

The SMART Search layer supports "running queries" where the user is subsequently notified in real-time with new relevant events or social media updates as they happen (as they are received and indexed). In this section, we develop a filtering model, as part of the query scoring subsystem, for the real-time filtering social media posts in Twitter (tweets), since Twitter is one of the most popular and fastest growing social media platforms. In particular, we tackle the specific challenges in real-time tweet filtering that do not necessarily exist in traditional filtering domains. We first provide a formal description of the real-time tweet filtering problem and we describe the news filtering approach that we build on (Section 4.1). We then describe the sparsity challenge and our query expansion approach to deal with it (Section 4.2). Finally, we summarise the evaluation conducted to assess the effectiveness of our filtering model. Further details on the implementation of the model and the evaluation conducted are published in [Albakour2013b].

4.1 Real-time Tweet Filtering

Adaptive filtering was previously investigated in the TREC Filtering track [Robertson2002] within the news domain. In adaptive filtering, and unlike a traditional search query, user information needs reflect a long-term interest and they are represented in a user's profile, which is usually a set of documents considered relevant by the user [Belkin1992]. Moreover, instead of searching a static document collection, an adaptive filtering system examines a stream of incoming documents over time and decides for each document whether it matches the user's profile such that it is displayed immediately to the user [Allan1996]. Generally, the filtering system starts with a user profile and a very small number of positive feedback examples (relevant documents). The profile can then be adapted using the feedback information provided by the user for the displayed documents [Belkin1992]. The problem we are focusing on in this deliverable is the real-time filtering of tweets. In the remainder of the section, we first provide a formal description of the real-time filtering problem. We then describe the state-of-the-art news filtering methods that we build on for this problem.

4.1.1 Problem Formulation

We instantiate the problem of real-time tweet filtering as an instance of the adaptive filtering problem. Given a user u with an initial information need, i.e. an input query q, at a certain starting time t_s and a small set of positive example tweets prior to t_s, the filtering system should decide whether subsequent tweets posted after t_s are relevant to the user and therefore should be displayed to the user. This should allow the user to stay updated in real-time by browsing relevant tweets for a developing topic. The user can examine these tweets and provide explicit feedback to the filtering system whether they are relevant or not. The filtering system can hence adapt the user's profile, which is defined to be the set of feedback tweets provided by the user. Indeed, Twitter already implements a filtering functionality ("X new tweets") in its search page,[1] however it does not allow the user to provide explicit judgments and, to our knowledge, there is no study published on the effectiveness of their filtering tool.

Similarly, in SMART, the user submits a query q, at a certain starting time t_s and retrieve back a ranked list of local events that has started prior to t_s. SMART should identify subsequent tweets posted after t_s that are relevant to the user (the local events he is interested in) so that the user is notified with these tweets.

4.1.2 Filtering with a Text Classifier

We introduce two effective adaptive filtering methods on news that can be employed on tweets, namely Incremental Rocchio [Allan1996], and Regularised Logistic Regression [Zhang2004a]. The Regularised Logistic Regression has shown to outperform Incremental Rocchio in adaptive news filtering [Yang2005, Zhang2004a]. However, the thorough evaluation we conducted, reported in [Albakour2013b], have shown the opposite for tweet filtering. This is due to the generative nature of incremental Rocchio, which may well suit adaptive filtering in the highly dynamic nature of Twitter in comparison to the more complex discriminative regression learning approach which typically requires a

[1] https://twitter.com/search/

large number of training examples to have a stable performance. Therefore, we will formally describe the Incremental Rocchio approach that we build on to tackle the challenges of real-time tweet filtering.

