José Pedro Silva

Semantic Good Morning
News collection, management and presentation
José Pedro Silva

Semantic Good Morning
News collection, management and presentation

Dissertação de Mestrado
Mestrado em Engenharia Informática

Trabalho realizado sob orientação de

Professor Pedro Rangel Henriques
José Pedro Novais
Acknowledgements

First and foremost I want to thank my family, who have supported me through my entire life. Without them, not only this dissertation, but my whole academic path would not have been possible. Second, I thank my supervisor, Professor Pedro Rangel Henriques, for guiding me in all the tasks I had to accomplish, for always providing helpful insight, and for being available to discuss any problem. I also want to thank my co-supervisor, José Pedro Novais, for helping me to understand the company vision for Semantic Good Morning, and for helping me to integrate myself in Ubiwhere. My thanks also go to Engineer Nuno Ribeiro and Engineer Rui A. Costa for giving me the possibility of working on this project, and too all the Ubiwhere trainees and employees who made me feel at home. Last, I want to thank everyone who was important in my academic path, and that today, I can call real friends. I will begin with my girlfriend, Catarina Magalhães; the room mates that I had during the the six years I lived in Braga: Nuno Silva, André Félix, Nuno Faria, Emanuel Gonçalves, Samuel Moreira, Pedro Miranda, André Parra and Joana Duarte; and other colleagues, like Diego Albuquerque, Daniel Quintas, Cláudio Novais and Daniel Oliveira. With them I shared moments i will never forget.
Resumo

Hoje em dia as televisões têm cada vez mais poder de processamento. Isto permite que se desenvolvam aplicações que correm na televisão. Com a evolução das diferentes frameworks das Smart TVs, o desenvolvimento de aplicações para televisão vai começar a ser mais comum.

O objectivo do projecto aqui relatado é estudar a viabilidade das tecnologias disponíveis de desenvolvimento para Smart TV, pesquisar os meios para obter e classificar artigos noticiosos, estudar técnicas para detetar documentos semelhantes e finalmente implementar um sistema dividido em duas partes: o Back-end, onde vão ser obtidos e geridos os artigos noticiosos; e o Front-end, que será uma aplicação para Smart TV, irá apresentar notícias filtradas de acordo com o rating, que é baseado em critérios padronizados ou pessoais.

Como resultado deste projecto, foi implementado um Sistema para obtenção, gestão e apresentação de notícias.
O Back-end reúne, classifica e deteta notícias semelhantes, obtendo ainda imagens relacionadas com as mesmas. Depois, e de acordo com as preferências de utilizador, as notícias são avaliadas e enviadas para o Front-end.
O Front-end é uma aplicação Samsung Smart TV. A Samsung Smart TV foi escolhida como a framework de Smart TV mais apropriada para o projecto. Enquanto as notícias estão a ser apresentadas no Front-end, é enviado para o Back-end feedback acerca de cada artigo noticioso, o que vai causar alterações na ordem em que as notícias são apresentadas.
Abstract

Televisions nowadays are shipped with more and more processing power. This allows the development of applications that run in the television. With the evolution of the different Smart TV development frameworks, application development for televisions will become more usual.

The aim of the project here reported is to study the viability of the available technologies for Smart TV development, research the means to gather and classify news articles, study techniques of similar document detection and finally to implement a system divided in two parts: the Back-end, where the news aggregation and management will occur; and the Front-end, a Smart TV application that will present to the user the news filtered according to the rating based on standard or customized criteria.

As a result of this project, a System for news collection, management, and presentation was implemented. The Back-end collects, classifies, and detects similar news articles, and also obtains news related images. Then, and according to user preferences news are rated and served to the Front-end. The Front-end is a Samsung Smart TV application. Samsung Smart TV was chosen as the best suited Smart TV framework for the project. While news are presented on the Front-end, feedback about each news article is being sent to the Back-end, which will cause changes in the news presentation order.
# Contents

List of Figures .......................... ix
List of Tables .......................... xi
List of Listings .......................... xiii

1 Introduction .......................... 15
1.1 Goals ................................ 16
1.2 Thesis ................................ 16
1.3 Contribution ......................... 16
1.4 Organization of the Dissertation .... 17

2 Smart TV .............................. 19
2.1 Smart TV Frameworks ................. 20
   2.1.1 Samsung Smart TV ............... 20
   2.1.2 Google TV ....................... 21
   2.1.3 LG Smart TV .................... 21
   2.1.4 Yahoo! Connected TV .......... 22
2.2 Summary ............................ 22

3 News: Gathering and Management ..... 23
3.1 News Articles ......................... 23
3.2 News Aggregating Services and Applications .......................... 23
   3.2.1 Icurrent ........................ 25
   3.2.2 TrapIt ........................... 25
   3.2.3 Pulse ........................... 25
   3.2.4 News360 ......................... 25
   3.2.5 News Republic .................. 25
   3.2.6 Zite ............................ 26
   3.2.7 Flipboard ....................... 26
   3.2.8 Google News .................... 26
3.3 Gathering the News ................. 26
3.4 Similar News Detection and Management .................. 27
3.5 News Classification .................. 28
   3.5.1 Tools For Document Classification and/or Document Clustering .... 29
3.6 Sorting News ........................ 31
3.7 Summary ............................ 31
CONTENTS

4 SGM Requirements and Architecture 33
  4.1 System Requirements ........................................... 33
    4.1.1 User Stories .............................................. 33
    4.1.2 Use cases .................................................. 35
  4.2 System Architecture ........................................... 39
  4.3 Summary .......................................................... 41

5 Development 43
  5.1 News Collection: feed, news content and on site media extraction .......... 43
    5.1.1 Feed collector .............................................. 43
    5.1.2 Full article extraction ..................................... 45
    5.1.3 Media on page extraction .................................. 46
  5.2 News audio and related media .................................. 47
    5.2.1 Audio ....................................................... 47
    5.2.2 Images ...................................................... 48
  5.3 News classification ............................................. 49
  5.4 Similar news detection ......................................... 50
    5.4.1 Integration in the System ................................ 51
  5.5 Web Service ..................................................... 51
  5.6 News Sorting ..................................................... 52
  5.7 Client side ....................................................... 53
    5.7.1 Authentication and main screen ................................ 54
    5.7.2 Category and Source preferences .............................. 56
    5.7.3 Similar News List and Similar News Presentation .................. 57
    5.7.4 Automatic and user requested feedback .......................... 59
  5.8 Results ........................................................... 59
  5.9 Summary .......................................................... 63

6 Conclusion 65
  6.1 Future Work ...................................................... 67

Bibliography 69

A Sample file for RSS 2.0 71

Acronyms 73
List of Figures

4.1 Block diagram of the system architecture ................................................. 40

5.1 Excerpt of the news database table columns ............................................. 44
5.2 Screen shot of the main page of the client side application .......................... 55
5.3 Screen shot of the edit category preferences menu ...................................... 56
5.4 Screen shot of the edit source preferences menu ....................................... 57
5.5 Screen shot of the similar news list menu .................................................. 58
5.6 Screen shot of the similar news presentation ............................................. 58
5.7 Performance in seconds of the most important System’s actions .................. 62
List of Tables

3.1 Number of visits per month and type of some popular Portuguese news websites 24
3.2 News aggregation services/applications overview 24

4.1 Advance news use case description 36
4.2 Choose categories use case description 36
4.3 Give positive feedback use case 37
4.4 Give negative feedback use case 37
4.5 Reset user model use case 37
4.6 Access other sources use case 38
4.7 Choose sources use case 39

5.1 API end points of the web service 52
# List of Listings

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Code necessary to extract the full article from a Web page, using Boilerpipe.</td>
<td>45</td>
</tr>
<tr>
<td>5.2</td>
<td>Excerpt containing original Boilerpipe source code.</td>
<td>45</td>
</tr>
<tr>
<td>5.3</td>
<td>Changes made to the excerpt of Boilerpipe source code displayed in Listing 5.2.</td>
<td>46</td>
</tr>
<tr>
<td>5.4</td>
<td>Excerpt of the response given by the AlchemyAPI service for a named entities request.</td>
<td>48</td>
</tr>
<tr>
<td>5.5</td>
<td>Excerpt of the response given by the AlchemyAPI service to a text categorization request.</td>
<td>50</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

We live in the Information Age, where in almost every nation in the world population has instant access to information and the ability to freely spread it. An important part of that information is composed of news articles, which keep us on track of what is happening in our cities, countries and even in the rest of the world. Newspapers and television newscasts have been around for quite long, but are slowly being set aside by the Internet. Trying not to get outdated, all major news channels and newspapers now have their own Website. In their Websites they make available for free, some of the information they have. Also, Really Simple Syndication (RSS) feeds have become increasingly popular, and are available in most of the news websites, to make life easier for the end user. With RSS feeds users can subscribe websites without releasing any personal information, and they also save time because accessing the website who provided the feed is only necessary if the summary interests them.

It has become natural that on-line services no longer flood the user with random information but instead, adapt it to the user preferences. One of these websites is ebay\(^1\) which frequently sends emails with items that are recommended for each user, based on previous buys. Other example is Amazon\(^2\) which just upon registering, asks the user what interests him, so they can make it easier for the user to find his next purchase. A consequence of such behaviour is that the content presented to one user is not always the same that is presented to any other. The Web gives in this way a feeling of personalization that pleases the users, by providing user driven content.

Ubiquitous (or pervasive) computing is becoming the replacement of the desktop paradigm. Aims at integrating technology in the life of the user in a natural way. Instead of forcing the change in the user environment, it blends itself in everyday devices. Examples of such devices include PDA’s, mobile phones, tablets, televisions or even fridges, which are often connected to each other and to the internet, making information available wherever it is needed. Development of television applications is a part of the context of ubiquitous (or pervasive) computing. This area is giving its first steps. Nevertheless it is easy to see the potential of this market, given that the television is already part of almost every families home. This evolution allows us to reach the user, on the same equipment he turns on every day.

Smart TVs are regular televisions that can access the web and have the ability to run software applications. The capabilities and features of Smart TVs are still unknown to many end-users. The same is valid for software developers, who are not familiar with the technologies needed to develop for the different Smart TVs.

\(^1\)http://www.ebay.co.uk/  
\(^2\)www.amazon.com/
CHAPTER 1. INTRODUCTION

How do users interact with the Smart TV and how can the information be displayed on (usually) big screens, are some of the new challenges presented.

Ubiwhere\(^3\) is an high-tech information and communications company focused on R&D activities. The need that user has of being up to date, the personalization that is becoming natural all around the Internet, the information overload problem, and the emerging market of Smart TVs were the motivation that led Ubiwhere to create the thesis proposal that resulted in this dissertation. Due to the dimension of the project, Ubiwhere managers decided to assign two M.Sc. students to the task, each focusing in different parts of the research and implementation. The other M.Sc. student - Ricardo Costa - studies Electrical and Computer Engineering at Faculdade de Engenharia da Universidade do Porto (FEUP), and the tasks that were assigned to him are exposed below, in Section 1.1.

1.1 Goals

The goal of my master project is to present news to the user, provided by some of the most reliable sources available on-line, through the television. To select the sequence of news to be presented to each user, the content available from the provider will first be filtered by category. At a second stage, the user feedback about each news will be used to rate the upcoming ones.

It is highly probable to get news about the same event from different sources. So it is imperative that these repetitions are detected and filtered. It will be on this subject that the other M.Sc. student, Ricardo Costa, will focus most of his efforts. More precisely on the module that compares two news stories for a level of similarity, and the module that chooses the most relevant news on a group of similar news. He will also develop part of the client side application, concerning the menus with the user preferences.

News will be presented to the user on a Smart TV, and will be read using a voice synthesizer, so that the user will not need to look at the television to be informed.

Taking in consideration that frameworks for TV application development are quite young, it is important to assess their quality and study their adequacy for the project. When the study is finished, the most suited framework will be chosen for the project.

1.2 Thesis

The news classification and similar detection can be used to significantly enhance news presentation, showing user driven content. Smart TV devices are capable of requesting news from a Back-end, presenting them, and sending feedback.

