UNIVERSITY OF SOUTHAMPTON

A Context-Sensitive Relevance-Based Intelligent Data-Ranking Agent

by

Thomas J. Bell

A project report submitted for the award of
MEng Electronic Engineering

in the
School of Electronics and Computer Science

November 2013

"Write a funny quote here."

If the quote is taken from someone, their name goes here

UNIVERSITY OF SOUTHAMPTON

Abstract

School of Electronics and Computer Science

MEng Electronic Engineering

by Thomas J. Bell

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too...

Acknowledgements

The acknowledgements and the people to thank go here, don't forget to include your project advisor…

Contents

List of Figures

List of Tables

Abbreviations

LAH	**L**ist **A**bbreviations **H**ere

Physical Constants

Speed of Light	c	$=$	$2.997\ 924\ 58 \times 10^{8}\ \mathrm{ms}^{-\mathrm{S}}$ (exact)

Symbols

a	distance	m
P	power	W (Js^{-1})
ω	angular frequency	rads^{-1}

For/Dedicated to/To my…

Chapter 1

Introduction

Context-sensitive techniques, data fusion and ranking agents are frequently used in industry applications and consumer software products. Combined, these techniques are useful for ranking social media according to a users preferences and context. This report describes an agent for the classifying, scoring and ranking of data according to its context-sensitive relevance to a user.

1.1 The Problem

Social media, productivity tools and internet-based information are abundant on mobile devices, leading to users being overwhelmed with information, despite only a small amount of it being of any interest to a particular individual at any given moment. This calls for a means by which such data can be ranked or filtered according to its importance, interest or relevance.

1.2 Project Objective

The objective of this project is to produce a scalable and highly modular context-sensitive mobile-content relevance-based intelligent ranking agent, to order social media, productivity and other web-based information according to its time- and situation-specific relevance to a user.

1.3 Goals

The following are core goals which this project sets out to achieve.

1. Develop a scoring algorithm by which to judge to relevance of an item of data based upon a users
   (a) Personality profile
   (b) Historical data such as click-history
   (c) Environment (conditional upon time-constraints)
2. To perform automatic remote topic analysis to judge the topic of an item of data
3. Develop a sorting/ranking algorithm to sort or insert scored items of data efficiently, into an ordered list
4. Develop a stable and robust data fusion technique to combine a range of data into a user-context and data-context object.
5. Abstract away this agent into an extensible Java API for use in
   (a) Smartphone apps (Android)
   (b) Web-apps (Spring MVC)
   (c) Desktop applications (Java Swing etc.)
6. To develop a consumer smart phone app to demonstrate the working API which automatically ranks a users data according to its relevance

These are the criteria by which the extent of this project's success will be evaluated.

1.4 Unique Features

Many of the aspects of this project have never been seen combined into a single research project before and others (such as query-less context-sensitive scoring of data) have recieved little attention. This project is unique in its endeavour to combine data fusion techniques with context-sensitive scoring in the development of a commercially viable prototype.

Chapter 2

Background Research

A progressive project in the sphere of cross-platform relevance-based intelligent ranking agents, using text analysis and a mathematically rigorous scoring algorithms, requires research in a range of areas across the entire spectrum of low- to high-level computational theory and existing product research. The following summarises the research undertaken before and during the research and design phases.

2.1 Existing Data-Ranking Implementations

A good number of content aggregators exist at present in various forms, yet all distinctly lack the complementary relationship between social media (and others) and context-sensitive text-analytics for relevance-based ranking. The following is a summary to the key players in this space at present.

Google Now Google Now is a mobile app which combines Google's search feature with useful information which is deemed relevant to the user's environment such as weather, a map to get them home after a night out or nearby events.

ViralHeat ViralHeat is a web-based social media content aggregator and filter, used for commercial uses of social media. It allows the user to filter content from twitter, Facebook and others according to its sentiment (positive/negative). It's not available as a non-commercial social media aggregator and does not perform topic analysis for ranking.

