FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

FEUP

Middleware for Facebook: Information Analysis and Filtering

Luís Carlos Calado Lameirão Gonçalves

Master in Informatics and Computing Engineering

Supervisor: Eduarda Mendes Rodrigues (Ph.D.)

Industry Advisor: Telmo Silva

23rd January, 2012

Middleware for Facebook: Information Analysis and Filtering

Luís Carlos Calado Lameirão Gonçalves

Master in Informatics and Computing Engineering

Approved in oral examination by the committee:

Chair: Name of the President (Title)
External Examiner: Name of the Examiner (Title)
Supervisor: Eduarda Mendes Rodrigues (Ph.D.)

10^{th} February, 2012

Abstract

The huge supply of information content in the Internet and its continuous growth makes the selection process more difficult for end users. Many filtering systems have been created to help users wade through the large amount of information, but they proved ineffective, not just because of the information overload, but also because of the type and quality of the information deemed relevant to users.

This particular problem is being experienced by Facebook users. They see their Facebook page incessantly populated with irrelevant content, which significantly hinders reading what is indeed relevant for them. Facebook has made a first attempt at solving this problem by creating their own filtering system,which sorts the user stream based on some notion of relevance. However, their approach is not very effective since it often misses to identify the posts which are indeed the most relevant to users.

Concerned with this problem, our research focused on developing a more effective filtering system, capable of finding the real user preferences. Recent studies point to the existence of an user profile that can be personal and selective, which becomes more efficient and responsive when it is built in a filtering system. So choosing the best features for a user profile is crucial on a filtering system.

The result of our research is the Facebook Filter Plugin, a web-service that uses four different APIs that work together to build a content filtering system. It uses Facebook API, Google Prediction API, Google Cloud Storage and Google Chrome Extensions.

Resumo

A enorme oferta de conteúdos de informação na Internet e seu crescimento contínuo torna o seu processo de seleção cada vez mais difícil para os utilizadores finais. Muitos sistemas de filtragem foram criados para ajudar os utilizadores a percorrer essa grande quantidade de informação, mas estes mostraram-se ineficazes, e não apenas por causa da sobrecarga de informação, mas também devido ao tipo e qualidade da informação considerada relevante para os utilizadores.

Este problema está a ser particularmente vivido pelos utilizadores do Facebook, que vêem o seu mural invadido incessantemente com conteúdos irrelevantes, que dificulta significativamente a leitura daquilo que é realmente relevante para eles. O Facebook fez uma primeira tentativa de resolver este problema criando um sistema próprio de filtragem, que classifica o fluxo do utilizador com base em alguma noção de relevância. No entanto, a sua abordagem não é muito eficaz, uma vez que muitas vezes falta para identificar as mensagem que são realmente as mais relevantes para os utilizadores.

Preocupados com esta problemática, propomos-nos a elaborar um projeto de investigação que ajude a encontrar um sistema de filtragem mais eficaz e que vá ao encontro das necessidades e gosto de cada utilizador. Recentes estudos apontam para a existência de um perfil de utilizador que seja o mais possível personalizado e seletivo, que incorporado no sistema de filtragem a torne mais eficiente e adequada. Assim, a seleção das melhores características é crucial para um perfil do utilizador num sistema de filtragem.

O resultado de nossa pesquisa é o Facebook Plugin Filter, um serviço Web que usa quatro APIs diferentes que trabalham juntas para construir um sistema de filtragem de conteúdo. Ele usa a API do Facebook, Google Prediction, Google Cloud Storage e Extensões do Google Chrome.

Acknowledgements

I would like to sincerely thank the people who helped to make this project possible. First of all, I would like to express my gratitude to my thesis supervisor Eduarda Mendes Rodrigues for guiding me through the right directions.

Secondly, I would like to thank Telmo Silva for giving me the opportunity to work in this very interesting project and also Pedro Abreu for his ongoing support during the development of this project.

To my big friends and colleagues João Sousa for lots of good ideas and helping me get through the difficult times when I was confused about the implementation of this project and also José Campos for supporting me when I needed him the most.

To all the volunteers who participated on the experiments, thank you for your time and comments.

For the support and encouragement, a special thanks to my father, Luís Lameirão, mother, Maria da Conceição and brother, Renato Lameirão.

To João Esteves and my "Zinde Team" (my best and closest friends) André Santos, Bruno Osório, Cândida Susana, Filipe Pires, Graça Carsoso, Isabel Regadas for your friendship, entertainment and caring provided. It was my pleasure to share this fantastic journey with you.

Luís Carlos Calado

Contents

List of Figures

LIST OF FIGURES

Abbreviations

API Application Programming Interface

CSV Comma-separated values

CSS Cascading Style Sheets

DOM Document Object Model

GPL General Public License

HTML HyperText Markup Language

HTML5 HyperText Markup Language, version 5

HTTP Hypertext Transfer Protocol

ID Identification

IP Internet Protocol

JSON JavaScript Object Notation

OAuth Open Authorization

RESTful Representational State Transfer

SVM Support vector machine

URI Uniform Resource Identifier

URL Uniform Resource Locator

XML Extensible Markup Language

Chapter 1

Introduction

Nowadays, surfing on the Internet allows us to a find all the information or content that we need. However, the enormous offer of information and its continuous growth make the selection process increasingly difficult and most part of those contents are not what users are looking for. In order to solve to this problem, many different search engines implemented content filtering techniques, but they didn't prove to be completely successful [DPDM09] [SK10], because this filtering process is often minimal or unsuitable. Recent studies [DPDM09] [BCCO08] report that this problem may involve the creation of a user profile that allows the system to determine the preferences and needs of each user individually. Our study seeks to make a synthesis of different development systems and create an application that ensures a fine filtering for Facebook, while trying to accommodate for the preferences and needs of each user.

This project specifically was developed at Shortcut, a consulting and information technology services company.

1.1 Problem background

Despite the short existence of Facebook, only seven years, its use has become problematic due to the large amount of information that it continuously adds and provides to the user, making the use of Facebook a real difficult task. One of the limitations of filtering techniques in use is that they are too broad. For instance, a user that likes Rock music sees his Facebook's News Feed page flooded with content from other music genres, just because it reflects his friends' tastes. Due to this limitation we can ask these questions:

- How to create a user profile that includes, in addition to gender, age, cultural level and social context, the user's preferences, tastes and needs?
- As the same user expresses different needs, interests and desires according to the time of the day, location, current status (e.g., if he is on vacation or at work), in addition to the owner of a given information, how to incorporate all these variables in the system?

- Which attributes should we integrate in the filtering technique, in order not to become too simplistic and restrictive?

- How to prevent a too thin filtering to leave out important content and how to make sure the user is exposed to the new content that may prove to be relevant for him?

1.2 Objectives

Facebook is a social network which is constantly growing and its importance in people's lives is increasingly significant. Since its use is becoming increasingly burdensome due to the excess of information, our main goal is to finding a technique or a combination of several techniques, which fulfils all the conditions and user's features that allows information sorting and filtering more effective. But what is the best way to study the user preferences? In order to create a custom profile and simultaneously non-restrictive, we propose to take the following methodological steps:

- **Features analysis:** The study of the best features for content evaluation and filtering is a major goal of this thesis. Reading or watching a video or image normally rely on several factors: humour, availability (being at work or at home), the person who shared the information, etc. Therefore, it is absolutely necessary to consider the key features in selecting the personal user.

- **Personal user features selection:** We will seek to use information from Facebook users profiles, which will be obtained from the Facebook API, which allows us to create a custom user profile as comprehensive as possible, integrating data and personal features related to their tastes, preferences and also needs, volitional factors according to day time, context and the source of the information.

- **Create the Filtering technique:** Having acquired all the necessary information about the user and creating his profile, we will try to find a filtering technique which will incorporate this profile in order to make it more effective in selecting the information. This method requires that each user has a personalized filtering model that considers his preferences, tastes and needs.

- **Implementation:** With the filtering model created and with the user features, then it will be applied to Facebook, which will make the information selection according to the user preferences.