1.3 Contribution

The work exposed in this dissertation deals with topics in the following areas:

\(^3\)http://www.ubiwhere.com/
1.4 Organization of the Dissertation

This dissertation is composed of 6 chapters starting with the current one, the Introduction. In Chapters 2 and 3 the background necessary to develop this master thesis project is presented. Chapter 2 contains information regarding Smart TV, the existent Smart TV Frameworks and services/applications that have similarities with this project. Chapter 3 explains which and how news will be collected as well as how they will be managed.

In Chapter 4, the requirements are exposed and the designed architecture is presented. In Chapter 5 details about the project’s development will be presented, along with the obtained results.

At the end, in Chapter 6 conclusions will be drawn.
Chapter 2
Smart TV

This Chapter presents the State of the Art, concerning the central topic of the project, Smart TV. It starts with a brief introduction to the Smart TV concept and to the short history of Smart TV’s. After that, the study of the maturity of the most important frameworks for Smart TV applications development, their popularity and the technologies used to develop for them will be presented.

History

Before exploring the different frameworks available for application development on Smart TVs, it is important to understand what is Smart TV. Just like smartphones are regular phones with extended capabilities, such as access to the web and the ability to run software applications, the same happens with Smart TVs. Smart TV, also known as Connected TV, is a device that can be a traditional television set with capability to connect to the internet and execute applications, or a set-top box that extends a normal television.

The Smart TV concept dates back to 1994 when a first patent\(^1\) was filed, and updated\(^2\) one year later. The goal was to create a device that would introduce advanced technology to the life of each individual, without the complication of the regular computers. This can be perceived by reading the following excerpt that belongs to the Prior Art section of the latter patent:

“... The present evolution of technology is towards rendering ever more common-place data technology and an ever more advanced introduction thereof into the life of each individual. In the short or medium term the use of the computer will become a necessity in the organization of the everyday life of the population. However, the present inventors consider that computer science is too complicated a technology to be used by most individuals.

Therefore the object of the invention is not to make the use of a computer as current as that of a television receiver, but instead to increase the field of use of television by linking it with data processing systems by means of digital or analog networks.”

Today, the main purpose of a Smart TV can be described as: to provide means to deliver on-demand multimedia content directly on the television, whether it comes from the Internet or from other computers or network attached storage devices on the network\(^3\) as well as to run native or

\(^3\)http://en.wikipedia.org/wiki/Smart_TV
Web based applications. Some of these applications are shipped with the Smart TV, granting access to a set of Internet based services such as Netflix\(^4\). In most of the cases, more applications are available for download.

Internet Protocol television (IPTV) is a technology used to deliver television services, which uses the Internet Protocol (IP) over the Internet. Smart TV shares some of its features with IPTV services, which are already very common in most of the developed countries. Usually the services offered by IPTV include live television, video on demand, and the possibility of recording TV shows. Some also have small applications, like simple games available for users to play. The main differences between IPTV and Smart TV are that on the first, the customer has a monthly fee while the second is free (in both cases, we are ignoring the costs of the devices that provide the service); and with the first we only have access to the content the ISP that is providing the service made available, while on the second, we have access to a Global market of applications and multimedia content. In spite of the access to Smart TV service being free, some of the applications or services that it offers are paid.

To have access to IPTV services, a set-top box is always required. This box can sometimes be rented or purchased. On the other side, to access Smart TV services, we can buy a Television with Smart TV capabilities, or a set-top box that extends a normal television. While it is still early to see, Smart TV technology will probably evolve faster than the rate which a normal user buys a new television. So buying a set-top box may be seen as a better investment, since that in most cases it is cheaper to buy a new set-top box rather than a new television.

### 2.1 Smart TV Frameworks

Nowadays there are a wide range of Smart TVs and boxes available in the market. However, some of the associated development frameworks do not have a Software Development Kit (SDK) available for download. This fact makes devices such as Western Digital Live Hub\(^5\) or Apple TV\(^6\) unsuitable for the purpose of this project. To attain our purpose, it is important that the chosen device provides a powerful framework capable of fully fill our needs, and that we can reach an high amount of end users with our application. An analysis of the most promising SDKs and Frameworks is made in the following sections.

#### 2.1.1 Samsung Smart TV

Samsung Smart TV was probably the first to get in the market of Smart TV devices, in 2007. Back then they started by calling it Internet TV, and it allowed the user to receive information from the Internet while watching TV. They did not stop there and developed Smart LED TV which added support to downloaded apps. Later it was renamed to its current name Samsung Smart TV\(^7\). Despite the success of their Smart TV\(^8\), it is believed that some of the Samsung TVs will be shipped with Google TV in the future\(^9\).

---

\(^4\)www.netflix.com
\(^6\)http://www.apple.com/pt/appletv/
\(^7\)http://en.wikipedia.org/wiki/Samsung_Electronics
\(^8\)http://twentyfoursevennews.com/gcc/media/samsung-enjoys-45-smart-tv-market-share/
\(^9\)http://googletv.blogspot.pt/2012/01/from-las-vegas-strip-to-your-living.html
2.1. SMART TV FRAMEWORKS

The released **SDK** only works on Windows. It allows the creation of Flash applications and also Web applications, that obviously use **HyperText Markup Language (HTML)**, **Cascading Style Sheets (CSS)** and Javascript. The simple tests made with the **SDK** (displaying text on the television emulator) were made to understand the ease of use of the **SDK** and its maturity. In the short amount of time used to test it, the **SDK** editor and emulator performed properly, being that no bugs were found and no crashes happened.

The framework is well documented and there are some tutorials available on the Samsung’s Website for developers\(^{10}\). Outside of Samsung forums there are not many places where we can find programming discussions about this framework. At this moment, there are more than 1000 applications registered in the Samsung Smart TV app store, and more than 2 million Smart TV units were sold. Registering a new application in the Samsung Smart TV app store is free.

2.1.2 Google TV

Google TV was originally launched in May 2010, it was co-developed by Google, Intel, Sony and Logitech\(^ {11}\). Google TV uses the Linux version of Chrome, and the Android OS to provide the user the ability to surf the web and install Android applications\(^ {12}\).

Google TV is integrated in some Sony televisions, but is also available through a set-top box.

The Google TV **SDK** is pretty much the same used to develop Android smartphone applications. It is integrated with eclipse and there is an extra add-on exclusively for tv development. The emulation process is different when compared to smartphone development, it uses **Kernel-based Virtual Machine (KVM)**. Given that **KVM** can only run on a Linux physical installation, this **SDK** was not fully tried.

Having the Android OS on the Google TV means that the main programming language used to create Google TV applications is Java. Also, the community support for developers and the Framework documentation are extensive.

With no official numbers found, the estimate of applications on the Android Market for Google TV is between 59 and 271. The results were obtained based on a search on the Android Market. There is an estimate of 50k active Google TV users, and 1 million units sold. Submitting a new Google TV application to the Android Market is also free.

2.1.3 LG Smart TV

It was firstly introduced in 2007 and named NetCast Entertainment Access\(^ {13}\). As they evolved the platform and introduced new features, the name was changed to LG Smart TV\(^ {14}\). One of the most remarkable characteristics of LG Smart TV is that the interaction with the TV set can be done with a Wiimote like controller.

---
\(^{10}\)http://www.samsungdforum.com/
\(^{11}\)http://en.wikipedia.org/wiki/Google_TV
\(^{12}\)http://www.google.com/tv/features.html
\(^{13}\)http://www.lg.com/us/netcast/index.jsp
\(^{14}\)http://www.lg.com/global/smarttv/index.jsp
CHAPTER 2. SMART TV

As with Samsung Smart TV's SDK, LG Smart TV SDK allows the development of Flash or Web Applications and it is only available for Windows. Most of the information for developers of LG Smart TV can be found at the LG Apps TV Developer Lounge\(^ {15}\). At this page we can find several guides that help with the creation of LG Smart TV applications, as well as a forum that can also be very helpful. Registering a new application on the LG Apps TV Seller Lounge\(^ {16}\) is free, as long as the application itself is free. In order to sell paid applications, a registration fee as to be paid. It was confirmed in CES 2012\(^ {17}\) that some LG TVs will be shipped with Google TV\(^ {18}\). Despite this fact, LG Smart TV won't be abandoned by LG in all the future TV models.

### 2.1.4 Yahoo! Connected TV

Yahoo! Connected TV\(^ {19}\) from Yahoo was announced in 2008. A Widget Development Kit is available at the developers website\(^ {20}\). Our goal is to create an application, not a widget, so this option is set aside from the beginning. Despite that, developers from Portugal cannot submit applications to the Yahoo Connected TV Store, and Yahoo Connected TV is only available in a very small set of TV sets. Also the documentation and general information for developers is short and hard to find.

### 2.2 Summary

In this chapter it was introduced the concept of Smart TV as well as the not so long history of its existence. Since the concept was introduced in 1994, it has evolved due to the technological developments made in the area. Today, Smart TV is no longer only a concept, it can be found in appliance stores and in living rooms all around the world. The number of SDK’s made available for developers is already significantly big, and they are improving at very fast rate. The same happens with Smart TV devices, that are becoming more easily accessible for the common customer and are capable of executing more demanding tasks.

Concerning the battle among Smart TV makers, Samsung is ahead, with more Smart TV units sold when compared to its competitors. Google TV is also among the devices for which developers can create and publish applications, and it has a good foundation (Android OS). This Operating System is currently the most used in smartphones\(^ {21}\) and therefore has a huge community that is used to develop for it. Nevertheless it is having some hard times, selling less units than expected. The products of the remaining brands seem quite behind, regarding their actual acceptance in the market, and do not look like they have as promising future as the two mentioned above.

\(^{15}\)http://developer.lgappstv.com/

\(^{16}\)http://seller.lgappstv.com/

\(^{17}\)Consumer Electronics Show - http://www.cesweb.org/

\(^{18}\)http://googletv.blogspot.pt/2012/01/from-las-vegas-strip-to-your-living.html

\(^{19}\)http://connectedtv.yahoo.com/

\(^{20}\)http://connectedtv.yahoo.com/developer/

\(^{21}\)http://www.gartner.com/it/page.jsp?id=1848514
Chapter 3

News: Gathering and Management

This Chapter describes the State of the Art, concerning now the news handling that is the focus of the project. It starts with a brief description of what will be considered a news article and how the news providers quality will be assessed. After this, an overview of some of the most interesting services that fit in the news aggregators category will be presented. It also includes an explanation of how the news will be gathered as well as the State of the Art for news classification and duplicate detection.

3.1 News Articles

A news article is a discussion or an exposure of recent events. It can contain images or videos related to the event. News articles can be found in newspapers, general news websites, sports news websites, magazines, technology websites, blogs, etc. In this project, the news articles considered will be the ones from general and sports news websites, as they are both easily obtained and usually the most looked for. From now on, they will be addressed as news articles or solely as news.

Everyday, and all around the internet, news articles are published. It is important to find the best sources to obtain the news. There are two main characteristics that the chosen news providers should have. The first is reliability of the source. The second is the acceptance of the source. If many people already visit a specific website, it is a strong indication that the users enjoy their news, and this is a measure of acceptance. Although the number of visitors/readers is not a rigorous measure of reliability, it is also a strong sign that those users find the Website reliable. In Table 3.1, it is presented an overview of the number of visits for some of the most popular portuguese news sites. The displayed data was obtained from Google Adplanner\(^1\), and is a starting point for the choosing of the default portuguese news sources, in our application.

3.2 News Aggregating Services and Applications

Searching on-line for “personalized news aggregator” it is easy to find services which have similar features to what is intended to create under this master thesis project. These aggregators gather news from different sources and then present them to the user. Some of them also include some kind of personalization. A thorough search was made to identify the more interesting ones.