Incremental Rocchio is based on a classifier that uses the popular Rocchio's relevance feedback approach to build a profile of the user's interests, which is then updated online using the explicit judgements provided by the user [Allan1996]. More specifically, at each point of time, t, the profile of the user is represented in the term vector space model by a vector $\vec{c}_t$, which is called the *centroid* of the user's interests. The centroid is calculated as follows:

$$\vec{c}_t = \frac{\alpha}{|R_t|} \cdot \sum_{d_i \in R_t} \vec{d}_i - \frac{\beta}{|N_t|} \cdot \sum_{d_i \in N_t} \vec{d}_i \quad (2)$$

where R_t is the set of tweets judged *relevant* by the user so far (at time t), N_t is the set of tweets judged *non-relevant* by the user so far (at time t), α and β are co-efficient parameters for positive and negative feedback documents respectively. For each incoming tweet, $\vec{d}$, the cosine similarity is computed between the centroid at the time at which the tweet arrives and the actual tweet $Sim(\vec{c}_t, \vec{d})$. If the cosine value $Sim(\vec{c}_t, \vec{d})$ exceeds a certain threshold η_R, the tweet is considered relevant and is therefore displayed to the user, otherwise it is not. When the tweet is judged relevant by the user it will be added to the set of relevant tweets for the next time point R_{t+1}, otherwise it will be added to N_{t+1}, and as a result the centroid $\vec{c}_{t+1}$ will be updated. Our initial experiments have revealed that penalising negative documents does not improve tweet filtering effectiveness and hence we only consider positive feedback documents (β), i.e the centroid is reduced to:

$$\vec{c}_t = \frac{1}{|R_t|} \cdot \sum_{d_i \in R_t} \vec{d}_i \quad (3)$$

The terms in the vector space are weighted using any appropriate term weighting models such as BM25 [Robertson2009].

4.2 Handling Sparsity

The acute sparsity issue in filtering tweets is a unique challenge caused by the shortness of tweets. As discussed before, this problem is less prevalent, for instance, in traditional filtering tasks, such as adaptive filtering of news streams for which the Incremental Rocchio method we build on was originally designed. Sparsity is particularly problematic when we know little about the user's interests or the topic itself, i.e. when the filtering system knows only a small number of documents (tweets) that the user is interested in. As a result, the centroid of the Rocchio's classifier has a small number of terms, which may not be representative of the topic or the user's interests.

To tackle the sparsity of the initial centroid representation in incremental Rocchio, our approach is to make use of query expansion (QE), a traditionally successful method for improving the retrieval performance in a number of IR tasks, such as adhoc retrieval [Xu1996]. In a *static* document collection, QE automatically derives terms that can be added to a user's original query from a pseudo-relevant set of documents (the initial set of highly ranked documents retrieved for the query). We apply QE as a mechanism to enrich our knowledge of the topic and the user's interests by deriving terms that are both topically relevant and temporally correct. For this, and to initialise our classifier, we apply query expansion using the user's query and a document collection comprising tweets posted publicly up to the time of issuing the query or triggering an interest in a tweet (by providing an explicit positive feedback).

Formally, at a certain time t_i and for a user with a query q, we use a set of tweets that were posted before t_i. We denote this set of tweets by T_S. T_S could be limited to an arbitrary number of tweets before t_i or an arbitrary period of time. We can then apply QE to score terms in the pseudo-relevant set of tweets $T_q \subseteq T_s$. We denote this set of terms by E. In particular, the centroid of the classifier at a certain time t can be computed as follows:

$$\vec{c}_t = \frac{1}{|R_t|} \cdot \sum_{d_i \in R_t} \vec{d}_i + \vec{e} \quad (6)$$

where $\vec{e}$ represents a vector that is comprised of all the terms in the expansion set E weighted with their scores provided by the QE weighting model. This vector can be expressed formally as $\vec{e} = \sum_{e_i \in E} w(e_i, q).\vec{e_i}$, where $\vec{e_i}$ is the standard basis vector for dimension (term) e_i.Note that $\vec{e}$ is particularly useful when the set of positive feedback documents R_t is small, e.g. only one tweet, which is the main purpose of adding this component to the centroid.