With the filtering model created and with the user features, then it will be applied to Facebook, which will make the information selection according to the user preferences.

1.3 Contributions

To solve the problem described in the previous sections, this thesis introduces an architecture that provides an application that will learn the user preferences and will analyse and filter all the information that appears in his Facebook's page, i.e., all the posts that his friends share and appear in the user's News Feed.

Our research focused on finding the best features that would identify the real important information from a post, and use this features to crate a personal training model that will classify and filter the posts according to the user preferences. To do this, we will use a new technology created by Google, called Prediction API, which is a service that uses many Machine Learning algorithms that can automatically recognize trends in data, and then make predictions about new data based on what it has learned.

For a better prediction effectiveness, the user will also interact directly with his *prediction model*. Google Prediction API has the ability to update the *prediction model* by giving new update samples. This will try to prove that, with the inputs collected through user interaction with Facebook combined with automatic retrieval of user preferences, improve the system's prediction processes.

In this thesis, we present a generic architecture for dependency representation and describe a concrete implementation of this architecture. The latter was implemented successfully using Google Chrome Extensions which combines all the APIs (Facebook Graph API, Google Prediction API and Google Cloud Storage) and alters the Facebook page interface and shows which posts are relevant or irrelevant. This demonstrate the effectiveness of our approach in classifying the News Feed information rather than sorting the user stream based on some notion of relevance which is actually what Facebook's current filtering system does.

1.4 Dissertation Structure

The second chapter presents a description of several information filtering and recommendation techniques that currently exist and what applications are using these techniques. It also explains the factors or features that can define the user preferences. The chapter 3 describes the Facebook social network, explaining how it works and what are its main features. Chapter 4, gives a full detail of all used APIs. It starts with a general description and operation, and then provides details about the APIs architecture, their components and technologies involved. Chapter 5 gives a general overview of the Facebook Filter Extension application, describing its challenges and an overview of the implementation and its details. Chapter 6 provides a full description of the Prediction Model, explaining which features and labels were selected and also the experiments and results in order to know the accuracy of the prediction model. Finally, chapter 7, presents some conclusions about the obtained results of our project and also some leads for future work.

Chapter 2

Information filtering and recommendation

The huge information supply on the Internet and its continued growth makes the selection process increasingly difficult for end users. Online shops such as Amazon[1] or ebay[2] contain an extremely vast amount of items that users can purchased, but maybe there is only a small fraction of those items that are actually relevant to users' interests. Another example are the services that allow video sharing, e.g., YouTube[3] and Metacafe[4], contain a huge range of content and are extremely popular with a wider audience. When these services extend its content provision, they complicate the video selection process for end users, who can only watch a small fraction of an overwhelming amount. In addition, there are series of videos that are bothersome or irrelevant or that are not in the user interest field. Most video websites are based in keywords or tags as a searching tool to find specific information. However, this rudimentary searching tool is not able to eliminate irrelevant information. This audiovisual material excess can be treated by a recommendation system that learns the user's preferences and help him find interesting information. Nevertheless, the current recommendation systems focus on metadata or the user history in order to select the relevant information, but it does not consider contextual information or social network relationships.

2.1 Filtering and recommendation techniques

Several technologies [DPDM09] [SK10] have emerged seeking to identify user behaviour patterns (history, research, etc.) using these patterns in the personalization of the relationship with users. Knowing or discovering the real users preferences through various factors, has always been a case study. From several different filtering methods and techniques based on a recommendation system, we highlight three of them: collaborative filtering, content-based filtering and hybrid systems.

[1] http://www.amazon.com/
[2] http://www.ebay.com/
[3] http://www.youtube.com/
[4] http://www.metacafe.com/

2.1.1 Collaborative Filtering

Recommendation systems as a solution to the lack of attention has been studied for some years [PNH08] [RBSC]. Probably, the best known approach is collaborative filtering, recommending items using the similarity of preferences with other users. This approach does not depend on the information of the item, but requires classification by users in their indication of preferences and it infer preferences similarity by overlapping (intersection) of items rated by users. In case of low ratings by users, it suffers from a "cold start". In other words, as initially it has little information on where to "guide", it will not make an efficient classification. The recommendations can be made from information and social processes. For instance, in a social network friends of a friend of mine are a potential user-friendly. User preferences may depend on the context in which information is received. For example, while a user at work will concentrate on the job then it will only displays information related to his work. But at home, that user will have broader interests, including YouTube videos that have nothing to do with his work [CNN+10].

2.1.2 Content-based Filtering

Another method, content-based filtering, is also widely used in recommendation systems. This approach provides recommendations through the comparison of the item content representation from a user with the representation of his main interest [BS97]. This method has been applied mainly in textual domain as news recommendation and hybrid approaches with collaborative filtering.

Content-based filtering is widely used in the Web field in information retrieval and focuses on building a robust user profile.

There are methods to build a robust user profile using common patterns from users "click" history, but these common patterns are static features, thus then this approach is not flexible enough to represent users interests [CNN+10]. In the context of content-based filtering, some authors have proposed a method of recommendation for dynamic content (like news) by building user profiles using metadata obtained from the user: preferences regarding user interests, demographics, and data implicit interaction [SK10]. But one problem is that this method, like with Collaborative Filtering, suffers from a "cold start" and depends on the quality of the user's history.

2.1.3 Hybrid Systems

Usually, a recommendation system compares the user's profile or history with some reference features. These features may belong to information items (content approach) or to their social environment (the collaborative filtering approach). Combinations of both approaches have been researched in so-called "hybrid recommendation systems". The more user and system features are considered, usually becomes increasingly difficult to understand the reasons and circumstances in which the recommendations given match the user profile. Many efforts have been made in research to focus primarily on studying the improvement of the recommendation models using all available knowledge and information from the user profile. However, few studies have addressed

the question of discovering item features, user preferences and system settings that are most significant when it is generated precise and non-precise recommendations. If these characteristics were identified, recommendation strategies could be improved by increasing or non-inclusion of its dependencies to the specific stereotypes of user profiles and information items. Many authors use Machine Learning techniques [BCCO08] [KB06] [BHC98] to analyze and learn which user features and system configurations in a personalized recommendation system are the most appropriate for accurate recommendations. In the proposed approach, for all evaluated recommendations by the user, they created a pattern. Standard attributes correspond to features that are intended to be analysed, and their values are obtained from an information database. It can be attributed two possible values to the pattern class, correct or incorrect, depending on whether the user has evaluated the recommendation as relevant or irrelevant. Sorting these standards, Machine Learning algorithms facilitate preferences analysis described above [BCCO08].

2.2 Information filtering in social networks

Recommendations are part of our daily lives. We usually rely on some external knowledge to make decisions about a particular artefact or action, such as when we go to the doctor or watch a movie. This knowledge can be derived from social processes. At other times, our judgments can be based on available information about an artefact and about our preferences. Several factors can influence a person to make decisions and ideally would be to create a model with all these factors in a recommendation system [SFD+10] [PNH08]. There are several types of approaches to this problem. They make predictions from:

- Evaluations made by a user;
- Based on user preferences areas;
- History of websites/articles visited;
- Only when the assessments are not enough, social-filtering makes sense when the system knows that there is a significant number of users with similar features. This relies on the current system state, number of users and the number and selection of films that have been classified. Thus, for example when a new movie comes out, there will be a short period of time that the number of evaluations is not enough and so the prediction by the system recommendation will not be efficient [BHC98].

2.3 Other applications using filtering techniques

The abundance of information and its related difficulty in finding interesting content items has already been discussed in various contexts [DPDM09][SK10]. Online shops such as Amazon use collaborative filtering to customize their on-line store according to each client needs. Articles/items that customers purchase or evaluate, will represent their interests and are used for

future recommendations. Netflix[5] is a rental DVD-mailing service for customers in the United States. After renting a movie, customers can classify it through Netflix website, with a score of 1 to 5. In October 2006, Netflix published a large set of data and started a competition to find the best recommendation systems. Nowadays, many research groups compete to find the best movie recommendation algorithm in this context.