\(^1\)https://www.google.com/adplanner
### Table 3.1: Number of visits per month and type of some popular Portuguese news websites

<table>
<thead>
<tr>
<th>Domain</th>
<th>Type</th>
<th>Number of Unique visitors per month</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmjornal.xl.pt</td>
<td>general</td>
<td>1.3M</td>
</tr>
<tr>
<td>jn.pt</td>
<td>general</td>
<td>1M</td>
</tr>
<tr>
<td>publico.pt</td>
<td>general</td>
<td>1M</td>
</tr>
<tr>
<td>noticias.rtp.pt</td>
<td>general</td>
<td>1M</td>
</tr>
<tr>
<td>dn.pt</td>
<td>general</td>
<td>910k</td>
</tr>
<tr>
<td>abola.pt</td>
<td>sports</td>
<td>2M</td>
</tr>
<tr>
<td>record.pt</td>
<td>sports</td>
<td>1.5M</td>
</tr>
<tr>
<td>ojogo.pt</td>
<td>sports</td>
<td>690k</td>
</tr>
</tbody>
</table>

### Table 3.2: News aggregation services/applications overview

<table>
<thead>
<tr>
<th>Name</th>
<th>Available on Web</th>
<th>Available as a Smartphone or Tablet App</th>
<th>Available for Smart TV</th>
<th>Auto adapts to user interests</th>
<th>Free</th>
</tr>
</thead>
<tbody>
<tr>
<td>icurrent</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Trap!</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Pulse</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>News360</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>News Republic</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Zite</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Flipboard</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Google News</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>

We start this section with an overview of the most important characteristics of each service/application and continue with the detailed analysis of each one.

### Overview

In Table 3.2 an overview of the analysed services/applications is presented, showing a comparison of some important aspects. From the presented data, we can withdraw three main conclusions. The first one is that these services/applications are usually available for Smart-phone or tablet and on the Web, but rarely for Smart TV. The second is that the acquisition and use of these applications is always free. Last, we can see that there is a trend to auto adapt to user interests. Even the ones that do not auto adapt, take into consideration user manually entered preferences, as we will see next.
3.2. NEWS AGGREGATING SERVICES AND APPLICATIONS

3.2.1 Icurrent

Icurrent\(^2\) is a personalized news aggregator service available online, for free. It has a set of pre-defined sections (News, Technology, Business, etc.) and inside this sections we can add channels that are the most related with our interests. On the home page the user finds some of the news of his top channels. In a channel’s page a list of the most recent news that the channel contains is presented. At the top, key topics and key sources of the channel are presented. The user is allowed to remove or add topics or sources. For each news, the user can also choose to say that he is or is not interested. When one of this options is selected, the topics and sources of that news are shown, and can be selected, so in the future more/less news with these topics will be presented. Only a portion of the news is available to be read on the Icurrent Web page. To read the complete news the user is redirected to the source’s Web page. When this happens, at the top of the page remains a frame from Icurrent.

3.2.2 TrapIt

TrapIt\(^3\), is a service very similar to Icurrent. Generally speaking, all the options present in Icurrent are also in TrapIt. What stands out in TrapIt compared to Icurrent is the design of the page. It is more organized and more appealing. In Icurrent it is easy to feel overwhelmed with information, wherein TrapIt that does not happen.

3.2.3 Pulse

Pulse\(^4\), is a personalized news aggregator application for smart phones and tablets. The application is divided in several pages. In each page the user can have a maximum number of blogs/sources/categories, which are totally configurable. For each one of those entries, the user can scroll horizontally, and see all the news from that entry. As in the services mentioned before, only a description of the news is available in the application. To read the full news, the applications opens the Web page of the article.

3.2.4 News360

News360\(^5\) is another solution, which is very similar to the ones presented before. It is available on the Web, as well as in smart phones and tablets. There is a set of pre-defined categories, but the user can create new ones. Inside each new category, the user can add one or more topics. To read the full articles, the user is redirected to the original page.

3.2.5 News Republic

News Republic\(^6\), another news aggregator application, is available on smart phones and tablets. On the main menu we can find the top stories, and on second menu (my news), the user can find

\(^2\)http://www.icurrent.com/
\(^3\)http://trap.it
\(^4\)http://www.pulse.me/
\(^5\)http://news360.com/
\(^6\)http://www.news-republic.com/
the topics he chose to track. This topics are customizable. Opening a topic shows the most recent news about it, and also the possibility of searching about it on Google, Wikipedia, Youtube or Twitter. When a news is selected, the majority of the article is displayed.

3.2.6 Zite

Zite\(^7\) is an iOS application. When it comes to the style it is quite different from the application-s/services mentioned before. When presenting the news, it gives to the user the sensation of browsing a magazine. It can connect to your Facebook, Twitter, Evernote, Google Reader and Google Plus accounts, in order to understand your interests, and set some initial categories. Alternatively the user can manually choose categories that interest him. While reading a news article, the user can also say that he does or does not like it. This will help the system to adapt the future news to his interests.

3.2.7 Flipboard

When it comes to presentation style, Flipboard\(^8\) is quite similar to Zite. The magazine style is there, and probably even in a more achieved way. It does not connect to the social networks to infer the user’s preferences, but it uses them to show Twitter or Facebook posts to the user. Social networks are not the only categories available on Flipboard; usual categories as sports and business are also present, and can be added to the user’s preferred categories.

3.2.8 Google News

Google News\(^9\) service seems quite simple but has many interesting features. It has a set of predefined categories and news sources, that we can extend, by adding new topics or selecting other sources from the ones that are available, respectively. The amount of news that will be presented from each category and source can be adjusted. Also, for each news, a set of other sources that may have reported the same event, is available.

3.3 Gathering the News

In spite of many news articles being available for free on the news Websites, collecting them automatically would require a Web Crawler\(^{10}\) as in \([2]\) and would probably end up being a violation of the websites’ terms of service. Fortunately, most of these Websites provide a RSS feed service. RSS is a Web content syndication system concerned with propagation of Extensible Markup Language (XML) documents containing short descriptions of Web news \([3]\). So, in this project the RSS service will be adopted to collect the news. In RSS 2.0 specification\(^{11}\), we can find the XML elements that compose a RSS file.

\(^7\)http://zite.com/
\(^8\)http://flipboard.com/
\(^9\)http://news.google.pt/
\(^10\)http://en.wikipedia.org/wiki/Web_crawler
\(^11\)http://cyber.law.harvard.edu/rss/rss.html
In Appendix A, one RSS sample file is presented. Each RSS file can only have one channel. The channel contains the description of the RSS feed. It is composed of several XML elements, some required, some optional. Below are shown the required elements, and the most important optionals:

- **title (required)** - name of the channel;
- **link (required)** - Uniform Resource Locator (URL) for the Website corresponding to the channel;
- **description (required)** - description of the channel;
- **language (optional)** - language the channel is written in;
- **pubDate (optional)** - publication date for the content in the channel;
- **category (optional)** - one or more categories that the channel belongs to.

The channel may also include items. An item may represent a story, or in our case, a news article. The item does not have any required fields, but at least title or description must be present. The most relevant item elements are displayed in the next list:

- **title** - title of the item;
- **link** - URL of the item;
- **description** - the item synopsis;
- **category** - one or more categories in which the item is included;
- **pubDate** - date when the item was published;
- **guid** - a string that uniquely identifies the item.

These are the most important XML elements, the ones that will be stored in a database, and that will be used either to help in the news management or to be presented to the user.

### 3.4 Similar News Detection and Management

When gathering news articles from different sources, obtaining more than one news article about the same event, will happen many times. The user, usually does not want to read the same news twice, unless he feels the need to have access to other source’s point of view. So, to prevent this from happening, each gathered news has to be compared with the existing ones, and then treated accordingly. Instead of eliminating the repeated ones, we can agglomerate news articles about the same event. With this method, we can let the user see preferably news from his preferred sources, or even allow the user to read the same article from different sources. In [2] they go a little further, and after grouping news into stories about the same event, they generate a summary of each event.

There are a lot of proposed solutions for detection of duplicate or near-duplicate documents. Most of these algorithms tend to calculate the distance between the two strings/documents, and
are influenced by the words sequence. In our case, we will be comparing two completely different
texts, that are about the same event. So, the sequence of words is not relevant.
The most appropriate algorithm found to do this was Cosine similarity which is explained on the
Review and Related Work section of [4] and in the Cosine Measure section of [5]. This algorithm
measures the similarity between two vectors, by measuring the cosine of the angle between them.
In one of the implementations of this algorithm, each vector has all the words present in the doc-
ument, and the number of times it repeats, i.e. a bag of words. There are a lot of changes that can
be done to this algorithm, such as removing all the stop words (e.g. the) and stemming (reducing
all the words with the same stem to a common form [6] e.g. wait is the stem of waited). Depending
on the case, this changes can result in optimizations.

3.5 News Classification

As showed in section 3.3, each channel and each item of a RSS feed can belong to one or more
categories. However, in both cases that element is optional. When analysing real news websites
RSS feeds, we found that many of them do not have any reference to the category in the entire
RSS file, only on the RSS feed URL. In some cases, all the items in one channel have the same
category. Finally, and only very rarely, items have appropriate categories, that are contained in
the channel’s category.
Our goal is to automatically apply categories to the gathered items, in order to apply better filters
and generate a better sequence of the news we present to the user.

There are many machine learning techniques proposed to enhance automatic organization of
text data. These techniques can be divided in two main categories, supervised (document clas-
sification) and unsupervised (document clustering)[7].

Document classification is also known as document/text categorization or routing and topic iden-
tification. It can be defined as assignment of one or more predefined categories to documents[8].
This assignment is based on a training set of labelled documents.
There are many methods for document classification, including Support Vector Machines (SVM)[9],
k-Nearest Neighbor (kNN) classifiers, Neural Network (NNet) approaches, Linear Least-
squares Fit (LLSF) mapping[10], and Naive Bayes (NB) classifier. According to [11] results,
that are based on some of these methods implementations, SVM, kNN and LLSF significantly
outperform NNet and NB when the number of positive training documents for each category is
small (less than ten). On the other hand, when the number of training documents is high (more
than 300), all algorithms have equivalent performance.
The main disadvantage of Supervised Classification is that we must assign training documents to
each predefined category. This process is error prone, and also very subjective.

Document clustering, also known as Text Clustering, is an unsupervised technique, which does
not require predefined categories and the matching training documents. As stated in [12], doc-
ument clustering can be divided in two main groups, hierarchical clustering algorithms and
partitioning clustering algorithms.
The first algorithm can build the clusters in two different approaches: agglomerative hierarchical
clustering, where each item starts as an individual cluster, and then the most similar pair of items
are merged; and divisive approach, which starts with all documents in a single cluster, that is di-
vided into smaller clusters, each iteration.
On the other hand, partitioning clustering algorithms, such as **K-means** and its variants, create k non-hierarchical and non-overlapping partitions. On the bright side, document clustering does not need training documents, and can be used to find content similarities that we are not aware (e.g. when we do not have predefined categories). As a down side, the result of the clustering process, may group the documents into unexpected/unwanted categories.

### 3.5.1 Tools For Document Classification and/or Document Clustering

The Machine Learning algorithms described earlier are already implemented in some tools. Next we will analyse some of those tools, and their ability of giving us the information we need.

**Weka**

Weka\(^{12}\) is a collection of machine learning algorithms for data mining tasks. Contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes. Since Weka is fully integrated in RapidMiner, we will step on to the analysis of RapidMiner.

**RapidMiner**

RapidMiner\(^{13}\) is an open-source environment for machine learning, data mining, text mining, predictive analytics, and business analytics. It offers tools for data Classification and Clustering, that are the ones that interest us the most. The analysis of text documents is possible if those documents are turned into word vectors. During the process of transforming the text into word vectors, a large number of text processing actions can take place. These actions include tokenization, extraction, filtering, stemming and transformation. They will usually improve the quality of the classification or clustering process.

When talking about Document Classification, it is possible to train a classification model using one of the available algorithms, and a set of training documents. It is also possible to apply this generated model, and use it to classify other data. At the end of the model generation and of the application of the model we can see each of the model classes distribution and the performance of the model when applied to the second set of documents.