StreamLife This app aggregates facebook content and tweets, but performs no topic/sentiment analysis or ranking, and provides no capability for including tasks, calendar appointments, SMS messages or emails.

2.2 Text analytics libraries

There are a good number of existing text analysics services available to the end user and the developer for a range of different types of analytics. These services include sentiment analysis, text categorisation, contextual targeting and a range of others. This section highlights some which are relevant to this study.

AlchemyAPI Alchemy provides an API which performs entity extraction, sentiment analysis, contept tagging, relation extraction and most notably, text categorisation (among others). It provides a free licence for up to 1,000 API calls per day. It provides all the core features which this project requires in terms of text analytics, however the text categorisation did not produce strong results for short amounts of text (such as tweets) and often failed to make any categorisation.

Semantria The best solution for sentiment analysis appears to be semantria. It gave consistently precise and accurate sentiment analysis for text containing more than 5 words.

Saplo Saplo was distinguished in its contextual analysis feature which allowed me to define a personalised textual context which could be matched against any type of text. This could have allowed me to define user-specific textual contexts against which to match data items, however it only allows users 2000 API calls per month on their free account.

Wingify This is a beta-stage contextual targeting API which can categorise text from a web page and extract key conepts. The online demo provided accurate results, yet there is not yet a public API available.

DatumBox DatumBox is a free machine learning API which performs sentiment analysis, subjectivity analysis, topic slassification, language detection, readability detection, educational detection, document similarity analysis, and gender detection. Many of these features may be useful for ranking text based upon its relevance to a particular individual. It has a simple API using http POST requests and a JSON response.

2.3 Data-mining

Outline: Facebook and Twitter API (phone, web and desktop), Android API, Android Calendar, Android Tasks, Android SMS, Android Sensors, Google Calendar and Tasks (web-based and desktop API).

2.3.1 Facebook

The Facebook statuses of a users friends can be fetched either using the Facebook API, or through wrapper libraries which simplify common Facebook API usage. The Facebook API is full-featured and well documented, yet for the purposes of demonstrating this ranking agent it's not necessary. Facebook4J is a java Facebook wrapper API which simplifies the most common Facebook API features into a more minimal library.

```
Facebook facebook = new FacebookFactory().getInstance();
facebook.setOAuthAppId(appId, appSecret);
facebook.setOAuthPermissions(commaSeparetedPermissions);
facebook.setOAuthAccessToken(new AccessToken(accessToken, null));
facebook.postStatusMessage("This is my status.");
//Gets a list of the users feed (friend's status updates)
ResponseList<Post> feed = facebook.getHome();
```

LISTING 2.1: Facebook4J example [1]

This library provides the capability to public messages and links, getting the users news feed, 'liking' a post, publishing a comment, searching for users, groups, events, places or locations and others. It supports pagination and reading options. Altogether it fulfills the needs of this project adquately.

2.3.2 Twitter

Simlar to Facebook, Twitter provides an API which is freely available yet overly complex for this project. Twitter4J is a free API which simplifies the Twitter API.

```
Twitter twitter = TwitterFactory.getSingleton();
//Gets and prints the users timeline
List<Status> statuses = twitter.getHomeTimeline();
for (Status status : statuses) {
    System.out.println(status.getUser().getName() + ":" +
                       status.getText());
}
```

LISTING 2.2: Twitter4J example [2]

Twitter4J provides sufficient documentation, support and features making it suitable for retrieving a users Twitter feed.

2.3.3 Android API

2.3.4 Google Calendar

Calendars from a users Google account can be retrieved on the Android platform using the Calendar Provider. This is a repository of a user's calendar events which can be queried.

```
    Cursor cursor = context.getContentResolver()
            .query(
                    Uri.parse("content://com.android.calendar/events"),
                    new String[] { "calendar_id", "title", "description",
                            "dtstart", "dtend", "eventLocation" }, null,
                    null, null);
```

LISTING 2.3: Calendar Provider example

This query returns a list of events which can be freely processed and sorted as required.