Lately appeared an user-generated content , such as photos, videos, favorite bookmarks, etc.. Web 2.0 applications use keywords (tagging) to sort contents, which are more pragmatic approaches than the traditional classification systems. Such metadata description, which is the contribution of whole community, is called Folksonomy and became very popular on the Web by 2004. Since users can write keywords in a different way and use other synonyms, recommendation systems have to deal with an additional problem [DPDM09].

2.3.1 Active Learning

Active Learning is a sub-field of machine learning and, more generally, artificial intelligence, in which learning algorithm is able to interactively query the user to obtain the desire outputs at new data points. The key idea behind Active Learning is that it can achieve greater accuracy with fewer training labels [Set09]. In many supervised learning tasks, creating a training set with labelled instance is very consuming and costly. Therefore, Active Learning is well-motivated for this kind of problems, so that the learner can actively choose the training data [TK02] [Sas02]. There are several situations in which active learners may pose queries and there are also different query strategies to decide which instances are most informative. Pool-based active learning (Figure 2.1) can be employed in several scenarios like:

- **Email filters**: users create personalized automatic junk emails, where in the learning phase automatic learners have access to the user history and interactively brings up past emails and ask the user if it is junk email or not. This creates a personalized filter for each user [Set09].

- **Classification and filtering**:learning to classify content information like articles, web pages, videos, etc., requires that users label each content with particular labels, like “relevant” or “irrelevant” [TK02].

[5]http://www.netflix.com/

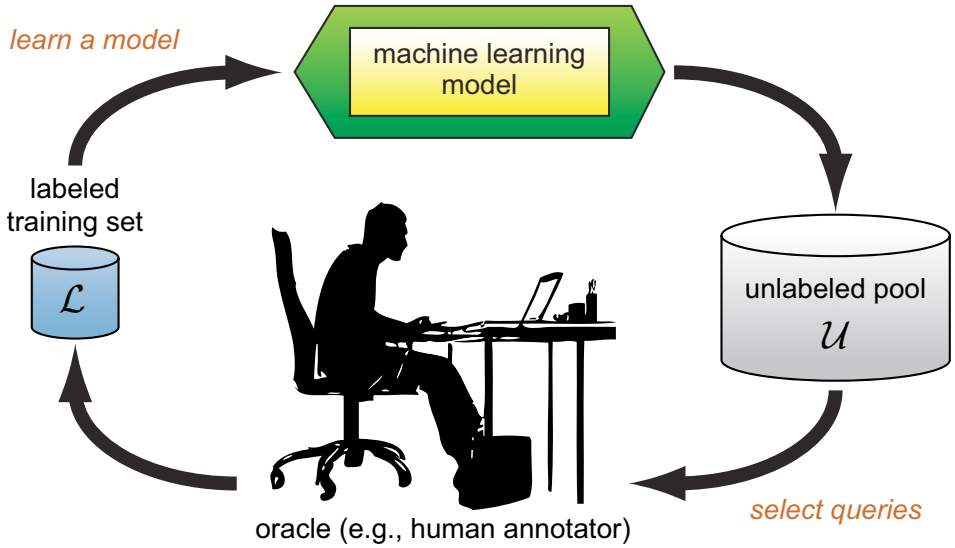

Figure 2.1: Pool-based active learning cycle

2.4 Machine Learning tools

One of the best techniques to solve problems and automate human lives is Artificial Intelligence, particularly Machine Learning techniques. One of its main features is the prediction of future behavior of a system based on data previously collected in a training / learning phase. One of these examples is spam filtering of many email clients in which, through the title, content or the email sender, this can be regarded as suspect or not as spam. Nowadays, there are many of these techniques [CYM08].

Other applications that users typically use in various types of websites are the recommendation system, without even realizing it. A well-known example is the Google search engine. If we do a search with the same word, even at the same time, we get different results [Par11]. These results are dependent on our browser's search history, its location and also historical research of similar users.

There are several tools that have Machine Learning algorithms that assist in the process of text classification. They are Weka[6], RapidMiner[7] and Prediction Google API[8].

2.4.1 Weka

Weka (*Waikato Environment for Knowledge Analysis*) is a set of tools that add various algorithms to analyze data and predicting models, in a graphical interface [Coe08]. This software was developed by the University of Waikato, New Zealand, and under General Public License General Public License (GPL), i.e., the respective code is free to be studied and changed.

[6] http://www.cs.waikato.ac.nz/ ml/weka/
[7] http://rapid-i.com/
[8] http://code.google.com/apis/predict/

Weka was developed in Java and is intended to add algorithms such as Support vector machine Support vector machine (SVM) [Joa98] [Joa01] [ZL06], Naive Bayes, k-nearest neighbour, etc.., which are from different approaches / paradigms in the sub-area of artificial intelligence dedicated to the research of Machine Learning [SFD+10] [PNH08]. This sub-area aims to develop algorithms and techniques that allow a computer to "learn", in order to obtain new inductive or deductive knowledge. The Weka conducts to computational and statistical analysis of data provided using data mining techniques that, inductively, try to predict probable solutions from the founded patterns.

2.4.2 RapidMiner

Other commonly used tools is RapidMiner, an open-source tool that was developed at the University of Dortmund. It was developed in Java and it provides a variety of procedures of data mining and Machine Learning for evaluation and optimization of processes, i.e., can be used for text mining and multimedia.

It uses a visual programming paradigm to facilitate the design of schema representation and it is stored internally as an Extensible Markup Language (Extensible Markup Language (XML)) to enable automated applications after the prototyping phase [KLR+10].

2.4.3 Google Prediction API

Google Prediction API is a service that enables the creation of "smarter" applications. This intelligence comes from capabilities of pattern combinations and machine learning to analyze data that is repeated on a recurring basis and predict possible future results. In other words, it predicts new results through historical data obtained by the user.

Giving a particular set of sample data for training / learning, it is possible to create applications that:

- through the historical views of the user, it can predict which movies or other products the user might like.
- categorize emails as spam or ham.
- analyze comments about a product that determines whether it was a positive or negative comment.
- Can guess how much a user might spend on a given day, giving his historical spending.
- Identify the language.

This API is a Representational State Transfer (RESTful) web service that implements a supervised learning algorithm (Machine Learning), which allows patterns to be tested in these data, always based on examples. This includes analysis of the user's feelings, language identification, product recommendation, fraud detection, among others.

2.4.3.1 Prerequisites

To use this API we need to:

1. have a Google account;
2. create a project "API Console Project" with both active services: Google Prediction API and Google Cloud Storage

2.4.3.2 Operation

To use the Google API and Prediction it is necessary to:

1. Create the learning / training data. It is necessary to create an appropriate learning data for questions we want to answer. This step is the most critical and complex, therefore, its creation requires time for a good planning and to have the learning data required.
2. Load the learning / training data to the Google Cloud Storage using the same tools.
3. Train the model with the learning data. Call the Prediction API learning method with the learning data.
4. Send a query for prediction. The Prediction API will return a response that will be an estimated numerical value or a categorization of the object queried, dependent on the learning data. The larger the data in the model, the better and more accurate results will obtain.

2.5 Summary

This chapter shows that there are already several recommendation systems using many techniques for a better user habits and interests understanding. However, there are several factors that influence these same user tastes, i.e., consumption context (time, place, etc..) has a significant influence on the content selection process. But there is still little scientific research that focus on learning the preferences that are really relevant, which provides efficient recommendations and which imply an anomalous behaviour in the recommendation engine [BCCO08]. Therefore, the main task of this research is to determine which users features are the most important for the true knowledge of their preferences in order to create a user profile, and then build a custom system to learn users needs / preferences and with that, it can filter the information with the content (articles, images, videos, etc.) that are truly relevant and that meet their needs. These preferences can be defined either explicit or implicit in the form of preferences, interests and goals of the user, either by the system parameters and settings.