On the other side, Document Clustering, does not need a training set of documents. So, after supplying the chosen clustering algorithm with the documents we want, the process is ready to start. As a result, the documents that were served as input will be split into the several clusters.

**Carrot\(^2\)**

(Carrot\(^2\))\(^{14}\) is an Open Source Search Results Clustering Engine. It contains two document clustering algorithms designed specifically for search results clustering: Suffix Tree Clustering

---

\(^{12}\)http://www.cs.waikato.ac.nz/ml/weka/

\(^{13}\)http://rapid-i.com/content/view/181/190/lang,en/

\(^{14}\)http://project.carrot2.org/
and Lingo. Carrot\textsuperscript{2} also contains components for fetching search results from several search engines, such as Bing or Google, but it also supports other sources of documents like Lucene, Solr or Google Desktop index. We can use Carrot\textsuperscript{2} library through the Carrot\textsuperscript{2} Document Clustering Workbench or through the Carrot\textsuperscript{2} Application Programming Interface (API) and JAR to integrate with Java code\textsuperscript{15}.

With the Workbench it is easy to cluster news from RSS feeds. For this, we have to transform the XML feed into Carrot\textsuperscript{2} format. In order to do this transformation the workbench uses a Extensible Stylesheet Language Transformations (XSLT) document, that has to be created previously. The XSLT document simply selects the relevant data of the feed, such as: title, description and link of each feed item, and rewrites it on the Carrot\textsuperscript{2} format. Having the link to the RSS feed and the path to the XSLT document, we can ask the workbench to process the information an generate the clusters. When the process ends, we can further tune the clustering algorithm's settings. For example, and depending on the algorithm, we can choose the number of clusters and number of labels per cluster.

AlchemyAPI

AlchemyAPI\textsuperscript{16} is a cloud based text mining platform. It can be accessed through the internet using the provided API. This platform is capable of providing a large set of natural language processing procedures including: named entity extraction, sentiment analysis, concept tagging, author extraction, relations extraction, web page cleaning, language detection, keyword extraction, quotations extraction, intent mining, and topic categorization. There are SDK’s available at AlchemyAPI developer tools page\textsuperscript{17} for a large set of programming languages, such as Java, Perl and Ruby.

The topic categorization has support for 8 languages, including: Portuguese, English and French. To categorize text using AlchemyAPI and the provided Java SDK, is very simple. After creating an AlchemyAPI object, we can categorize from an URL or from text with only one method call to the object.

AlchemyAPI offers 1000 calls per day for free after registering, but after proving to be an academic user, we are granted with 30000 API calls per day. If more API calls are needed, bigger packages are available through a given price.

OpenCalais

OpenCalais\textsuperscript{18} is also a cloud based text mining platform. It is very similar to AlchemyAPI, but offers a smaller set of services and supports only 3 languages: English, French and Spanish. Also, the provided SDK’s are somewhat outdated, and the examples aren’t so self explanatory. Currently, OpenCalais offers 50000 API calls per day, for free.

\textsuperscript{15}http://download.carrot2.org/head/manual/index.html#chapter.introduction
\textsuperscript{16}http://www.alchemyapi.com/
\textsuperscript{17}http://www.alchemyapi.com/tools/
\textsuperscript{18}http://www.opencalais.com/
3.6 Sorting News

At this time, and mainly because of the information overload problem it is mandatory that some kind of user preferences are obtained and used to sort the news, before they are presented to each user. Otherwise there would be too many news for the user to read, and most of the ones he would be able to read, probably would not interest him. If user preferences are obtained and used properly, we can provide in first place the most interesting news for each user.

One way of building the initial user model is by asking a series of questions. This approach is taken in [13], where information such as age, gender and occupation are taken as helpful when trying to infer user preferences. But we can not stop after building this first model because the user information needs changes as a direct result of the interaction with the information. This fact was discussed one of the first times in [14]. Taking in consideration some of these previous studies and our own opinion, we feel that updating user's preferences based on user feedback is a very important feature for this kind of service.

One way of updating the user preferences is using the user feedback relative to the news. This feedback can be obtained by forcing the user to directly rate news stories as in [14, 15, 16], by inferring the user’s opinion about a news based on the user browsing activity (e.g. ear the news until the end, skip the news, ask to see other sources for the same news) as in [13, 17], or by taking the benefits of both approaches.

There are a lot of proposals for storing, updating and using user preferences. In [14] a hybrid user model is presented which consists of separate models for a user’s long-term and short-term interests. The short-term model is build based on recent observations, and can adjust more rapidly to the user’s changing interests. These recent observations consist on the n most recently rated stories. The long-term model’s main purpose is to model a user’s general preferences for news stories. In [15] user feedback about publications incrementally builds a user model. Depending on the user feedback each news will be taken as a negative or positive training example.

3.7 Summary

The hardest and more laborious part of the project resides on the gathering and managing the news and user preferences, so it is easy to understand the length of this chapter compared to the previous one.

---

19http://www.evri.com/
20http://developer.yahoo.com/search/content/V2/contentAnalysis.html
We started by defining the meaning of news in this context and by presenting some news aggregating solutions. By studying what these have to offer we can see how the news are currently being presented to the users, and take some lessons from them.

Problems and methods for the news gathering were presented, and the use of RSS feeds seems to be the only viable solution. Also, on each RSS feed item we can find most of information we need and can legally use.

To properly manage the news, a first study was made about classifying and detecting similar news. Concerning classification, many tools that allow text classification were described, and it seems that building a solution from scratch is not the best option when compared to using one of the existing ones. Some of these tools are really mature, are easy to use and can categorize news stories, that is what we need.

Regarding similar news detection, some techniques were presented that are able to compare two documents, and some ideas of what to do with similar news articles were shared.

In the last section we presented some methods of acquiring and maintaining user preferences, which will be used to sort the news.

Classifying, ordering and detecting similar news is very important in order to give the user a better experience, showing him the news he prefers first and not repeating the ones he already knows.
Chapter 4
SGM Requirements and Architecture

In the current chapter the Requirements from Semantic Good Morning (SGM) (the system to be developed) are presented in detail. The system requirements include user stories and use cases. At the end, the Architecture of SGM is exposed.

4.1 System Requirements

In this Section the specification of SGM is presented. This specification includes User Stories, and Use Cases.

4.1.1 User Stories

User Stories are sentences used to capture the actions that the user needs to do. These actions are then used as the base for the functionalities the system will provide. These sentences are usually composed of three parts, the ‘who’ (the entity that executes the action), the ‘what’ (what the user needs/wants to do) and the ‘why’ (why the user wants to do it).

Being part of the system requirements, the user stories were written before we begun to implement the solution. As it is normal, during the development process it was verified that some user stories did not make sense and that some were missing. Moreover, each sentence ends with a priority grade, and if applicable, with the changes made to the initial user story.

US01 - Always available

As an user I want the application available anytime.

Priority: High

US02 - Time by session

As an user I want the option to set the time of each session (in which the application presents the news stories), so the presentation adapts it self to the time I have.

Priority: Medium

Changes: This user story was not implemented. The length of each news description does not vary much from news to news. In addition to this fact, the stories are presented from the most preferred to the last preferred. So, despite the time the user has available to use the application, he will always be presented with the most relevant news.
CHAPTER 4. SGM REQUIREMENTS AND ARCHITECTURE

US03 - Voting

As an user I want the possibility to give positive or negative feedback, for any news.
Priority: High

US04 - Reset preferences

As an user I want the possibility to reset my preferences, in case the system as failed to interpret my feedback.
Priority: Medium

US05 - Login

As an user I want the possibility to have a login, so I can share my Smart TV with other persons, and have my preferences available in other devices.
Priority: Medium
Changes: In order to simplify the use of the application, as soon as the user opens the application for the first time, a new user is created. From this point on, the user is identified on the server by an unique hash. This way, the user doesn’t need to go through the registration process to use the application.
On a future version of the application, this feature would be mandatory, so different users could use the same application and keep their settings.

US06 - Skip news

As an user I want to be able to skip any news, in case they do not interest me.
Priority: High

US07 - Choose categories

As an user I want the option to choose the categories of the presented content.
Priority: High
Changes: This user story suffered small changes, that can be seen in the new version:
As an user I want the option to choose how much I like each available category of the presented content.

US08 - History/User Model

As an user I want the system to maintain an history of my choices/actions in order to learn and adapt to my profile.
Priority: High
Changes: The actions of the users are not kept on the system, but the the result of their actions is: the updated preferences and consequently the updated news rating.
US09 - Add new news sources

As an user I want an option that allows me to add my own sources of information.

Priority: Average

Changes: This user story was dropped. The main reason for this decision was that this feature would require a lot of attention from the System Administration. After an user added a new news source, we would need to manually verify if the source did not include unwanted content, such as illegal or inappropriate.

US10 - Access to other sources

As an user I want that during the news presentation I am informed about other news sources that reported the same event, so I can explore some content more deeply.

Priority: Average

US11 - Access to news original links

As an user I want the original link of each news to be accessible, so I can access the story’s original web page.

Priority: Average

Changes: Since it is not possible to open a native browser from an Samsung Smart TV application, it was decided that the link is shown but there is no way of opening the page it leads.

US12 - Compatible

As an user I want the application to be compatible with other devices so I can also use it on my smartphone, tablet or computer.

Priority: Low

Changes: The access to the Back-end is made through a Web Service, so it is platform independent. Though only the Smart TV Front-end was implemented, any other client side application could be developed without changes to the Back-end.

4.1.2 Use cases

An Use Case is a description of an action that a specific type of User can accomplish. Typically this interaction occurs between a given role (Actor) and a System. In our case, the Actor is always the End User, and the System is the Smart TV Application. The description is composed of the Use Case name, the Actor that performs the action, the pre and post-conditions that have to be true before the action starts and after it ends respectively, and finally by the flow of events necessary to execute the given action.

Tables 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7 present the use case descriptions that we consider the most important:
<table>
<thead>
<tr>
<th>Use Case</th>
<th>Advance news</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The user wants to skip the news that is currently being displayed</td>
</tr>
<tr>
<td>Priority</td>
<td>High</td>
</tr>
<tr>
<td>Actor</td>
<td>System user</td>
</tr>
<tr>
<td>Pre-conditions</td>
<td>The system recognizes the user</td>
</tr>
<tr>
<td>Post-conditions</td>
<td>The news being presented is ignored and the next in line is displayed</td>
</tr>
<tr>
<td>Flow of events</td>
<td>1 - The user asks to skip current news; 2 - System asks if the user wants to send negative feedback regarding the current news; 3 - The user confirms; 4 - The user model is updated; 5 - System presents the next news;</td>
</tr>
<tr>
<td>Alternative 1</td>
<td>3a - The user denies to send negative feedback; 4a - System presents the next news;</td>
</tr>
</tbody>
</table>

Table 4.1: Advance news use case description

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Choose Categories</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The user wants to set the categories that he is interested in and how much he is interested in</td>
</tr>
<tr>
<td>Priority</td>
<td>High</td>
</tr>
<tr>
<td>Actor</td>
<td>System user</td>
</tr>
<tr>
<td>Pre-conditions</td>
<td>The system recognizes the user</td>
</tr>
<tr>
<td>Post-conditions</td>
<td>User model is updated</td>
</tr>
<tr>
<td>Flow of events</td>
<td>1 - The user accesses the options menu; 2 - Selects edit categories sub-menu; 3 - Selects the level of interest for each category; 4 - Asks to save changes; 5 - The system asks for confirmation; 6 - The user confirms; 7 - User model is updated;</td>
</tr>
<tr>
<td>Alternative 1</td>
<td>4a - The user asks to discard changes; 5a - System returns to options menu;</td>
</tr>
<tr>
<td>Alternative 2</td>
<td>6b - The user doesn’t confirm; 7b - System returns to edit categories sub-menu;</td>
</tr>
</tbody>
</table>

