ICS 2208 – Intelligent Interfaces 1
Assignment 2015/16

Personalised Adaptive RSS Reader and Recommender

Joel Azzopardi
joel.azzopardi@um.edu.mt

February 2016

1 Aims

The main aim of this assignment is to implement a Personalised RSS News Reader and Recommender System that is operational and actually provides personalised recommendations and automatically perform some content and link adaptation.

On the 'non-personalised' level, the system should act as an RSS reader (similar to *feedly*[1]. The system contains a default list of RSS URL sources categorised into some generic categories. This list (including the categories) can be modified by the user. The system downloads the RSS listings from the source and it downloads these RSS listings, and outputs the news listings to the user sorted according to publication date. The user should be allowed to view: all the news items; the news items originating from a particular category (e.g. "Local News"), or the news items coming from a single RSS feed (e.g. the "Times of Malta" feed.

[1] *http://www.feedly.com*

Then, on the 'personalised' level, the system should provide personalised recommendations from the news items originating in the RSS feed list. The system should have a way to identify each user. A user model should be constructed for each user, and this model should be updated continuously using implicit feedback based on the user's navigation, and/or explicit feedback provided by the user himself. This user model should then be used to provide personalised recommendations and content/links adaptations for the user.

The system should be made to work live online, and the user can interact with it via the web browser. You will be provided with Microsoft Azure passes so that you can upload the system live on Azure.

The deadline for this assignment is **Friday 27th May, 2016**.

2 Background

2.1 RSS feeds

Nowadays, every major news corporation provides at least one **RSS feed** for its news listings. Multiple RSS feeds are used to provide listings relevant to different categories (e.g. "International News", "Technology", etc).

An RSS feed is essentially XML-formatted plain text. Each feed contains some meta-information about the feed itself, and then a series of news items. Each news item is typically enclosed within the `item` tags, and includes the following fields:

- `title` – the title of the news item;
- `description` – a snippet from the news item (typically the first few sentences from the full news report);
- `link` – the link to the full report;
- `guid` – a unique ID for that news item; and
- `pubDate` – the publishing date for that item.

The main news RSS feed from the "Times of Malta" is: *http://www.timesofmalta.com/rss*

BBC news provides various news feeds, e.g.:

- World News – *http://feeds.bbci.co.uk/news/world/rss.xml*
- Technology News – *http://feeds.bbci.co.uk/news/technology/rss.xml*

More information about RSS feeds may be found on: *http://en.wikipedia.org/wiki/RSS*.

2.2 RSS Readers / Aggregators

RSS Readers or **Aggregators** are software systems that present the RSS feed data to the users. Users add the URL of RSS feeds to these systems either through manual input, or by clicking on the RSS icon. Some aggregators also suggest feeds to the users.

Google used to provide an *RSS Aggregator* service – *Google Reader*. However, this service has been discontinued. Nowadays, there are various aggregator services that are available, even freely. A popular service (that is also free) is called *Feedly* – *http://www.feedly.com*. Figure 1 shows a screenshot from this service.

You are encouraged to check out such services yourselves.

2.3 User Modelling and Adaptive Hypermedia

The lectures covered in class will provide a very thorough coverage of user modelling – what do user models consist of, how they can be updated etc – and Adaptive Hypermedia – how a system can adapt the content and links according to the user viewing that content. You should obviously familiarise yourself with the lecture material. All lecture slides will be online on the VLE. Discussions during the lecture about the content, and about how it relates to the assignment will be very welcome.