Furthermore, we investigate adding the entire set of pseudo-relevant tweets to the centroid. In this case, the vector is calculated as follows:

$$\vec{c}_t = \frac{1}{|R_t|} \cdot \sum_{d_i \in R_t} \vec{d_i} + \frac{1}{|T_q|} \cdot \sum_{d_i \in T_q} \vec{d_i} + \vec{e} \quad (7)$$

As for the documents in R_t, each document in T_q is weighted with an appropriate term weighting model and the weights are normalised. Note that the terms in the vector $\vec{e}$ are a subset of the terms in T_q.

4.3 Summary of the Evaluation

Using the real-time filtering task of the Microblog track of TREC 2012, we thoroughly evaluate our adaptations against the standard Incremental Rocchio and a state-of-the-art news filtering technique based on Regularised Logistic Regression. Our empirical results, which have been published in [Albakour2013b], show that the traditional news filtering techniques are not effective in the context of tweet filtering. They also show the power of the adaptations we propose to mitigate the problem of sparsity in Twitter. Using our approach to tackle the sparsity of tweets yields a significant improvement in the overall filtering effectiveness estimated by the official TREC filtering measures (T11SU and F_0.5). In addition, our adaptation with QE outperforms the best TREC 2012 run using the F_0.5 measure.

5 Anticipation in SMART

The anticipation subsystem in the SMART Search Layer aims to anticipate events in the future and recommend to the user activities to do or locations to visit based on the context of the user i.e. their location and time. To do so, the anticipation subsystem mines temporal patterns of sensor observations about the environment collected by SMART Edge Nodes in order to predict future observations in a certain location at a certain time in the future. In particular, the temporal patterns have two important aspects that we can model in order to make such predictions: (a) *trend* and (b) *seasonality*.

a. *Trend*: certain locations can become increasingly more popular (crowded) over time or less popular. This happens when a certain event is taking place. For example, if a festival is taking place during the day in a city's square, people will be increasingly arriving to the city's square in the morning and they peak during midday. After that they gradually leave the square.
b. *Seasonality*: activities that people do vary depending on the time of the day, the week or the year. For example, shopping activities take place during the day, while theatre displays usually takes place in the evening. Moreover, bars and night clubs are usually busy in the weekends and not on working days.

In this section, we develop our model for anticipation in SMART. In particular, we apply time series modelling techniques that model trend and seasonality patterns to make a forecast for future observations in a certain location. The forecast allows us then to be able to make predictions of popular locations or events at a certain location in a certain time in the future. Therefore, we can use these predictions to anticipate and recommend locations of interest to the user who has not explicit query but rather a context (when and where they are). Section 5.1 introduces our time series modelling and forecast technique and it describes how we perform these models on sensing data from the FourSquare[2] location-based social networks. In Section 5.2, we describe how we can use the anticipation model to recommend locations that may be of interest to the user context. Finally in Section 5.3, we give a summary of the evaluation conducted to measure the accuracy of our forecasting models in predicting venue occupancy levels using the FourSquare social network.

5.1 Time-series Modelling and Forecast with FourSquare

A time-series consists of time ordered data points at uniform intervals. SMART Edge Nodes produce time-series data of measurements about the environment in the location they are deployed in. This includes the time-series of the crowd-level, the noise or the tweeting activity in a location. We aim to use time series analysis to model the temporal changes in certain locations and forecast future trends. In our modelling, we aim to represent the two important aspects in temporal patterns introduced earlier, namely *trend* and *seasonality.* We apply the *exponential smoothing* (HoltWinter methods) [Holt2004] for time series modelling of our Edge Node, which represents both the trend and seasonality components, which are represented in the model. In a nutshell, the HoltWinter model takes a time series of real-world observations represented by real numbers e.g. crowd density, noise level, etc, and for a given time in the future it is capable to predict the real-world observation at that time.