Table 4.2: Choose categories use case description
### 4.1. SYSTEM REQUIREMENTS

#### Use Case: Give positive feedback

<table>
<thead>
<tr>
<th>Description</th>
<th>User wants to give positive feedback relative to the current news</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority</td>
<td>Medium</td>
</tr>
<tr>
<td>Actor</td>
<td>System user</td>
</tr>
<tr>
<td>Pre-conditions</td>
<td>The system recognizes the user and is presenting one news story</td>
</tr>
<tr>
<td>Post-conditions</td>
<td>User model is updated</td>
</tr>
</tbody>
</table>
| Flow of events | 1 - User presses positive feedback button;  
|              | 2 - Application sends the information to the server;  
|              | 3 - Server updates user model;                                |

Table 4.3: Give positive feedback use case

#### Use Case: Give negative feedback

<table>
<thead>
<tr>
<th>Description</th>
<th>User wants to give negative feedback relative to the current news</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority</td>
<td>Medium</td>
</tr>
<tr>
<td>Actor</td>
<td>System user</td>
</tr>
<tr>
<td>Pre-conditions</td>
<td>The system recognizes the user and is presenting one news story</td>
</tr>
<tr>
<td>Post-conditions</td>
<td>User model is updated</td>
</tr>
</tbody>
</table>
| Flow of events | 1 - User presses negative feedback button;  
|              | 2 - Application sends the information to the server;  
|              | 3 - Server updates user model;                                  |

Table 4.4: Give negative feedback use case

#### Use Case: Reset user model

<table>
<thead>
<tr>
<th>Description</th>
<th>User is not in line with the way the System inferred his preferences and wants to reset the user model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority</td>
<td>Medium</td>
</tr>
<tr>
<td>Actor</td>
<td>System user</td>
</tr>
<tr>
<td>Pre-conditions</td>
<td>The system recognizes the user</td>
</tr>
<tr>
<td>Post-conditions</td>
<td>User model is updated</td>
</tr>
</tbody>
</table>
| Flow of events | 1 - User accesses settings menu;  
|              | 2 - Selects reset preferences option;  
|              | 3 - Server resets user model;                                                                         |

Table 4.5: Reset user model use case
<table>
<thead>
<tr>
<th>Use Case</th>
<th>Access other sources</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>User wants to access the same news from a different source</td>
</tr>
<tr>
<td><strong>Priority</strong></td>
<td>Medium</td>
</tr>
<tr>
<td><strong>Actor</strong></td>
<td>System user</td>
</tr>
<tr>
<td><strong>Pre-conditions</strong></td>
<td>The system recognizes the user and is presenting a news story</td>
</tr>
<tr>
<td><strong>Post-conditions</strong></td>
<td>The same news is presented but from a different source</td>
</tr>
<tr>
<td><strong>Flow of events</strong></td>
<td>1 - User selects option: other sources; 2 - System presents all the sources that reported the same news; 3 - User selects a source; 4 - System presents the news from the chosen source;</td>
</tr>
<tr>
<td><strong>Alternative 1</strong></td>
<td>2a - There are no other sources for this news; 3a - System presents the next news;</td>
</tr>
<tr>
<td><strong>Alternative 2</strong></td>
<td>3b - The user chooses to go back to the previous menu; 4b - System presents next news;</td>
</tr>
</tbody>
</table>

Table 4.6: Access other sources use case
### 4.2. System Architecture

As mentioned earlier, the system will be made up from two main components, the Back-end and the Front-end.

The Back-end will enclosure all the automatic procedures related to news gathering and management, and includes the following modules:

- **News Collector** - for each existing source, collects the most recent news;
- **News Classifier** - for each collected news, tags it with suitable themes;
- **Similar news Detector and Manager** - for each collected news, checks for existing similar news (about the same event); and treats it if the result is positive;
- **Unread news Selector (for specific user)** - selects all unread news for given user;

---

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Choose sources</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>User wants to choose the sources he likes the most, the ones he likes the less and the ones he is not interested in</td>
</tr>
<tr>
<td><strong>Priority</strong></td>
<td>Average</td>
</tr>
<tr>
<td><strong>Actor</strong></td>
<td>System user</td>
</tr>
<tr>
<td><strong>Pre-conditions</strong></td>
<td>The system recognizes the user</td>
</tr>
<tr>
<td><strong>Post-conditions</strong></td>
<td>User model is updated</td>
</tr>
<tr>
<td><strong>Flow of events</strong></td>
<td>1 - User accesses settings menu; 2 - Selects sources options; 3 - Chooses the sources that interest him and how much they do; 4 - User asks to save changes; 5 - System asks for confirmation; 6 - User confirms; 7 - User model is updated;</td>
</tr>
<tr>
<td><strong>Alternative 1</strong></td>
<td>4a - The user asks to discard changes; 5a - System returns to settings menu;</td>
</tr>
<tr>
<td><strong>Alternative 2</strong></td>
<td>6b - The user doesn’t confirm; 7b - System returns to user preferences menu;</td>
</tr>
</tbody>
</table>

Table 4.7: Choose sources use case

Like the user stories, Use Cases are subject to changes during the development process; relevant changes are explained below.

Concerning the Use Case described in Table 4.1, we decided that when the user asks to skip a news article, the System should not ask if the user wants to send negative feedback. We felt that it was negative to force this kind of feedback. So, when the user skips a news article, no feedback is sent.

The Use Case presented on Table 4.3 was dropped. We decided that all positive feedback sent by the user would be automatic, inferred through his actions.
CHAPTER 4. SGM REQUIREMENTS AND ARCHITECTURE

- **News Sorter (for specific user)** - supplies the user with news, sorted according to his preferences;
- **Preferences Updater** - updates the user preferences regarding specific themes or sources, given the user feedback.

The Front-end will include simpler features, and below are the main modules that compose it:
- **Unread news Requester** - requests unread news from the Back-end;
- **News Evaluator** - sends rating for current news;
- **Source and Category preferences Updater** - requests the change of the user preferences, concerning the categories or the sources;
- **User Register and Authentication** - creates new user accounts and performs users authentication.

Figure 4.1 is a block diagram that explains in more detail the system architecture.

![Block diagram of the system architecture](image-url)

Figure 4.1: Block diagram of the system architecture
4.3 Summary

This chapter registers the planning of the whole system, which includes the Back-end that collects and manages news articles and the Front-end that presents the news on a Smart TV. At first, the creation of the user stories allowed us to identify the functionalities that have to be available for the end user. Some of them, later, suffered some changes and some were not even implemented. The use cases helped on a more technical level, on how each feature would be implemented.

After that, taking in consideration the system requirements, a system architecture was designed. The architecture contains the modules that the system includes as well as what they do. By defining the system architecture, we were also able to determine part of the system’s work-flow.
Chapter 5
Development

In the current chapter the major decisions, algorithms and tools present on the different phases of development process will be exposed. This Chapter is divided into two major topics, the first six Sections, from 5.1 to 5.6 concern the Back-end of the project. The language chosen to develop the Back-end was Java, mainly because both the students that participate in the project are familiar with it, and the language easily provides us the capability not only for the news gathering and management, but also, for the implementation of the web service that will provide the news to the Front-end. The following Sections, 5.7 and 5.8 are devoted to the Front-end and to the final system built, respectively.

A more extensive description of each chapter is given in the next paragraph.

Section 5.1 describes the steps required to collect the news articles and the corresponding on site media, as well as which and how this data is stored. Following, in Section 5.2 it is explained how the transformation of each news article into audio format is accomplished, as well as how the images related to the news stories are obtained. In the next two Sections, 5.3 and 5.4, the process of news classification and similar news detection is exposed. In Sections 5.5 and 5.6 the API end points of the Web Service that enables the communication between Back-end and Front-end is exposed; and details about the the news rating and sorting are presented, respectively. In Section 5.7, the framework chosen to develop the Front-end is explored on a deeper level, as well as the implementation process. In the end, Section 5.8, results are presented.

5.1 News Collection: feed, news content and on site media extraction

The news collection process is composed of three sub-processes: the collection of the feeds and all the information they contain, the extraction of the full article and also the extraction of the news media, both from each of the original stories’ Web page. These topics will be addressed in the following sub-sections.

5.1.1 Feed collector

All the news collection process relies itself on RSS feed services. As explained in Section 3.3, an RSS feed is a document in XML format, where we can find all the important content of several news stories. To read, interpret and store the data present on each XML feed, we need to parse
the document.

Given that the feed documents are XML files, our first approach was to use a regular XML parser. But then we got to know ROME\(^1\), which is a library that includes a set of RSS and Atom Utilities for Java. These utilities include parsers and generators\(^2\).

ROME simplifies the process of obtaining the different elements that a RSS item includes, by providing an abstraction layer on top of the various syndication specifications. In addition, it is already prepared to face some common issues, such as parsing dates that are not in the correct format. The pubDate element should conform to the Date and Time Specification of RFC822\(^3\), as the example given below:

Sat, 07 Sep 2002 00:00:01 GMT

But soon we found feeds that had the date on a different format. To solve this we simply add extra date-time masks to ROME, by creating a rome.properties file with the extra masks. The extra date-time masks that are used in the project are the following:

datetime.extra.masks=yyyy-MM-dd HH:mm:ss|EEE, dd MMM yyyy HH:mm:ss

Which can parse dates in these formats:


So, after failing to read a date that is not compliant with the RFC822 specification, it tries to use the extra masks.

After surpassing the parsing problems, the relevant data is stored on the database. An excerpt of the news table structure is shown in Figure 5.1, which contains the data fields that are taken from the RSS feed.

![news table structure](image)

Figure 5.1: Excerpt of the news database table columns

All the stored data is used in the project. The title and the description are shown to the user; the link is used to extract the news full article and the media available on the page, and is also

---

1https://rometools.jira.com/
2Parsers and generators of feeds
3http://asg.web.cmu.edu/rfc/rfc822.html
shown to the user; the *pub_date* is shown to the user and also used when calculating the order in which news are presented; and finally, taking in consideration that the *guid* uniquely identifies each item, it is used to prevent the same news to be added more than once to the database.

### 5.1.2 Full article extraction

The description of the news that each of the feed items contain, is only an excerpt of the complete news as it is on the website it came from. The complete news article is usually too big to be displayed and read to the users, the way we intend to. In fact, presenting the full article in our application may be a violation of the websites terms of service.

Anyhow, the full article can be of use on other subjects, being that it is more complete than the description. As it will be shown in following sections, the full article will be used on the news classification and also on the entity extraction tasks.

Obtaining the main content present on a web page is also known as *boilerplate extraction*. It removes all the non main content, which includes navigational elements, and advertisements for example. This procedure is far from being simple, and there are many approaches to accomplish it; some are based on Machine Learning algorithms, some in Heuristics and some are site-specific solutions.

Maximum Subsequence Segmentation\(^4\), jusText\(^5\) and Boilerpipe\(^6\) are libraries that can perform *boilerplate extraction*. By using their on-line demonstrations we can confirm that they perform good. Among these options, Boilerpipe was chosen, mainly because it is written in Java, what eases the integration with the rest of the System.

Boilerpipe provides algorithms to detect and remove the surplus around the main content. This library is based on concepts that came out of the paper \([18]\). One of the paper’s authors, Christian Kohlschütterpaper, is also the library’s creator.

The library is very simple to use. With the lines presented in Listing 5.1, the main article, present on the page that is referred by the given *link*, is returned.

**Listing 5.1**: Code necessary to extract the full article from a Web page, using Boilerpipe.

```java
URL url = new URL(link);
String article = new String ( ArticleExtractor.INSTANCE.getText(url));
```

The feeds that are used on the project, are from Portuguese on-line newspapers. So, special characters such as ‘ç’, ‘á’, or ‘ã’, are very common. *Boilerpipe* has encoding problems, when reading this characters. So, after many attempts, we got to a solution that works for most of the cases. This solution converts the extracted *HTML* content into *UTF-8* right after extracting the content from the web page. An excerpt of orginal Boilerpipe source code is displayed in Listing 5.2, and in Listing 5.3, the changes made to the original source code are presented.