The effectiveness of this method also depends on the user interaction with the system to ensure and extend the field of filtration. Many studies [KB06] show that with text patterns and with the user help in selecting patterns and features makes the classifier more efficient by making more accurate predictions.

Chapter 3

Facebook Social Network

Facebook is a social network service launched by Mark Zuckerberg in 2004 and quickly became the most used social network in the world today with an estimated 800 million active users.

Facebook has become itself a virtual world. But what began as a mere dorm application to increase students online database, it appears Mark Zuckerberg, "a student" came from Harvard, who has developed an international organization that employs more than 400 programmers, graphic artists, etc. with a estimated revenue of 4.27 billion dollars.

For many people, Facebook is considered a social experience, a place to see old friends and/or colleagues and meet new people. The possibility and the ease of communication and information sharing are the strong points for the success of this social network.

3.1 Main Features

Figure 3.1: Facebook's User Profile interface

Facebook social networkwing platform comprises these main features:

Profile

In Facebook, users create their profile with their personal data and photo. They can also create a list of personal interests, contact information and other personal information.

Wall

The Wall is the original profile space where Facebook users' content is shown which allows posting all kinds of content like messages, pictures, videos, etc.. The Wall of each user is visible to anyone with permission to see their full profile.

Messages and chat

It is possible to communicate with other people in various ways, i.e, through private or public messages, or via chat. Public messages are placed on users Wall.

Photos and Videos

One of the main features of Facebook is the ability to share photos and videos. It's possible to create photo albums with different viewing permissions. One of the interesting features is the possibility of photo tagging, i.e., people who appear in the photograph can be identified with the respective name and Facebook account.

Events

Facebook events are an easy way to get members to give information about upcoming events and to organize social gatherings. These can be public or private, where they can describe the whole event with information about the place, time, description and photographs.

News Feed

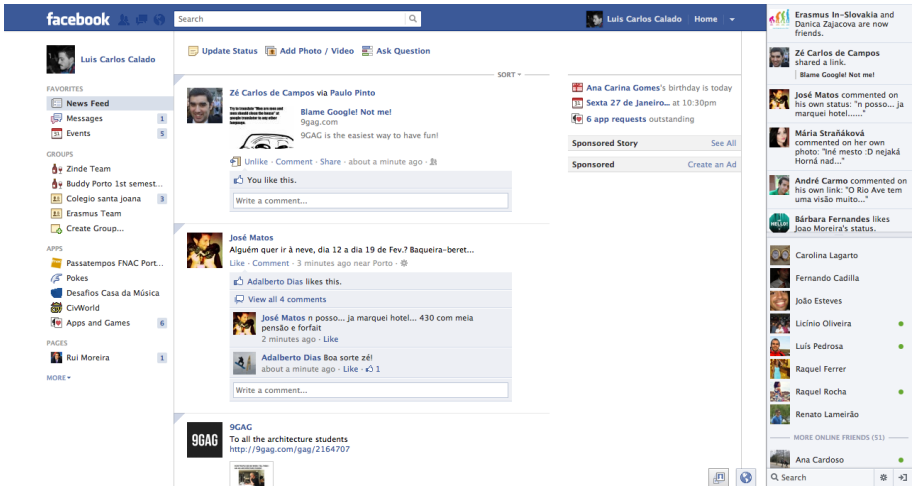

Figure 3.2: Facebook's News Feed interface

News Feed is a list continuously updated with stories and pages of people who users "follow" in Facebook. Here it is shown users publications, photo identification, membership groups, etc.. If a user liked a publication or a video that a friend shared, he can put a "Like" on that post. Users may also prevent friends from seeing updates about several types of especially private activities.

Groups and Like Pages

Facebook has the ability to create pages groups or "Like Pages" where all users can join. These pages are used for discussions, events, etc.. Typically, these pages are used for fans of certain people, brands, advertising, etc..

3.2 Facebook filtering

Due to the large amount of friends that a user has and if each of them share a lot of information per day, the user's News Feed increasingly accumulates more posts, making the process of choosing and reading much more difficult. Herewith Facebook also thought of a filtering method so it created its own filtering system. Unnoticed to the common user, this filtering system exists.

For each user, Facebook has a default setting which displays messages from users to interact more, i.e., friends we follow, talk, put comments and "Likes" are those who remain on our *News Feed*. Therefore, if we do not see some of our "friends" on our *News Feed* does not mean that they no longer exist or do not use Facebook regularly. This Facebook's filtering system "thinks" we do not care to some users just because we recently did not interact with them.

3.3 Privacy

Graph API provides an easy access to all public information about an object, i.e., username, last name and profile photo. For additional information about a user, first we must obtain an authorization for user access. So we can make requests on behalf of those authorized users, including that access token (authorization) on the requests we made to the Graph API. In case we need another type of user information, such as email address or work history, it is necessary to ask for specific permissions.

3.4 Summary

Facebook is a social network the people use to connect with friends and others who work, study and live around them. People use it to upload photos, posts links and videos and also to learn more about the people they meet.

As each user is constantly adding more friends, the huge supply of information content that appears in *News Feed* is also consequently increasing. Therefore, Facebook created a filtering system which sorts the user stream based on some notion of relevance. But unfortunately this

filtering system is not very effective because as it mainly shows the posts from friends that he interacts the most, it hides a lot of information that might be relevant for the user.

Another important fact for Facebook policy is privacy. This was designed to assist users in understanding how Facebook collect and use their personal information that they provide and also to make informed decisions when they use applications that needs access to their personal information.

Chapter 4

Architecture

A recommendation system should work for a particular user as an server able to recover explicit or implicit preferences, content and offers related to items on which the user showed interest [RBSC]. This chapter presents the architecture of the Facebook Filter Extension that we have developed.

4.1 Overview

The architecture is represented in Figure 4.1

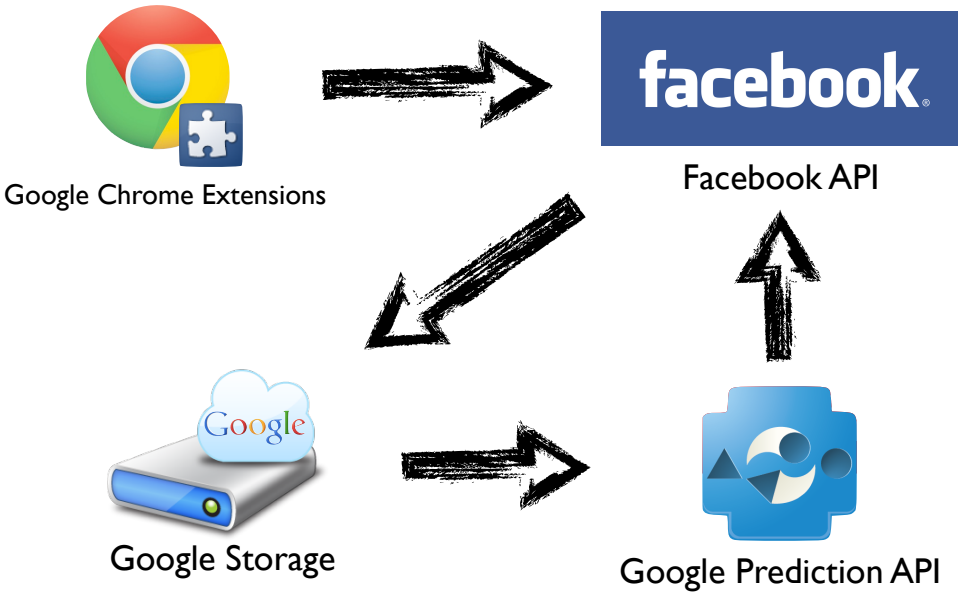

Figure 4.1: Project Architecture

The application runs on the browser as a Google Chrome extension. It adds some changes to Facebook page to "activate" the true application features.