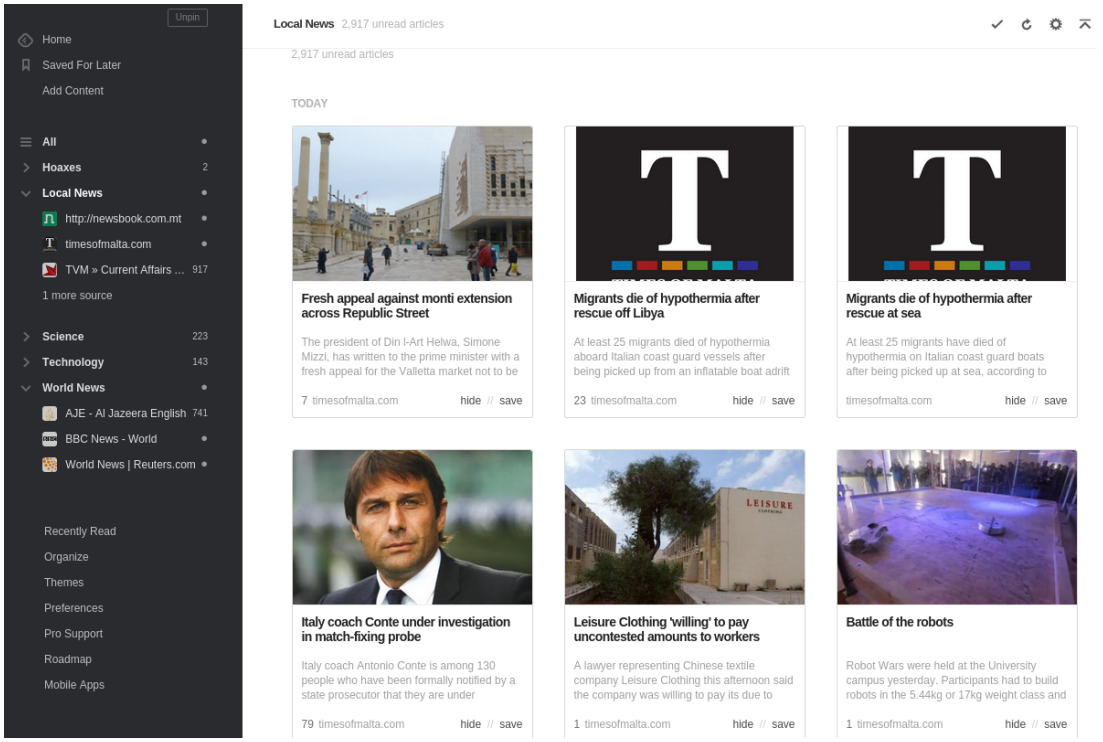

Figure 1: The Feedly interface

2.4 Other Useful Background Information

The material covered in the Web Intelligence study unit is also very relevant and essential to be able to implement this assignment. The most important topics are:

- Text Analysis – especially tokenisation, stop-word removal, stemming, and the construction of Inverted Indices;
- Information Retrieval Models – nmamely, the Vector Space Model; and
- Text Categorisation and Document Clustering.

3 Implementation Details

3.1 Server Setup

The system you develop should be implemented as a live system that is accessible from a web browser anywhere from the WWW. This task is not as trivial as it may sound, and you are advised not to leave it till the last minute. There will be marks assignment to the 'operationality' of a live system.

To put the system live, you will need access to a server space: set-up a home server (even using an old computer or a raspberry pi); obtain access to a cloud solution (Amazon Cloud, Google Compute, Microsoft Azure, ...) or purchase a Virtual Private Server. You will be provided with Microsoft Azure passes ($100 per month for 6 months for each student), and you are encouraged to use Azure. You will have a talk by Microsoft Malta representative on the Azure platform, and he will be showing you how you can use Azure.

The server OS can be anything. However, the use of a linux OS is advised to avoid licensing issues – remember that Linux is free, and the usual server software (PHP, MySQL, Apache, etc)can be installed without any problem. I will be dedicating a session on how to set up a Linux Web Server on Azure.

Another possible solution is to use a Virtual Private Server (VPS). A VPS is a virtual machine that is hosted in a server farm somewhere. There are various sellers of VPS. A VPS hosted in Europe will obviously provide faster access to Maltese connections that one hosted in USA or in another content. The company *UltraVPS*[2] offers VPS hosted in Germany or Netherlands with a fixed IP for as little as 2 euros per month for a minimum of 6 months. However, they will provide you a working OS, and you will be responsible to install the needed software.

Please note that I am not asking you to use a specific solution. I am just suggesting various possible solutions. There are many more options which can do the same job. However, given the Azure passes, and the trend towards cloud solutions, I think that using Azure is the most practical solutions.

[2] *http://www.ultravps.eu*

You are encouraged to try to setup a server yourselves. Next year, you will have assignments that will require you to provide a live system. Also, these skills will surely come useful for you later on in your careers. You can find lots of help online.

3.2 Domain Name

You are **not** required to purchase a domain name for your system. To provide a link to the live system, you can do the following:

- If using Azure, you can use the domain name provided by Azure;
- Provide the link using the IP address;
- Use a free domain/sub-domain; or
- Use a free dynamic DNS service such as *No-IP*[3].

3.3 Programming Language/s and other Tools

You can use any programming language that you want. You can even have seperate components coded using different languages that work in coordination with each other. Obviously, if you are going to produce a live system accessible from a web browser, you will need to use some web scripting language (HTML, Javascript, PHP, etc) to produce the front-end. You can have a seperate back-end that was developed in another language, e.g. Java.