\(^4\)http://www.jeffreypasternack.com/software.aspx
\(^5\)http://code.google.com/p/justext/
\(^6\)http://code.google.com/p/boilerpipe/
Listing 5.2: Excerpt containing original Boilerpipe source code.

```java
final byte[] data = bos.toByteArray();
return new HTMLDocument(data, cs);
```

Listing 5.3: Changes made to the excerpt of Boilerpipe source code displayed in Listing 5.2.

```java
final byte[] data = bos.toByteArray();
byte[] utf8 = new String(data, cs.displayName()).getBytes("UTF-8");
cs = Charset.forName("UTF-8");
return new HTMLDocument(utf8, cs);
```

Where `bos` is a `ByteArrayOutputStream` that contains the content of the HTML page and `cs` is the original character set in which the page comes.

After solving this issue, we were able to store the full article in the database, so we could use it later. The library performs very well for most of the feeds that were used in the project.

### 5.1.3 Media on page extraction

The presentation of the collected news articles will occur on a Smart TV device. Televisions, when compared to smart phones, tablets or computers, usually have bigger screens. Also, users are less used to read big portions of text on a television. These are the reasons why we want to give more emphasis on the news presentation to: the news audio, which will be explained in more detail in Section 5.2.1; and to the images related to the news article.

News articles available on Web pages typically include one news related image. The extraction of this image is the focus of this sub-section. The collection of more related news media is discussed in Section 5.2.2.

Due to the nature of the process explained in Section 5.2.2, the images collected through that process are not always concerned with the news article under consideration. On the other hand, the image of the news article, available on the news article Web page, is always related. This is why it is so important to collect this image, when it is available.

The news sources available on the System are static, and can only be changed by the System administrator. Being that the automatic extraction of the image that is part of the main story is a complex task, our approach is a semi-automatic, site dependant process.

In more detail, the steps involved in the image extraction are:

- for each source (Público, Jornal de Noticias, etc.), the first step is to manually identify the HTML element that contains the related image. For some sources, this element is not always the same, and 2, 3 or 4 different tags can be found for the same source;
- add this information to the System;
- when the news articles are being collected, and depending on the news source, find the proper container, and download the image.
5.2 NEWS AUDIO AND RELATED MEDIA

This process is possible because, as stated before, the sources are static. Although it was not implemented, a totally automatic solution was studied. Boilerpipe, the same library used for full article extraction, is also capable of extracting images from the news articles. A small amount of time was spent testing this feature. Usually, Boilerpipe extracts more than the image related to the news article. Being that Boilerpipe is less precise than the described algorithm, we decided not to use it for media on page extraction. Still, it can be used as a starting point to enhance this process.

Apart from the news related images, it was also our initial intention to collect the news related videos, when available. Most of the videos are displayed using Flash what makes them very difficult to download. Also, it was not entirely decided how the videos would be integrated on the news presentation. For these reasons, the videos are not collected.

5.2 News audio and related media

As good as it may be to read a news article from a web page and have access to one or two related images, when it comes to a TV application we need to think differently. A television user is used to sit back and receive all the information, with little effort. Usually, and ignoring subtitles, only small portions of text are supposed to be read by the users.

So, presenting news on a television set as we would present on a web page seems a mistake and a slightly different approach was taken.

In order not to force the users to read the news descriptions, all the description text is transformed into audio format using a speech synthesizer. Also, given that each news story audio takes some time to be played, it would be more appealing to show more than one news related image. Using the entities found on the news story and a web search engine, more images are extracted and shown to the user.

These actions are explained in more detail in the following Sections 5.2.1 and 5.2.2.

5.2.1 Audio

There are available some free speech synthesizers that we could use without having to pay, like eSpeak\(^7\), MARY\(^8\), Festival\(^9\) or MBROLA\(^10\). The problem with these synthesizers is that some do not support Portuguese or when they do, the quality is not very good.

Google Text-to-Speech service was used for demonstration purposes only, due to the good quality of their synthesised speech, even in Portuguese.

This service is accessed through the web, on the same end point as the Google translate. It has some limitations, being that it only does the Text-to-Speech transformation of 100 characters each time.

To overcome this, we start by splitting the text on white spaces, and we obtain a list of words. Then, we build strings with the words from the list, until a maximum of 100 characters is reached. For each of these strings, we make a call to the service, which returns the audio for the text that

\(^7\)http://espeak.sourceforge.net/
\(^8\)http://mary.dfki.de/
\(^9\)http://www.cstr.ed.ac.uk/projects/festival/
\(^10\)http://tcts.fpms.ac.be/synthesis/
was sent. Then, we compile all the audio on a single file. This audio file is associated with the correspondent news story, and is sent to the Front-end when the news is being presented.

### 5.2.2 Images

As stated previously in Section 5.1.3, after trying to collect the news related image from the web page the news was taken, we may already have one image associated with each news. For the reasons stated in Section 5.2, we want to get more images than the one that may or may not exist on the news web page.

As it will be explained in a later section, the cloud based text mining platform *AlchemyAPI*, which was exposed in Section 3.5.1, was the one chosen to categorize the news articles. Fortunately, besides topic categorization, this service also provides named entity extraction.

By extracting the main entities present on an article we can use a search engine to fetch images of the entities found.

Each call to the *AlchemyAPI* requesting named entities, returns the results in *XML* format. Below, in Listing 5.4 we present an excerpt of the output of a call to the *API* of this type.

Listing 5.4: Excerpt of the response given by the *AlchemyAPI* service for a named entities request

```xml
<results>
  <language>portuguese</language>
  <entity>
    <type>Person</type>
    <relevance>0.607453</relevance>
    <count>3</count>
    <text>Julian Assange</text>
  </entity>
  <entity>
    <type>Country</type>
    <relevance>0.504047</relevance>
    <count>2</count>
    <text>Equador</text>
  </entity>
  <entity>
    <type>Company</type>
    <relevance>0.500141</relevance>
    <count>2</count>
    <text>WikiLeaks</text>
  </entity>
</results>
```

As we can see, the result contains N entities, and each entity is composed of type, relevance, count (number of times it appears on the text) and text (that is the name of the entity).

The name of the entity is obviously used when we search for images on the web search engine, and the relevance is also used for us to achieve better results. To get only images for the most relevant entities of the text, we only use the entity if its relevance is greater than 0.5 (relevance varies from 0 to 1).

As stated previously, the named entities are used to perform an image search on a web search engine. There are plenty of web search engines today. Most of them have very good results,
and some of them also have a Search API, that can be used programmatically. *Google* with the Google Custom Search\(^{11}\), *Bing* with the Bing Search API\(^{12}\) and *Yahoo!* with the Image Search\(^{13}\) (which was shut down last year) are just some examples. Bing Search API was the chosen one, simply because it offers more API calls per month (5000) than the Google service (3000).

During the development process, the Bing Search API moved to the Azure Market Place\(^{14}\) which caused changes on how the API is called. This made us redo all the code that was related with the Bing Search API call.

In more detail, for each entity found in the news article which has more than 0.5 of relevance, we make a call to the Bing Search API asking for the first three results for the given entity, also specifying that we only want large images. Each API call returns a XML response, which contains among other things, the URL of the images, that is what we use.

For the three image URL’s the API call returns, we use the first image that is available.

The image gathered for each entity is not always related to the entity, given that it depends on the search engine results, for the search query. The search results for entities like countries or famous people achieves good results, while the results for company names or not so famous people are not so good.

### 5.3 News classification

In Section 3.5 the state of the art for text classification was exposed, as well as the existing tools that perform text classification. From the study of the different tools and services, *AlchemyAPI* was chosen to be used in the project.

It is very simple to categorize text through the *AlchemyAPI* service, using the SDK they provide, which is available in JAVA among other programming languages. It is also very accurate when classifying pieces of text which are not very small. Besides the simplicity, when compared to tools like *Weka* or *Rapidminer*, and the good performance, *AlchemyAPI* can be used for free, in our case with an academic license, which offers us 30000 API calls per day. *OpenCalais* and *AlchemyAPI* are very similar and both perform good on text classification. *OpenCalais*, offers more API calls per day, but it does not support Portuguese language and the SDK provided was significantly outdated. So between *OpenCalais* and *AlchemyAPI*, we chose the second.

*AlchemyAPI* classifies documents into 12 different categories, including: Business, Law & Crime, Science & Technology, Sports and Computers & Internet. When it is unable to classify a given document, the document is classified as unknown. The response given by the service to a text categorization request can be in several formats, including XML and JavaScript Object Notation (JSON). An example of the response in XML format is presented in Listing 5.5.

---

\(^{11}\)https://developers.google.com/custom-search/v1/overview

\(^{12}\)https://datamarket.azure.com/dataset/bing/search

\(^{13}\)Image Search

\(^{14}\)http://datamarket.azure.com/
LISTING 5.5: Excerpt of the response given by the AlchemyAPI service to a text categorization request