The system uses Facebook API to get all the objects, in this case, users features, that are really important for the evaluation of user preferences. These data is then sent to Google Cloud Storage

where they are stored in the Cloud. After this, the Google Prediction API is used to train the prediction model with the data that was sent to Google Cloud Storage. From here, in the case of reaching a new example or content, the model tries to make a prediction and decide if that content is relevant or irrelevant to the user. Relevant content is coloured in green, and irrelevant content in red. The filtering system relies on the user's "Like" history to train a personalized model. The system keeps adapting the model by receiving further "Like" or "Dislike" signals from the user, which in turns keeps improving the relevance of the filtered content.

4.2 Facebook API

Graph API is the core of Facebook Platform that allows reading and writing data to Facebook. It provides a simple and consistent social graph.

At the Facebook's core it is the social graph where there is all information about people and connections from each user. Graph API provides a simple and consistent vision of Facebook's social graph, which represents all uniformly objects in the graph (e.g., people, photos, events and pages) and the connections between them (e.g., friendship relations, shared content and photo tags). In the social graph, all objects have an unique identification (Identification (ID)), so if we wish to access properties of an object, it is requested the link: "https://graph.facebook.com/ID". Figure 4.2 is an example of the type of properties that we can obtained from an object.


```
{
   "name": "Facebook Platform",
   "type": "page",
   "website": "http://developers.facebook.com",
   "username": "platform",
   "founded": "May 2007",
   "company_overview": "Facebook Platform enables anyone to build...",
   "mission": "To make the web more open and social.",
   "products": "Facebook Application Programming Interface (API)...",
   "likes": 449921,
   "id": 19292868552,
   "category": "Technology"
}
```


Figure 4.2: Facebook API objects example

Alternatively, people and pages with user names can be accessed using the same username as an ID. All responses are the object JavaScript Object Notation (JSON).

All objects of the social graph are connected together via relations, i.e., the user X is a fan of Coca-Cola, then we can examine the connections between these two objects using the following structure for Uniform Resource Locator (URL): "https://graph.facebook.com/ID/CONNECTION_TYPE". These connections support people and pages.

4.3 Google Prediction API

4.3.1 Training the model

For a Google Prediction API usage, we must first train it with a set of training data. At the end of this process, the API creates a Prediction model for that data set. Each model can be a categorization model (if the answer is a string - Figure 4.3) or regression (if the answer is numbers - Figure 4.4). The model learns only through the original training session; it does not continue to learn from queries that we send to it.

The training data is submitted as comma-separated values (Comma-separated values (CSV)), where each row is an example, which consists of a data collection plus an answer (i.e., a category or a value) for the example showed in Figures 4.3 and 4.4. All responses of the training file must be in type or number, we can not mix the two. After loading the training file, it is indicated for Prediction API to train the model with the same file. Then, Google Prediction API examines the file, looking carefully for patterns in the entire file.

```
Description   Subject line
"spam"        "Get her to love you"
"spam"        "Lose weight fast!"
"spam"        "Lowest interest rates ever!"
"ham"    "How are you?"
"ham"    "Trip to New York"
...
```

```
...Most likely: Spanish, Full results: [
  {"label":"French","score":-46.33},
  {"label":"Spanish","score":-16.33},
  {"label":"English","score":-33.08}
]
```

Figure 4.3: Two examples of Categorization Models

```
Temperature  City       Day_of_year   Weather
52           "Seattle"  283           "cloudy"
64           "Seattle"  295           "sunny"
60           "Seattle"  287           "partly_cloudy"
...
```

Figure 4.4: Regression Model example

After training is complete, we can query the model. All queries are submitted to a specific model with an only Internet Protocol (IP) (which is the name of the model file). The Prediction API uses patterns that are found in the training file so that it can find the category closest to the sample submitted (if a model of categorization) or estimate a value for the example (in case of a regression model) based on the training data and returns the category or value. It is possible to update the model, where we can add samples individually. The key to using the Prediction API is to structure the training data so that it return a meaningful response, and that includes all data that may be related to the response.

4.3.2 Google APIs Explorer

To use Google Prediction, Google has a great tool called APIs Explorer, an interactive tool that lets us easily try out Google APIs right from the browser[1].

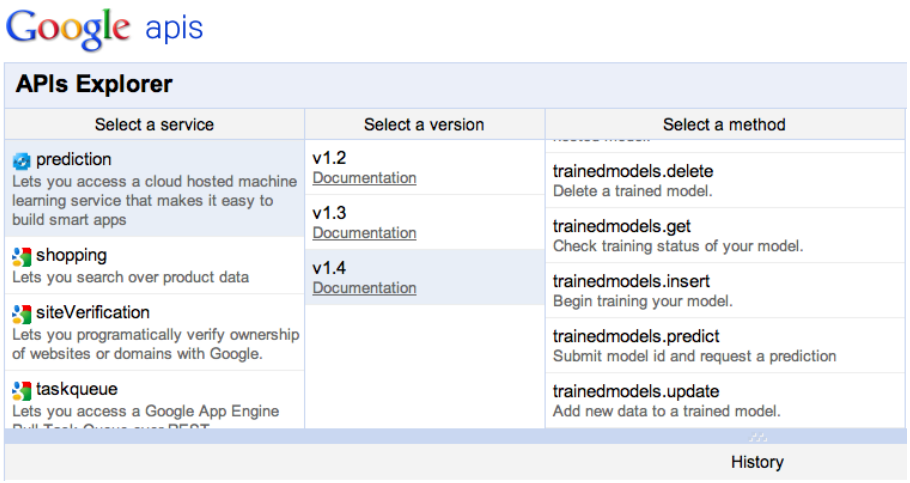

Figure 4.5: Google APIs Explorer interface

By selecting the Prediction API we can see all the available methods and parameters along with inline documentation. The predictions methods available are:

- **trainedmodels.insert:** this is to create the training model;
- **trainedmodels.get:** to check the training status of the model, i.e., classification accuracy, number of instances, number of labels, etc.
- **trainedmodels.predict:** the main method to submit a model and request a prediction.
- **trainmodels.update:** to add new data to a trained model.
- **trainmodels.delete:** to delete a trained model.

As some trained models are private, to get its private access it is needed to "Switch to Private Access" and authorize the Explorer to do so.

4.4 Google Chrome API

An extension is a small software program using web technologies - HyperText Markup Language (HTML), Cascading Style Sheets (CSS), JavaScript, etc. - that add functionalities to Google Chrome browser. Extensions are essentially web pages which can use all the browser APIs, from XMLHttpRequest to JSON or HyperText Markup Language, version 5 (HTML5).

[1] http://googlecode.blogspot.com/2011/03/introducing-google-apis-explorer.html

4.4.1 Architecture

Many extensions have a background page, i.e., an invisible page that holds the main logic of the extension. This may also contain other pages that provide the extension interface. If this needs to interact with other Web pages than those which are included in the extension, we should use a script content (e.g.: flash videos downloads).

4.4.2 Content scripts

If an extension interact with a Web page, then we need a content script. A content script is a JavaScript file that is executed in the context of a page that was loaded by the browser. We could say that a content script is like a part of the loaded page, and not as part of the extension where it is included. The scripts can read the detailed content of Web pages that our browser visit and it can make changes on those pages. In Figure 4.6, the content script can read and modify the Document Object Model (DOM) of the page that is currently being shown. It cannot, however, modify the DOM of its parent extension's background page.

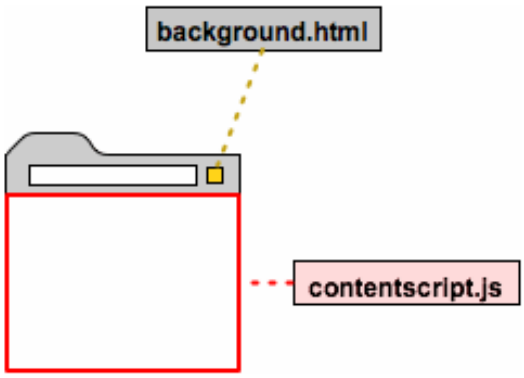

Figure 4.6: Chrome Extension architecture

4.5 Summary

This chapter presented the proposed architecture for the Facebook Filter Extension. It was described all the technologies that we have used to create this application and how they work separately. This way, we can understand better what each API can individually do so then we can explain how they connect with each other. The next chapter describes the actual system implementation and how all of these technologies work together to build the Facebook Filter Extension.