We experimented with our anticipation model on sensor observations produced by FourSquare. Foursquare has recently emerged as a location-based social network that allows users with mobile devices to share their location with their friends and update their social media profiles, e.g. on Facebook and Twitter, with this information.

We use FourSquare API to obtain a time series data of the venues within the urban area of a city, where we have a database of venues and a time series for each that gives the occupancy of the venue at different points of time. The occupancy is estimated by the FourSquare social network using the number of actual user "check in"s in the venue.

Using the time series modelling with HoltWinter and the time-series of occupancy levels for each venue collected from Foursquare, we can predict the occupancy level of a certain venue at a certain point in

[2] http://foursquare.com

the future. Considering the set of all venues in a city V, each venue $v \in V$ has a time series of occupancy level $\omega_{v,t}$. At certain time t, we can estimate the occupancy level of the venue $\widehat{\omega}_{v,t}$ using the Holt-Winter model.

5.2 Contextual Recommendation by Anticipating Locations

We develop a ranking function that can recommend venues to a user in a certain context (location and time). Formally, given a user u with zero-query (no explicit query) in a context identified by $\langle l, t\rangle$ (the user is located in location l identified by the geographical coordinates of that location (longitude and latitude) and the time of the query, t), we can first predict the occupancy level for all venues that are close to the user and develop ranking function that uses the occupancy level of those venues as an input. Here we will investigate an intuitive, yet effective approach to develop this ranking function, which simply uses the occupancy level to rank the venues,

$$R(v, \langle l, t\rangle) = \begin{cases} \widehat{\omega}_{v,t\ \mathrm{i}}; \ if\ v \in Z(l) \\ 0; \ if\ v \notin Z(l) \end{cases} \quad (8)$$

where $Z(l)$ is the set of venues located in the user's zone, which we limit to a circle with a radius of 2000 meters centred around the location of the user l.

In Table 1, we show two different rankings generated for a user at the same central location in London for two different times (Friday afternoon and Saturday night). These rankings are produced using past data over a period of two weeks between the 20th of March and the 2nd of April 2013.

Table 1 Examples of venue recommendations produced by our model for user in a central location in London at two different times

Friday 03 April 14:00	Sunday 5 April 00:00
Debenhams	Novikov Restaurant & Bar
Natural History Museum	Boujis(nightclub)
Selfridges & Co	
National Gallery	
Apple Store	
London Victoria Railway Station	
Victoria and Albert Museum (V&A)	
Millbank Tower	
Science Museum	
Piccadilly Circus	

5.3 Summary of Evaluation

We have conducted an experiment to evaluate the forecast accuracy of a venue's occupancy level using the time series modelling introduced in Section 5.1 and compare it to alternative baselines. We collected occupancy level data for trending venues in London. The data was collected between the 20th of March and the 23rd of April 2013. We use the first four weeks of the data as training data, and test the forecast accuracy on the fifth week (17th to the 23rd of April). We experimented with different cycle periods for the HoltWinter forecast (daily 24 hours, and weekly 168 hours). The results show that using weekly cycles significantly outperform the daily cycles for accurately predicting the venue's occupancy level. A naive baseline that predicts the current occupancy level based on what is observed an hour earlier outperformed our model, however our approach is superior as it is capable of predicting well in advance (e.g. a week later).

6 Conclusions and Next Steps

In this deliverable, we have developed the core components of the query scoring and anticipation subsystem of the Search Layer in SMART.

Local Event Retrieval is a major functionality of the query scoring subsystem in the SMART Search Layer. It allows end users and applications to retrieve a ranked list of local events for an explicit keyword queries. To develop this component, we have devised a novel event retrieval framework that is capable of identifying and ranking local events in a response to a user query. The retrieval framework combines evidence from the content of geo-tagged social media content and the change in the sensor observations (both physical and social) to accurately identify and rank local events. For the next release of the deliverable, we aim to extend the framework to deal with the caveats we observed in our evaluation. For example, when considering small areas in a city and to tackle the sparsity issue, tweets in nearby locations can also be exploited to improve the ranking function (e.g. by using the tweets in those locations for smoothing). Most importantly, we will evaluate our framework on physical sensor observations using the data that is being collected in the main city square of Santander. In addition, with the availability of such data, we aim to explore Learning-to-Rank approaches that have the power to harvest the features of the metadata in order to effectively identify and rank events.