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<results>
  <status>OK</status>
  <language>portuguese</language>
  <category>culture_politics</category>
  <score>0.847009</score>
</results>
```

Where: language is the language detected for the submitted text; category is the category the text was classified in; and score stands for the confidence of the deduced category, which varies between 0 and 1 (where higher is better).

To request a news category, we use the full news article. It has more information, so it is easier for the service to classify it and with more accuracy.

5.4 Similar news detection

Early in this document, at Section 1.1, it was explained that the project that culminated in this dissertation would have the participation of another student. It was also mentioned that his contribution would be made mainly on the similar news detection module, more precisely, he developed the code that compares two texts and returns a level of similarity between them. Since it is not my work, I will only briefly explain how this process was implemented.

The integration of the solution in the system, was done with the participation of Ricardo Costa. Therefore, the algorithm for similar news detection is exposed at the end of this section.

In the moment of writing, the development of the similar news detection module is an ongoing task. It already has two versions, being that the first one is very simple and not so precise. On the first version, the nouns are extracted from each text, and then are used to build vectors with the root of the noun and the frequency it occurs on the text. After both vectors are created, the TF-IDF (Term Frequency - Inverse Term Frequency) is calculated for each term, and the cosine similarity is applied, returning the level of similarity of the vectors.

The TF-IDF value is calculated by multiplying the TF for the IDF. These two values are calculated as follows:

- Term Frequency - is the number of times the term appears on the text, divided by the maximum number of times any of the terms appears on the text;

- Inverse Document Frequency - is the total number of documents divided by the number of documents in which the term occurs.

On the newer version of the similar news detection module, the initial vectors with the term (noun) and the frequency of the term in the text are also created. But then the process differs, given that each term present on the vector is replaced with a vector, containing Wikipedia articles that are a semantic representation of the term. Some procedures are applied to the vectors, and at the end the level of similarity between the two texts is returned.

This version is more precise, and one of the reasons for that, is that this version contemplates synonyms or alternative designations for the same concept. This enables the algorithm to give
more emphasis to a concept, for example, when there are many synonyms of a term on the text. Another reason is that the algorithm can disambiguate, for example, the terms apple (fruit) and apple (enterprise). The similar news detection module is only working for the Portuguese lexicon.

5.4.1 Integration in the System

The similar news detection process is applied to each new article uploaded on the system. This process occurs at last, when the system finishes gathering the news, the related media and the classification. Each new article is compared for similarity, with all the articles that have already been processed. In order to narrow the number of times the comparison occurs, the new article is only compared with articles of the same category, and with articles that do not belong to the same source he does.

The justification for these criteria is the following:

- if two news articles are similar, they should be part of the same category, and;

- if two news articles come from the same source, they will not report the same event.

The level of similarity returned by the algorithm, varies between 0 and 1. When the system finishes the comparison process for a new article, any comparison which returned a level of similarity greater or equal than 0.5, means that a similar news has been detected. Among the news that were flagged as similar, it is chosen the one that returned the higher level of similarity.

After detecting two similar news, the system groups them, so they can be presented as similar to the user.

If the news that was already processed is part of a similar news group, the new article becomes part of that group. If not, a new similar news group is created and both news become part of it.

5.5 Web Service

The communication between the Front-end and the Back-end is implemented through a web service. This web service was also implemented in Java and the web server that responds to the Front-end requests is Apache Tomcat\(^{15}\).

The most important API end points of this web service: return the news for a specific user; create a new user and return string that uniquely identifies him; or update the preferences of a given user. A small description of these end points and the parameters they receive are presented in Table 5.1.

\(^{15}\text{http://tomcat.apache.org/}\)
5.6 News Sorting

In section 5.5 it was explained that the news are requested by the Front-end to a web service. These news articles are not fetched randomly but are ordered based on their rating, which depends on many factors. Following, the factors that are used on the evaluation of the rating, and the equation that returns the final value are exposed.

Early in the process, it was clear that it would be important to define the factors that would be significant on the news sorting. It is easy to see, according to what has been described in this document, that the preferences of the user regarding the category of the news is one of the factors. Below, the factors and their description are stated:

- **Personal Category Rating (PCR):** Value that represents the level of preference of the user concerning one category;

- **Personal Source Rating (PSR):** Value that represents the level of preference of the user concerning one source;

- **General Source Rating (GSR):** Value that represents the rating of the source, given by the system administrators. This is important because many sources offer feeds with content of...
very low quality. These sources should be considered less important than others, by the
ordering equation;

- **Is the Most Relevant News on Group (IMRNG):** Ricardo Costa developed an algorithm
  that chooses the most relevant news, among the news on the same group. The algorithm
takes into account the length of the description and the publication date of the news;

- **Rating of the Similar News Group (RSNG):** Value that represents the relevance of the
  similar news group the news may be part of;

PCR, PSR and GSR are values directly obtained from the database, the first two are in constant
change, depending on the user action, and the last one is fixed by the System Administrator. The
value of IMRNG is 100 if the news is the most relevant of the group, and 0 if it is not. No more
details will be added about this algorithm.

RSNG value should be closer to 100 if there are many news on the same similar news group, and
closer to 0 on the opposite situation. So, with the help of Microsoft Excel, and its ‘Add Trendline’
feature, we compute the following equation:

\[
RSNG = -0.0053 * n^4 + 0.3024 * n^3 - 6.2163 * n^2 + 54.562 * n - 70.947;
\]

where \(n\) is the number of news in the similar news group.

To obtain the final rating of each news article, all these factors are used, each one with a different
relevance. The relevance of each factor was based on the expected importance of the factor to
the end user. The final equation is the following:

\[
final\_rating = 0.40 * PCR + 0.30 * RSNG + 0.15 * PSR + 0.10 * IMRNG + 0.05 * GSR;
\]

Again, the relevance associated to each factor was obtained based on the importance we belief
the factor has to the end-user. The category of the news is in our opinion the most important factor,
so it has the highest relevance. News category is closely followed by the the rating of the similar
news group. In spite of most news not being part of a similar news group, when they are, we
think that it should make them very important. The source preference is also important, but less
then the category preference, so it has less relevance. Finally, IMRNG is important to differentiate
news of the same similar news group, and GSR to differentiate news with similar rating but that
come from different quality sources. Because of that, their relevance is smaller.

Even though this is the current version of the \(final\_rating\) equation, both the factors and their
relevance can be easily changed, if necessary.

### 5.7 Client side

Most of the work described in the previous sections of this chapter was part of the news manage-
ment process. Each step gathers more information related to the news articles. This information
helps making the news presentation more appealing and more personal. So, after all the inform-
ation is collected and treated, the system is ready to supply the news to the client.

\[^{16}\text{http://office.microsoft.com/en-us/excel/}\]
In this section, the details of the creation of the Smart TV application, that will present the news to the end user, are exposed. The preference menus, that are part of Ricardo Costa’s work, will also be presented, but in less detail.

Among all the studied Smart TV frameworks, Samsung Smart TV is the one who is having more success. Google TV is very promising, but still is not getting attention from the major part of the consumers. The remaining frameworks, face the same problem that Google TV, and even the information for developers is not so good. So taking these aspects in consideration the client side of the system was developed for the Samsung Smart TV.

Samsung Smart TV gives us the options to develop Flash applications or Web applications. Being that Flash is becoming obsolete and is usually known for performance issues, it was decided that the client side would be a Web application. Developing a Web application for Samsung Smart TV is very similar to developing a normal Web application. HTML specifies the static content, CSS gives style to the HTML content and with Javascript we can add dynamic content and communicate with the Back-end. To develop Web applications, Samsung makes available the AppsFramework which provides high-level wrapper APIs that eases some tasks, such as playing audio or displaying images.

In the Samsung framework, each Web page is seen as a scene, or as it says in Samsung Smart TV, Guide17, “A scene is a kind of layer – it is what the TV user sees and interacts with to perform a single task or a group of tasks”.

The developed application has 5 different scenes: the main scene, which presents the news; two scenes that show the similar news related information (one presents the similar news list and the other presents one of the similar news); and finally, two scenes related to user preferences (one for categories and the other for sources).

5.7.1 Authentication and main screen

It was agreed between the parts involved in the project, that forcing the user to create an account before starting to use the application, would keep away many potential users. So, as soon as the application opens, it verifies if a file with user information already exists on the TV. This file contains a string that uniquely identifies an user. If it does not exist, the application asks the Back-end to create a new user and send the corresponding randomly created string. The system is partially prepared for login/password authentication, but it was decided that it would not be implemented for now.

After the authentication process is complete, the news from the news feeds that the user is subscribing and has not read yet are fetched, and the news presentation begins. Figure 5.2 is a screen shot taken from the client side application, which shows the news presentation page.

5.7. CLIENT SIDE

Figure 5.2: Screen shot of the main page of the client side application

As can be seen in Figure 5.2, there are three sections on the page:

- the news presentation area, which contains:
  - the news title (top left),
  - the related images slide show (middle left),
  - the news description (middle right),
  - and the news source, category and publication date (top right),

- the news play list area, which contains a preview of the news that have been read, and the ones that are still going to be (bottom, above the help bar),

- the help bar, which contains information about keys that control the the application and their functions (bottom, below the news preview bar).

As soon as a new news article is presented, the news audio starts to play. Also, the related images slide show starts to present all the images that were collected for the given news. First, it presents the image that was extracted from the original news Web page, if it is available. Then, it presents the images collected by the Back-end through the Bing Search API. The images are exchanged automatically, as long as the news is being presented. The news article presentation ends, as soon as the audio file ends playing. At this time, the next article’s presentation starts, which changes the whole news presentation area content.

The news play list contains the preview of 6 news articles. A news preview is composed of the first image related to the news article, and the title of the news.

When the application ends the presentation of a news article, the news play list is updated. The preview of the news that has been read is blurred, and the preview of the news that is going to be presented is highlighted. When the application finishes presenting the 6 news included on the
news play list, the play list is filled with 6 more news, if available. When no more news articles are available, the application stops presenting news, and alerts the user that he does not have any more news to read.

5.7.2 Category and Source preferences

As already stated, these two scenes were part of Ricardo Costa’s work, and for this reason, only a brief explanation of their purpose and how they look, are present in this document.

In the categories scene (Figure 5.3), the user can adjust the level of preference for each one of the twelve categories. The same occurs in the sources scene (Figure 5.4), but now, for the sources available on the system. After updating the preferences on any of the scenes, the information is sent to the Back-end, where the news rating for this user are updated, in order to reflect the new preferences.

![Figure 5.3: Screen shot of the edit category preferences menu](image-url)
5.7.3 Similar News List and Similar News Presentation

In Section 5.4 it was explained how the similar news are detected and how the systems manages them. Each news may be part of a similar news group. If the news that is being presented on the Smart TV application is part of a similar news group, the user has access to a new menu. This menu has a list of news, that are on the same similar news group as the one that is being displayed. This way, when an user for any reason wants to access similar news articles, he can do it, by accessing this menu.

As it can be seen in Figure 5.5, the user can navigate through the similar news list, where he has access to the news title, source and publication date, of each similar news.
In the case of being interested in reading one of the similar news, the user can press the correct key, and the selected news article is presented. The scene where the similar news is displayed, shown in Figure 5.6, is very similar to the scene where normal news are presented but without the news play list.

When on the similar news presentation scene, the user can go back to the similar news list. At any time, when on the similar news list scene, the user can go back to the main news presentation.
scene, and continue reading the news, starting from the point it stopped.

5.7.4 Automatic and user requested feedback

As explained in Section 3.6, user information needs to change as a direct result of the user interaction with the information. To do so, it was decided that there would be two automatic ways of sending positive feedback to the Back-end, and one manual way of sending negative feedback.

So, every time an user waits for the application to automatically change the news, it means that he listened to the whole news article. It is considered that the user likes the article and positive feedback is sent to the Back-end.

If the user simply skips a news article, no feedback is sent. It is difficult to assume the user does not like the news article, because he could already be aware of that news.

When the user consults the news related list, it is assumed that the news article he was reading interests him, enough to spend more time reading other news about the same topic. So, when this happens, positive feedback is sent to the Back-end. When compared to the positive feedback sent when the news is automatically skipped, the feedback in this case has more impact on the preferences, because it is understood as a stronger 'i like this article' statement than the previous one.

Finally, while reading a news story, the user has the option to send negative feedback about a news article, by pressing the proper key. After doing so, the news is skipped.

All the feedback, positive and negative, sent by the client application to the Back-end, is relative to the category of the corresponding news. So, the feedback that is given concerning a news article, only affects the rating of the articles of the same category.

The preferences of the sources are not automatically changed, because it is also very hard to assume that the negative or positive feedback not only concerns the content of the news article but also the source.

5.8 Results

When the development phase ends, it is important to assess the overall project and to measure the results obtained, comparing them to the goals that were established in the beginning. The outcomes of the several tasks of this project are presented below.

Main Goal

The major goal, which included creating an application that would present news to users, on a Smart TV, was accomplished. The images shown in Section 5.7 were taken from the last version of the Smart TV application created.