Chapter 5

Facebook Filter Extension

Using the specification of chapter 4 as guideline, and as it was mentioned before, the project's architecture combines four different API's, which work together to create the Facebook Filter Extension.

This chapter presents the most relevant implementation details where it includes the description of the technologies used, the interaction between all the API's and why some features were chosen instead of others to create a specific training model. Some of these implementations are on the client-side, which is the application that is installed. Others are in the server-side where all data is processed. It is fully explained the interaction between the client and the server, in order to aid the understanding of that communication. Finally, it is described how to install the application and how it works and interact with the user.

5.1 Challenges

The goal of Facebook Filter extension is to automatically get user preferences using only a small amount of effort from the user. The design and implementation of a suitable filter system present a number of tough challenges. Ideally, the filter extension should satisfy the following requirements:

- **Low Effort:** the filter extension may solicit input from the user. Thus, the user's input should be natural as Facebook's normal usage, i.e, as Facebook already have default buttons as the "Like" button, it was added a "Dislike" button next to "Like" button, so the the filter extension can use both of them to get input from the user.

- **Visible Data:** In addition to the user's input, the Facebook Filter extension may also use information that it can gather and process automatically, i.e., without any user intervention, the application gets user preferences from Facebook's Graph API so that it should accurately reflect the user's real preferences ("Like").

- **Incremental Accuracy:** As the user provides more input, the accuracy of the resulting predictions should improve.

Uncertainty Sampling

Ultimately, the accuracy achieved by the training model depends on two factors: (1) The user activity in order to provide more input for training, and (2) the accuracy of the training model in predicting the relevant or irrelevant posts.

As Facebook is a service where it only reveals the user preferences ("Like"), there is a large discrepancy between labels, i.e., initially there are no "Dislike" samples that training model could use to learn. That's why from an initial phase, the filtering model has an incremental accuracy.

5.2 Authorization: OAuth 2.0

The Open Authorization (OAuth) 2.0 authorization protocol enables a third-party application to obtain limited access to an Hypertext Transfer Protocol (HTTP) service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing the third-party application to obtain access on its own behalf [EHL11]. In other words, OAuth 2.0 focuses on client developer simplicity while providing specific authorization flows for web applications, desktop applications, mobile phones, and living room devices. This specification is being developed within the IETF OAuth WG [1] and is based on the OAuth WRAP [2] proposal. Applications that access online services often need to access a user's private data. OAuth has emerged as the standard way of letting users share their private resources across sites without having to hand out their usernames and passwords. Google and Facebook have actually implemented OAuth 2.0 endpoints [Goo11b].

5.2.1 Facebook Authentication

Facebook Platform supports OAuth 2.0 flows for user login in the client-side (known as the implicit flow) where the flow is used when it is needed to make calls to the Graph API from a client, such as JavaScript running in a Web browser or from a native mobile or desktop app [Fac11].

User authentication and app authorization are handled at the same time by redirecting the user to Facebook's OAuth Dialog. When the user authorizes the application, the access token is passed in a Uniform Resource Identifier (URI) fragment and where it is retrieved by the application in the client-side.

[1] https://www.ietf.org/mailman/listinfo/oauth
[2] http://wiki.oauth.net/w/page/12238537/OAuth%20WRAP

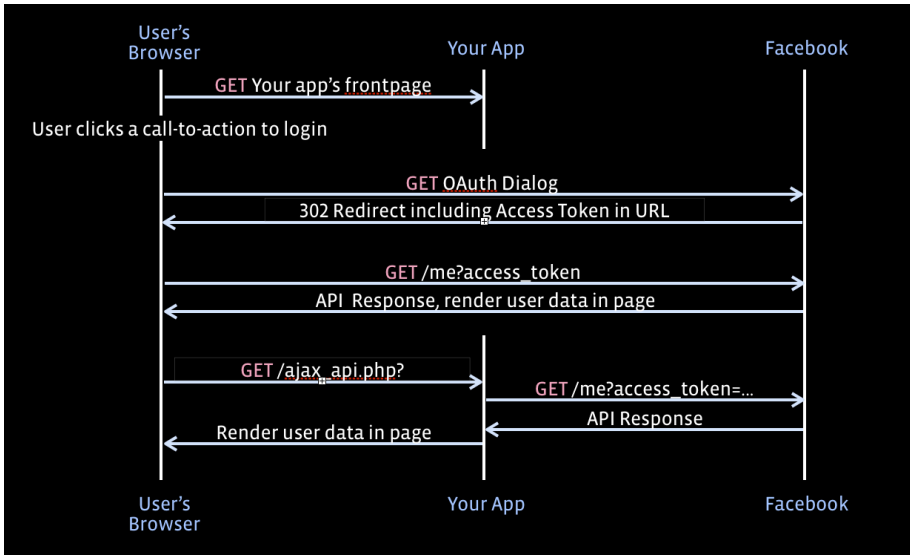

Figure 5.1: Diagram of HTTP calls made through the client-side flow

5.2.2 Google Authentication

Google APIs use the OAuth 2.0 protocol for authentication and authorization. The Google's authentication system can be used as a way to outsource user authentication for applications, which removes the need to create, maintain and secure a username and password store. Google supports several OAuth 2.0 flows that cover common web server, JavaScript, device, and installed application scenarios.

OAuth 2.0 is a simple protocol which can be integrated with Google's OAuth 2.0 endpoints. It needs to register the application with Google, redirect a browser to a URL, parse a token from the response, and send the token to the respective Google API [Goo11a].

Installed Application

The Google OAuth 2.0 Authorization Server supports desktop applications. User authentication begins by redirecting a browser to Google URL with a set of query parameters that indicate the type of Google API access the application requires. Then, Google handles the user authentication, session selection, and user consent where the result is an authorization code. Afterwards, once the application receives this authorization code, it exchanges the code for an access token and a refresh token. With this, the application has now access to the Google API. Once the access token expires, the application obtains a new one with the refresh token. The complete precess is summarized in the diagram from figure 5.2.

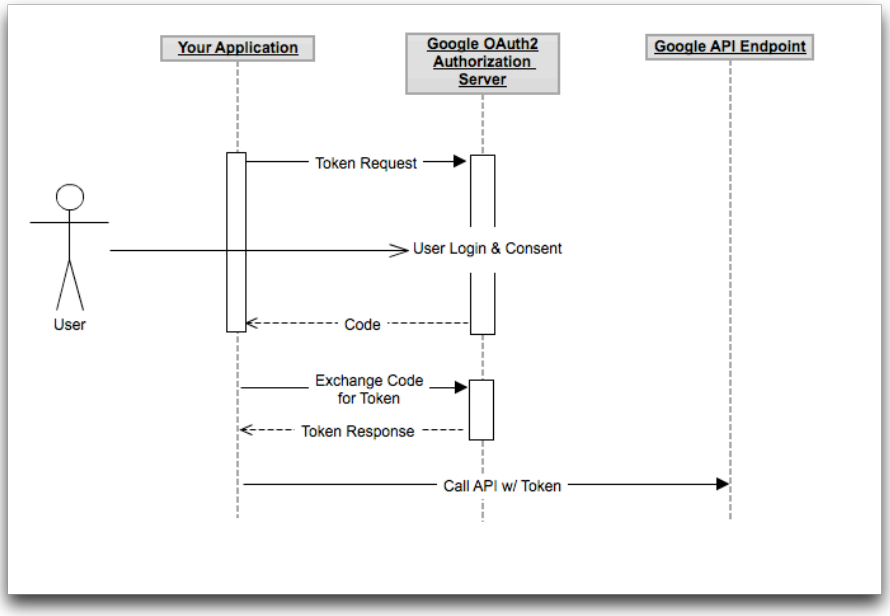

Figure 5.2: Native flow diagram

5.3 Client-Side

In this dissertation, we present a client-side approach using Google Chrome Extensions. Client-side extension technologies such as JavaScript allow sites to extend and "personalize" the behaviours and interfaces of their services, with portable user-interface elements that integrate transparently into the browser's existing interface [FGC+97]. The main objective of the application, in this case the extension, was to be the most user friendly as possible, with only a small amount of effort from the user, i.e., the user's input should be simple in form and also limited in quantity.