2.3.5 Google Tasks

In Android, Tasks can be retrieved from a Google accoutn using the Google Tasks API by prompting the user for their account credentials, retrieving an AuthenticationToken to create a GoogleCredential and using the Tasks Builder to create a Tasks Service. This is demonstrated in Listing 2.4.

```
        List<Task> tasks = service.tasks().list("@default").execute().getItems();
```

LISTING 2.4: Google Tasks example

2.3.6 Android SMS

SMS messages can be recieved in Android applications using a BroadcastReveiver which collects incoming SMS messages.

```
public class receiver extends BroadcastReceiver {
      public String str = "";
        @Override
        public void onReceive(Context context, Intent intent) {
            Bundle bundle = intent.getExtras();
            SmsMessage[] msgs = null;
            if (bundle != null) {
                Object[] pdus = (Object[]) bundle.get("pdus");
                msgs = new SmsMessage[pdus.length];
                for (int i = 0; i < msgs.length; i++)
                {
                    msgs[i] = SmsMessage.createFromPdu((byte[]) pdus[i]);
                }
            }
        }
    }
```

Listing 2.5: Android SMS example

This example shows a BroadcastReceiver which collects SMS messages when they're received and adds then to an array of SmsMessage objects for processing.

2.4 Semantic Analysis

This project requires topic analysis of items of mobile data, in order to compare them to the user's preference and ascribe relevance to them. Other semantic analysis capabilities would prove beneficial for increasing the range of criteria by which relevaance may be judged and to elliminate irrelevant data as early on as possible. These include readability, gender, subjectivity and language detection. The DatumBox API is chosen for semantic analysis for its coverage of these requirements, its ease of implementation and the fact that its use is free.

2.4.1 DatumBox API and usage

2.5 Context-Sensitive Scoring Algorithms

This is some sample text.

2.6 Ranking Algorithms

Insertion sort etc.

Chapter 3

Specification

Describe the overall specification - what it should do.

3.1 Data sources

This is some sample text.

3.2 Scoring Algorithm

This is some sample text.

3.3 Ranking Algorithm

This is some sample text.

3.4 Android Demonstration

This is some sample text.

Chapter 4

Algorithm Design

This is some sample text.

4.1 Data Acquisition

This is some sample text.

4.1.1 Data Item Acquisition

Converting each source item of data into a DataItem

4.1.2 User Data Acquisition

Getting info about the user from somewhere.

4.2 Classification

TODO: Discuss lexicon based vs. learning based techniques - lexicon requires dictionary, but learning requires training, garbage in - garbage out (good clean text),

4.3 Scoring Algorithm

Pre-processing, education/gender/readability etc, relevance equations etc.

4.4 Ranking Algorithm

i.e. Modified insertion sort

Chapter 5

Implementation

This is some sample text.

5.1 Data Acquisition

This is some sample text.

5.1.1 Data Item Acquisition

This is some sample text.

Chapter 6

Planning and Progress

This is some sample text.

6.1 A Section

This is some sample text.

6.1.1 A Subsection

This is some sample text.

6.2 Another Section

This is some sample text.

Chapter 7

Testing Strategy and Results

This is some sample text.

7.1 A Section

This is some sample text.

7.1.1 A Subsection

This is some sample text.

7.2 Another Section

This is some sample text.

Chapter 8

Critical Evaluation

This is some sample text.

8.1 A Section

This is some sample text.

8.1.1 A Subsection

This is some sample text.

8.2 Another Section

This is some sample text.

Chapter 9

Conclusion

This is some sample text.

9.1 A Section

This is some sample text.

9.1.1 A Subsection

This is some sample text.

9.2 Another Section

This is some sample text.

Chapter 10

Further Work

This is some sample text.

10.1 A Section

This is some sample text.

10.1.1 A Subsection

This is some sample text.

10.2 Another Section

This is some sample text.

Appendix A

An Appendix

This is an appendix.

Bibliography

[1] Facebook4j website. URL http://facebook4j.org/en/code-examples.html.

[2] Twitter4j website. URL http://twitter4j.org/en/code-examples.html.