The news are presented automatically as soon as the application starts. The user can also navigate forward or backward to access other news, as well as pause the presentation.

In global terms, it is possible to state that the first and main goal of this project was fully attained.

User preferences: categories and sources

Each user has an user account, created when the application is opened for the first time. This
account, that is kept in the Back-end, saves all the user personal information, which includes user preferences and the news already read by the user.

News articles are collected, categorized and similar news are detected. These processes occur in the Back-end. Then, when all these procedures end, the news are ready to be sent to the front-end.

As it was explained, the System adapts to the user preferences concerning the several categories, based on user actions. Some of the actions that change these preferences are transparent to the user. Others are explicit, and the user only takes them when he does not like a news article. Depending on the user action and on the news being displayed, the System changes the category preference for this user. Each action has different impact.

User feedback is being collected and used, which is an accomplishment of another of the proposed goals.

The actions chosen to change category preferences and their impact, seem to us the ones that lead to the most accurate capture and interpretation of user feedback. Although, we admit that depending on the use of the application, the user preferences can vary very rapidly, or even, lean most of the categories to higher levels of preference. After reading some news (more or less 12), the changes in the preferences can be significant, originating big changes on the news presentation order. These quick changes can be good or bad. Without further testing, we can not evaluate the accuracy of this process.

Source preferences are only affected by manual changes on the sources setting menu. Any change on the source preferences can affect the news presentation. Changing a source preference to zero, stops that source from being part of the news presentation. Other adjustments will also change the rating of the news that belong to this source, which can result on a change on the news presentation order.

A finer assessment requires a large set of tests with end-users.

News rating

News ratings are used to order the news before they are presented to the user. The higher the news rating value, the earlier it will be shown to the user.

The results obtained with the news rating system are very dependant on the equation that calculates each news rating. And, the news rating equation is dependant on the accuracy of user preferences, given that the user preferences are included on the equation.

During the period available to work on this project, it was not possible to perform standardized functional tests with users. So, it can not be said that the adaptation to the user preferences, or the equation that rates the news are very accurate.

What can be said, is that with the proper tests, both the adaptation process and the rating equation can easily be adjusted, leading to better results.

Associated Media

Some tests were made to measure how often news pages have news related images and evaluate how well does the system collect them.
A test was performed with 135 news articles, taken from the following sources: Diario de Noticias\textsuperscript{18}, Jornal de Notícias\textsuperscript{19}, Expresso\textsuperscript{20} and Publico\textsuperscript{21}. From these sources, the system managed to get 83 images. Concerning the remaining 52 articles, from which no image was gathered, only 3 of them had a news related image. Meaning that in 135 news articles, 86 or 61.48% of the news articles had images and that 96.5% of the existing images were captured by the system.

Another test was performed, with 1000 collected news, from the same sources and some more. Given the large amount of news articles on the test, it was not verified on how many pages the system failed to collect the news. The test only shows the amount of images that were gathered from these articles. From these 1000 news articles, the system collected 680 images, which is 68%.

With these tests we came up with two conclusions. The first is that images on news articles pages are quite frequent, they exist in more than half of the tested pages, however they are not always present. So, we can not rely on them as the only source of media. The second is that the news extraction is very accurate, as expected, given the nature of the algorithm (it uses manually entered information about the location of the image on the page, for each news source).

**Global Performance**

It is important to assess the overall performance of the entire news collection process. For the System to be up to date with the most recent news, the process can’t be too long. The performed test was accomplished using a network with a 10MBit internet connection and a machine with the following specifications:

- **Processor**: 2.53 GHz Intel Core 2 Duo;
- **RAM**: 8 GB DDR3 (1067 MHz);
- **Hard Drive**: 250 GB (5400 RPM).

On this test, the news were collected from 10 different sources. The news collection process was executed three times, each one was separated apart by several hours. The database was emptied before the test, but not during the process.

As a result of the test execution, several news and images were collected. The number of images and news collected for each execution are presented below.

- **News**:
  - 1st iteration: 154;
  - 2nd iteration: 30;
  - 3rd iteration: 37;

- **Images**:

\textsuperscript{18}http://www.dn.pt/
\textsuperscript{19}http://www.jn.pt/
\textsuperscript{20}http://expresso.sapo.pt/
\textsuperscript{21}http://www.publico.pt/
The most important metric acquired from this test was the duration of each news collection process. Those results are presented on Figure 5.7.

![Figure 5.7: Performance in seconds of the most important System's actions](image)

On Figure 5.7 we find four different processes, and the time they take on each iteration, as well as the total time of each iteration. The first one is the feed collection, which includes: the collection of the important data from the news feeds; the gathering of the news on page media; the acquisition of the full news article; and the creation of the news audio file. The second is the related images collection, which includes: the gathering of the entities for each news; and the collection of the news related images from the Bing Search API. The third and fourth are, as their name states, the news classification and duplicate detection processes, respectively.

The data present on Figure 5.7 shows us that the first iteration takes longer than the following ones. It is normal, given that when the news collection process is executed for the first time, none of the news present on the feeds, are available on the System. So, in the first iteration all the news have to be collected. This can also be seen by comparing the number of news and images collected on the first iteration, with the following ones. The duration of the individual System actions reduces significantly from the first iteration to the second and third, because they directly depend on the number news being added to the System. That is not so evident on the duplicate detection process. The more news on the System, the more comparisons between news articles. So, on the second and third iterations the number of news that need to go through the duplicate detection process decreases, but there are more news on the System, the new news articles have to be compared to.
5.9 Summary

In this Chapter, the most important part of the development process was reported. It includes implementation details, problems and decisions concerning the Back-end and the Front-end of the system. The Chapter was organized bearing in mind the different modules of Back-end. A detailed explanation of how they work is presented in the different sections. The communication between the Back-end and the Front-end, and the implementation details of the Front-end were also discussed. At the end, the results attained were exposed, along with a brief analysis that compares the goals proposed in the beginning, with the actual outcome.
Chapter 6

Conclusion

When this project begun, two major goals were outlined: the first was to study the main Smart TV frameworks and assess their capabilities; and the second was to develop an application, that presents news articles on the most suited framework, taking in consideration user preferences.

Smart TV concept is still new to most of the general public. Still, Smart TVs are slowly replacing regular televisions on the shops, and are therefore becoming quite common on many homes. Smart TV's are evolving very rapidly, are capable of accessing the Internet, and also of running different types of applications. They have more and more processing power, enabling them to execute complex applications, but they are not as capable as a desktop, for example. Despite these qualities, Smart TV creators and developers are still trying to find the best ways of showing information to the user, and the best way for them to interact with the TV. Most of the users are too accustomed with the small number of features and with the little interaction necessary to use regular televisions.

Among the several Smart TV producers, Samsung Smart TV is at the moment the most usual Smart TV device. It provides quality documentation for developers, and a good SDK which is frequently updated. These were the reasons that led us to chose Samsung Smart TV as the target framework for our Front-end application.

The System developed was divided in Back-end and Front-end. The Back-end executes more demanding tasks, including collecting, classifying and detecting similar news. Choosing news feeds as the source for the news articles was quite a straightforward decision, given that it was the only feasible solution found. News feeds contain information that we are allowed to use, and contain the most important information of the news articles.

What the news feeds do not contain, but the original news Web page does, are: the news related media and the complete news article. The news related media is used on the Front-end, in the news presentation, and the complete news article is used on the news classification and entity extraction.

On the current setting, given that news sources can only be added by the System administrator, the most effective way of extracting the news related media is by using specific rules for each source. One way of making this process fully automatic was studied, and could be implemented with the help of the library Boilerpipe. It was not implemented, but it is an option to consider if the sources become dynamic.

The extraction of the complete article, also known as boilerplate removal, is not an easy task. Sometimes, in a business setting, it is not possible nor advisable to create our own solution for every problem. Some existing tools were studied, and the most suited 3rd party library, Boilerpipe, was chosen to perform the boilerplate removal.
CHAPTER 6. CONCLUSION

The text-to-speech feature is available on some free to use libraries. Listening to a bad quality generated speech, for a long time, can be like a torture. The quality of the speech generated by the free text-to-speech libraries found, is not very good, at least in Portuguese. So, and just for testing purposes, the Google text-to-speech service was used. Before making this application available on the market, the use of this service would have to be replaced, probably by a commercial application.

Many options were taken in consideration to classify news articles, which was thought to be one of the hardest tasks of the project. After studying many possibilities, we chose an online service to perform the classification of the news articles. This service fully fills our text classification needs, so we had no reason to ignore it, making a text classification tool from scratch. It is easy to use and achieves good results, on text classification. While using this tool spared us some work, we are aware that it can limit our options, in future developments. If, for example, we wanted to classify the news in different categories, or add some to the ones available, it would not be possible. So, this solution is very interesting for the current situation but could become useless in a different scenario.

The development of the Front-end application proved simpler than expected. The fundamental knowledge to develop a Samsung Smart TV Web application is the same necessary to develop any Web application. Things like the AppsFramework, are new for a regular Web developer, but HTML, Javascript and CSS are the basis. The main adversities found during the development are related to the Integrated Development Environment (IDE) provided and the documentation. The Samsung IDE is very CPU-intensive, using most part of the CPU since the moment it opens, until it is closed; and also, it is quite unstable. Recently the developers can choose the Eclipse IDE instead. It is expected that at least the CPU problem disappears, but we did not get to test the SDK with Eclipse.

While developing normal Web applications there are tools that allow the developer to inspect and edit the HTML and CSS after the Web page is rendered. These tools are usually provided by the browsers and are very helpful. The Samsung Smart TV Emulator can be started in debug mode, where the inspection of HTML elements is possible, but the edition is not. The lack of this feature slows the debugging process. The documentation is in general quite good, providing very relevant information about the SDK and also some interesting tutorials. Nevertheless, there are some links missing, on the tutorials, and also, some incorrect or missing fields on the API tables.

All the proposed goals were attained, so even though we could not assess the quality of all the tasks performed by the System, we are happy with the results obtained. Working on a business environment, on a project proposed by a real company, was a brand new experience. Although I was not part of one of the existing company developing teams, I had the possibility to watch the routines of the company employees. Also, I had to follow part of the same procedure they do, like registering on a project management application (Redmine) the tasks we intended to accomplish and how long they would take. The work developed along with Ricardo Costa was well organized, and we managed to separ-

1http://www.eclipse.org/
2www.redmine.org/
6.1 Future Work

It is important to stay alert for updates concerning the different Smart TV frameworks, given their fast evolution. The Smart TV market, and the framework capabilities can be very different in the near future. Developing for Samsung Smart TV was in our opinion a right decision, but that can change with time.

Besides classifying, it was also our intention to connect tags to the news articles. The tags would include the most important entities of the news. This idea was abandoned because it would make the System a lot more complex, and would take more time than we had to implement it. Still, we think the existence of tags, and its use on news rating, would significantly improve the news rating accuracy. Other improvement that we think it could be very important, concerns the preferences system. The existence of a long-term user model, that keeps long term user preferences, could help on making user preferences not so volatile.

Allowing users to add new sources to the System would be a good improvement. As a consequence, there would be the necessity to create a simple Back-office to manage the sources added by the users. Dynamic sources would also make it almost mandatory to make the on page image extraction algorithm totally automatic.

The user interface of the Front-end needs to be polished, making it more appealing for end-users. We think that the organization of the information on the screen is good, but the style of the application needs some work.
Bibliography


Appendix A

Sample file for RSS 2.0

appendix/rss2sample.xml

1 From: http://cyber.law.harvard.edu/rss/examples/rss2sample.xml
2 <?xml version="1.0"?>
3 <rss version="2.0">
4   <channel>
5     <title>Liftoff News</title>
6     <link>http://liftoff.msfc.nasa.gov/</link>
7     <description>Liftoff to Space Exploration.</description>
8     <language>en-us</language>
9     <pubDate>Tue, 10 Jun 2003 04:00:00 GMT</pubDate>
10    <lastBuildDate>Tue, 10 Jun 2003 09:41:01 GMT</lastBuildDate>
11    <docs>http://blogs.law.harvard.edu/tech/rss</docs>
12    <generator>Weblog Editor 2.0</generator>
13    <managingEditor>editor@example.com</managingEditor>
14    <webMaster>webmaster@example.com</webMaster>
15    <item>
16       <title>Star City</title>
18       <description>How do Americans get ready to work with Russians aboard the International Space Station? They take a crash course in culture, language and protocol at Russia’s Star City.</description>
19       <pubDate>Tue, 03 Jun 2003 09:39:21 GMT</pubDate>
20       <guid>http://liftoff.msfc.nasa.gov/2003/06/03.html#item573</guid>
21    </item>
22    <item>
23       <title>The Engine That Does More</title>
25       <description>Before man travels to Mars, NASA hopes to design new engines that will let us fly through the Solar System more quickly. The proposed VASIMR engine would do that.</description>
26       <pubDate>Tue, 27 May 2003 08:37:32 GMT</pubDate>
27       <guid>http://liftoff.msfc.nasa.gov/2003/05/27.html#item571</guid>
28    </item>
29   </channel>
30 </rss>
Acronyms

**API** Application Programming Interface 30, 31, 48, 49, 51, 66

**CSS** Cascading Style Sheets 21, 54, 66

**FEUP** Faculdade de Engenharia da Universidade do Porto 16

**HTML** HyperText Markup Language 21, 45, 46, 54, 66

**IDE** Integrated Development Environment 66

**IP** Internet Protocol 20

**IPTV** Internet Protocol television 20

**JSON** JavaScript Object Notation 49

**KVM** Kernel-based Virtual Machine 21

**RSS** Really Simple Syndication 15, 26–28, 30, 32, 43, 44

**SDK** Software Development Kit 20–22, 49, 65, 66

**SGM** Semantic Good Morning 33

**URL** Uniform Resource Locator 27, 28, 30, 49

**XML** Extensible Markup Language 26, 27, 30, 43, 44, 48, 49

**XSLT** Extensible Stylesheet Language Transformations 30