The extension is used for three things:

- modify Facebook's page (adding "Dislike" button and color posts).
- communicate with the server (Prediction).
- use Facebook's Graph API: to get user's information and News Feed.

Modelling the HTML DOM

The DOM specification defines a standard for accessing HTML and well-formed XML documents. It is used to inspect or modify a Web page from JavaScript code. Anything found in an XML or HTML document can be accessed, changed, removed, or added "programmatically" by means of a tree structure defined by the DOM [BJDA11]. We used DOM to interact with Facebook's page so we could add some modifications to the page. This gave us the ability to add a "Dislike" button and to color posts according to their prediction results. User Input and Events

Client-side events represent how users interact with Web pages [dSB10]. Most visible HTML tags can react to mouse events, if an event handler was registered. The latter is implemented either by adding a handler function via *element.addEventListener()* or by directly assigning it to the attribute that denotes the event type, e.g. *onclick* [BJDA11].

Since we need the user's input to evolve the predicting model, we have added an event handler to the "Like" an "Dislike" buttons. For instance, when a user clicks the "Dislike" button, this event triggers the update function that gets all the post information, manage the important information and send it to the prediction model. At the same time, to show the user that the button worked, we also used a DOM event, i.e., when the user have clicked that button, it changes the same button with a "Dislike" image.

Collecting User Information

With the structure and data obtained from the Facebook user's Graph, it is possible to extract all the information we need. This information is all in a JSON structure, when we need to parse and evaluate the main content. Initially, in the sampling and prediction model creation phases, to create the prediction model it needs to access the graph to get the user preferences or, in this case, the "Like" history. For the user history, we have chosen the posts that the user have shared or comment and also the "Like Pages". This gives us a great reliable information about the user preferences, so this way we can use it to create the prediction model. For the prediction phase, we need to search for the new content information that is showing in the user's Facebook Page. So, we also need to use the graph, particularly the information from News Feed. Here, we need to evaluate the main features of that post so that we can use them as input to send to Prediction and then get results, i.e., each posts has a respective color corresponds to the prediction result ("Like" or "Dislike").

Cross-Origin XMLHttpRequest [3]

Modern web applications are taking advantage of the *XMLHttpRequest* object to provide seamless communication between the client and the server, by eliminating the need for a complete refresh of the web page [SR06]. This extension communicates with the server and also with the Facebook's Graph API, and to do this, it uses *XMLHttpRequest* to make HTTP requests. The request methods used were GET and POST.

5.4 Server-Side

The server, with which the client application communicates, have many methods and functions that receives and format all the data from the extension. It is also a "bridge" for communicating with Google Prediction API, i.e., all the information is received from the client where it formats all the need data and sent it to Google Prediction depending on the called method. There are

[3] http://code.google.com/chrome/extensions/xhr.html

three prediction methods that are being used by the Facebook Filter Extension: Train, Predict and Update. Each of this methods make HTTP request, Train and Predict methods make POST requests and Update method use PUT request. In section 5.5, there are each method diagram, where it is show how all APIs communicate with each other.

5.5 User Interaction and APIs communication

As proposed in section 5.1, one of the challenges was to create an application that would be extremely simple to use, well-suited for typical (non-technical) users and that would use only a small amount of effort from the user. The hardest work would be made automatically on the background, where the client-side and server-side work together to process all the steps to get the right information and then communicate with the Google Prediction to get the needed results.

5.5.1 Train

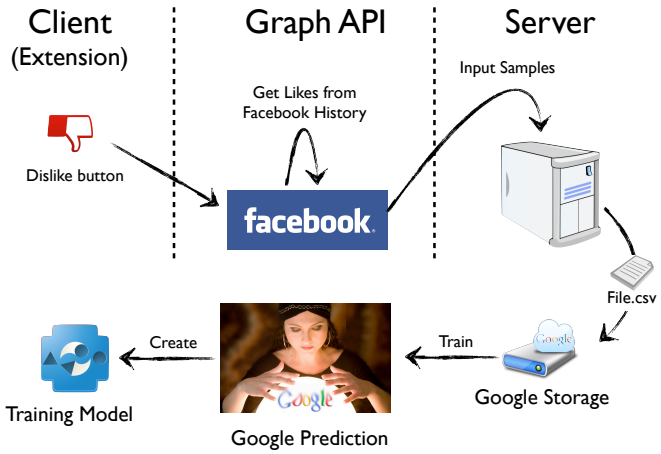

Figure 5.3: Training diagram

In the sampling and prediction model creation phase, to create a training model it is necessary that exists two different labels at minimum. So, to "activate" our application, the user needs to click in a post that he doesn't like. Therefore, when this happen, it gets that post's information from Facebook's Graph where at the same time the extension is extracting all the like history from the user. When it finished this extraction, all this information is sent to the server where is saved into a CSV file that is then sent to Google Cloud Storage. After this stage is complete, the server gets the user's CSV file path and use the POST method to send all the training configuration so the Google Prediction API creates the user's training model with the user's preferences data.

5.5.2 Predict

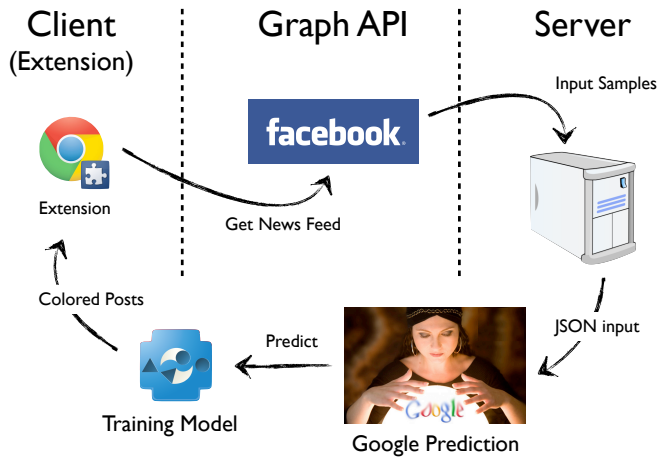

Figure 5.4: Prediciton diagram

When the user starts the Facebook main page and the training model has already created, the extension gets all the posts from News Feed, which are then sent to the Server. Here, this data is formatted into a JSON format, which is then included in the Predict calling method. Therefore, the Prediction API returns an answer with the probability for each label and with his prediction result, i.e., “Like” or “Dislike” . This way, The Facebook Filter extension saves this information in an associative array and, as it “reads” the Facebook page throught HTML DOM, it searches each ID post in the associative array and applies the corresponding colour to the respective post in the HTML page. Figure 5.5 shows an example of the outcome of the predictions with relevant posts highlighted in green and irrelevant ones in red.

Figure 5.5: User interface: green for the relevant post and red for an irrelevant post.