You can use any other tool that you want as long as you reference it properly in your write-up. Obviously, you will need to implement the user modelling and recommendation part yourself.

You can also use any ready-made template, and/or ready-made design elements found online to have a neater interface. You can use the **Bootstrap** framework**??** and use ready-made front-end components. Another good resource can be *http://www.bootsnipp.com*.

[3] *http://www.no-ip.org*

3.4 RSS feeds

The default list of RSS feeds should contain the following feeds:

- **Local News**
  - `http://www.timesofmalta.com/rss/local`
  - `http://www.independent.com.mt/newsfeed`
  - `http://www.maltatoday.com.mt/rss/news/national/`
- **International News**
  - `http://feeds.reuters.com/reuters/worldNews`
  - `http://www.aljazeera.com/xml/rss/all.xml`
  - `http://feeds.bbci.co.uk/news/world/rss.xml`
- **Science and Technology News**
  - `http://feeds.newscientist.com/science-news`
  - `http://rss.acm.org/technews/TechNews.xml`

You can obviously add to this default list.

Users should be able to edit their feeds list. Remember that your system should be able to cater for multiple users, and each user can have a different list.

3.5 User Models

The structure of the user models is up to you. You should make the decisions based on the material covered in class. In your write-up you should provide a justification of the approaches that you took.

Remember that a user model should not be static. It should be updated based on the user's use. You can use any user model update approach – explicit, implicit, or

a hybrid approach. You will need to describe why you chose the approach you took in your write-up, and why you think it was the most practical approach.

You will also need some way to identify the users – as before, the choice is up to you, but will need to be justified in the write-up. Obviously, certain approaches that we cover in class, e.g. using a 3rd party application is not feasible. A possible approach can be to use logins linked with Facebook, or Google authentication.

One other problem that you will face is when you have a new user. Your system should allow the registration of new users. Each new user means that a user model needs to be initialised for him/her. You need to decide how to initialise the user model. In a possible scenario, a new user is asked for some information (or information can be obtained from his facebook profile), and the new user model is initialised based on this information. Initially, the user model can be classified with a stereotype.

3.6 Recommendations

The system should recommend *unread* news items to the users. Besides relevance, the system should take into consideration news freshness as well. A relevant news items that is old will probably not be of interest to the users.

There can be different types of recommendations, such as:

- Normal recommendations – the user is in the *Home Screen*, and recommended news items are shown to him/her.
- Recommendations based on a news story – a news story can be opened in a browser frame, and news items similar to that story can be recommended.
- Feed recommendations – new RSS feed suggestions can be given to the user based on what feeds other 'similar' users have and are finding 'interesting'.

You decide what type of recommendations to give to the user.

3.7 User Interface

The user interface should be browser based.

It should contain at least the following features:

- The viewing of news items in a 'non-personalised' manner sorted according to publishing date, allowing the user to filter the news items by category and feed as well.
- The display of recommended news items.
- User customisation controls – allowing the user to modify the list of feeds, and perhaps change the display mode, etc.

The user interface should also be made adaptive – e.g. if the user seems to prefer viewing news from a certain category, the system can open by default that caegory, or read articles can be 'hidden'. We will be discussing such systems better during the lectures.

4 Write-Up

Your system should contain a write up that includes the following sections:

- **System Specifications/Description** – describing your system's features, and how to utilise the different features. This section should be similar to a short user manual that can be used to 'market' your system.
- **System Design** – describing the main approaches used by your system on the basis of the lecture notes. This section should justify the design choices.
- **System Implementation** – describing how the system works.
- **Conclusions and Future Work** – an appraisal of your system's performance, and suggestions for Future Work.

- **Division of Work** – describing what parts were implemented by yourself, what resources you utilised, and how the work was divided amongst the different members of the team.

5 Team Setup

You should work in teams of 2 or 3 people. You should **not** work by yourselves, since this assignment involves too much work for a single person. Teams of 3 people are advised. It stands to reason that a team of 3 persons should produce a system that involves more work that a team of 2 persons would.

6 Mark Distribution

The marks will be distributed as follows:

- **System Quality and Performance** – 35%
- **System Live Operation** – 15%
- **Write-up**
  - **General Quality** – 5%
  - **System Specifications/Description** – 10%
  - **System Design** – 10%
  - **System Implementation** – 10%
  - **Conclusions and Future Work** – 10%
  - **Division of Work** – 5%

7 Deadline

The deadline for this assignment is **Friday 27th May, 2016** as 12:00.