The real-time filtering of tweets is another major functionality of the query scoring subsystem. Real-time tweet filtering is an emerging IF task that can still be tackled using traditional filtering techniques with appropriate adaptations to address the unique challenges prevalent in Twitter. Our thorough evaluation, using a TREC collection, shows that traditional state-of-the-art news filtering techniques are not as effective when applied on tweets. However, the modifications we proposed on a traditional news filtering approach have mitigated the acute sparsity issue that is prevalent in Twitter. In particular, our QE approach to tackle the sparsity of tweets yields significant improvement on the filtering effectiveness by deriving a richer representation of the user profile with relevant and temporally correct terms. For the next release of the deliverable, we aim to explore using dynamic knowledge resources, such as the Wikipedia, on a timely manner to complement our QE approach. This may allow the classifier, for example, to capture terms that are not picked up by QE, and hence obtain a richer representation of the user's information needs.

Finally, we presented and evaluated a model for anticipation in SMART that mines previous patterns of sensor observations in order to make future predictions and anticipates locations of interest to an end user. Using the occupancy level of venues generated from user "check in"s in the location-based social network of Foursquare, we show how such model can be applied to recommend venues in certain location a certain time. Our evaluation also uncovers important factors when modelling the seasonality patterns of venue occupancy observations. In the next release of the deliverable, we aim to extend our anticipation model to consider multiple sensor and social feeds and evaluate it on physical sensor observations using the data that is being collected in the main city square of Santander.

7 References

[Albakour2013a] M. Albakour, C. Macdonald, and I. Ounis. Identifying local events by using microblogs as social sensors. In *Proceedings of OAIR,* 2013.

[Albakour2013b] M. Albakour, C. Macdonald, and I. Ounis. On sparsity and drift for effective real-time filtering in microblogs. In *Proceedings of CIKM,* 2013.

[Allan1996] J. Allan. Incremental relevance feedback for information filtering. In *Proceedings of SIGIR*, 1996.

[Allan1998] J. Allan, J.G. Carbonell, G. Doddington, J. Yamron, and Y. Yang. Topic detection and tracking. pilot study final report. In *Proceedings of DARPA,* 1998.

[Belkin1992] N.J. Belkin, and W.B. Croft. Information filtering and information retrieval: Two sides of the same coin? *Communications of the ACM*, 35(12):29–38, 1992.

[Holt2004]. Forecasting seasonal and trends by exponentially weighted moving averages. *International Journal of Forecasting, 20(1): 5-10,* 2004.

[Robertson2002] S. Robertson. Threshold setting and performance optimization in adaptive filtering. Information Retrieval 5(2):239–256, 2002.

[Robertson2009] S. Robertson and H. Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Information Retrieval*, 3(4), 2009.

[Xu1996] J. Xu and W.B. Croft Query expansion using local and global document analysis. In *Proceedings of SIGIR*, 1996.

[Yang2005] Y. Yang, S. Yoo, J. Zhang, B. Kisiel. Robustness of adaptive filtering methods in a cross-benchmark evaluation. In *Proceedings of SIGIR*, 2005.

[Zhang2004a] T. Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of ICML*, 2004.

[Zhang2004b] Y. Zhang. Using bayesian priors to combine classifiers for adaptive filtering. In *Proceedings of SIGIR*, 2004.

[Zubiaga2011] A. Zubiaga, D. Spina, V. Fresno, and R. Martinez. Classifying trending topics: a typology of conversation triggers on Twitter. In *Proceedings of CIKM,* 2011.