5.5.3 Update

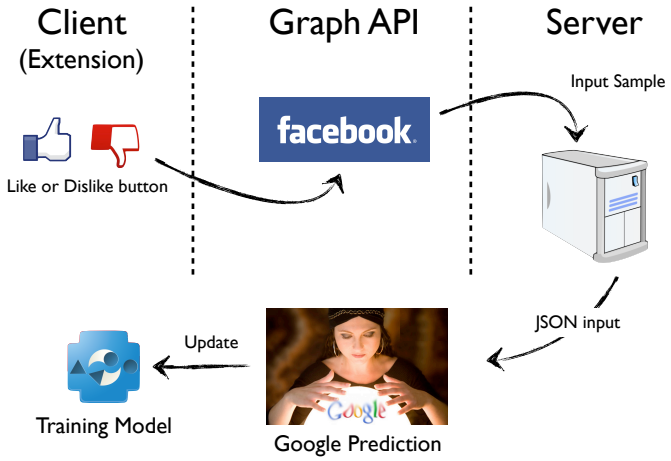

Figure 5.6: Update diagram

In order to make the prediction model to evolve and improve its predictions effectiveness, we needed the user's feedback about the predictions relevancy and, at the same time, that it would be the most user friendly. So, to do this, we added a click event to the "Like" and "Dislike" buttons so it would get the user's input without to much user's effort. This way, both buttons have an invisible feature for the user, that when clicked, they get all the post information from Facebook's Graph and then send it to the Server. Again, this data is formatted into a JSON format and which is then included in the Update calling method that updates the user's training model. For this method there is no need to use Google Cloud Storage because the HTTP PUT request updates the training model directly in the Prediction API.

Google Prediction API does not update the training model each time it receives a new input sample. The current Prediction API waits until it receives a certain number (as many as 100) of update samples before it actually performs the update. Once it gets enough samples, it then updates the training model.

5.6 Summary

This chapter explained all the relevant implementation details of the Facebook Filter Extension. In response to the challenges that we proposed, we developed this application considering the design principles described in chapter 4. For the implementation of the application, this had to be separated in two sides: client-side and server-side. Client-side is the extension (installed application) which has the ability to extract all the Facebook information from the user and also that changes the Facebook's main page. It adds a "Dislike" button next to the "Like" button and also colors posts depending if they are relevant or irrelevant. To do this, it communicates with the server by sending all the needed data. In the server-side, all queries are then sent to the Google Prediction

to create or update the training model and also to make predictions. The response from Google Prediction it is analysed by the client-side, where it uses it to apply on the Facebook's page.

Chapter 6

Prediction Model

As mentioned in chapter 2, many researches [DPDM09] [BCCO08] were done in order to find the perfect filter/recommendation system using several techniques. In our project we decided to use Google Prediction API because it is a new technology created by Google and which experiments showed us great results as we demonstrate here. Also, if the sampling process is interactive, it is important that the classifier used in that phase be efficiently updatable, and Google Prediction has the ability to update the training model. This is very useful as our application needs user interaction to obtain the desire outputs so that it can be more accurate and precise in its predictions. Given this form of user interaction, it is natural to view the training model as a binary classifier, trained using the training data (posts) that the user has labeled. In the machine learning literature, this scenario, in which the learner requires that users label each content with particular labels, is commonly known as active learning [FL10].

Google Prediction API learn patterns from historical data where Google's machine learning algorithms can automatically recognize trends in data, and then make predictions about new data based on what it has learned. Officially, Google does not give to much information about their machine learning methods. Google's service provides a kind of a scalable machine-learning black box, i.e., data goes in one end, and predictions come out the other. Google's black box actually contains a whole suite of different algorithms. When data is uploaded, all of the algorithms are automatically applied to find out which works best for a particular job, and the best algorithm is then used to handle any new information submitted [Goo11c].

6.1 Feature Selection

In order to build an effective classifier, it is important to select a good set of features. For the purposes of this work, we had to identify the best features from a post, so we tried to find the answers for the question:

- Which features from a post makes a Facebook user to read, comment or put a "like"?

From the content of post obtained with Facebook's Graph API, we made a full analysis to the post' JSON file, and so we identified these key features:

- **Title:** tags from a title or message of the post (e.g., "This is funny", "Rock in Rio Lisboa",...).
- **Author:** the ID of the post's owner or author, i.e., the friend that shared the post (e.g., "John", "Kathy").
- **Type:** sometimes a type of information is important. For instance, sometimes a title doesn't call our attention (e.g., ballet) but if it is a funny photo, maybe that post is relevant.
- Number of friends that shared/liked the same post: if more than 3 friends shared the same post, maybe that is relevant information for the user.
- **Friend Type:** sometimes the friends type, like family, is important for the relevancy of a post.
- **Work or leisure related topics:** Reading or watching a video or image normally rely on several factors (e.g., humour and availability - being at work or at home).
- **Day time:** the time of the day can also influence our content choices. For instance, normally, in the morning, a person wants to know the news and the weather for the day. But in the afternoon, maybe that is not so relevant.

From this features and from the research that we have made, we think that there are three main features that are the most important for a relevant post: title, author and type. The title is essential for the relevancy of a post, because it reveals the main subject of that content. The author of the post is also crucial to influence a user to read a content, because one factor that has great weight in deciding a user comment or put a "like" facebook, is the level of proximity in relation to the author of the post, especially if it's family or close friend, even if the title of the post is not from the user main interest. Finally, another important feature from a post is the type. There are many types of posts: video, link, photo and status. For instance, if a person don't have time or patience to read the news and prefers to see a video, maybe the link of a subject is irrelevant for him, but if he sees a video, maybe that same subject can be relevant.

Survey

To get some users opinion, we also made a survey where we asked one question: "Which main features influence you to read, put a "Like" or comment a post in News Feed from Facebook?":

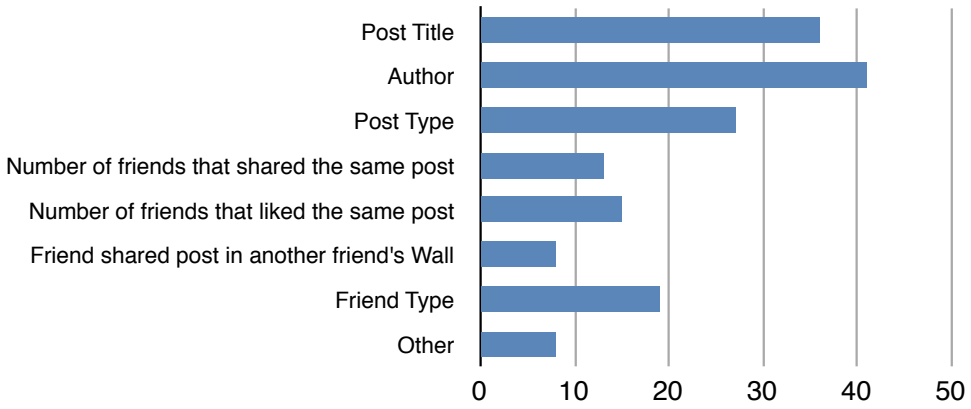

Figure 6.1: Survey chart: number of times features were selected

From the 80 answers,obtained clear top three main features: Post Title (45%), Author (51%) and Post Type (34%). There is no doubt that the post title, the owner (Author) and the type of the post have a great influence in the post's relevance to the user. Therefore, this survey confirmed out intuition about the key features.

Implicit Features

As the main post features are selected, there are also other features that are implicit. To learn the user preferences, it is important to know where to get them. Facebook's Graph API offers rich data that enables us to determine almost everything that the user does. So one important factor that we think relevant for the user preferences, is that if a user shares posts, that because the content of those posts was to some extent relevant. Therefore, to get those shared posts, the application gets the title and the type of the posts he shared on his *Wall*.

Another implicit feature to get the user preferences is the *Like Pages* history, i.e., when a user wants to know updates from a music band, places that he goes, he can search for those interest in Facebook and if they have a Facebook page, the user can put a "Like" on that respective page. Thus the ID of that page is readily available, so we know that if that ID (author) shared a post, maybe it is likely to be important for the user.

Inevitable Situations

There are situations that are hard to identify because sometimes there are cases where sentences have irony. For instance, if I like a post where it says: "Now I am convinced that I don't know nothing!". For sure that this is not true, but the training model will believe it is.

6.2 Model Training

As each user has his own preferences and tastes, a training model has to be consequently different for every one of them. Using the extracted features and user input, the Facebook Filter Extension constructs a *training model*, which is used to automatically "read" and filter the user's News Feed. As the user provides more input, the *training model* evolves and adapts automatically. Therefore, our research focused also on finding the most important set of features that represent the real user preferences. To do this, we had to use Facebook's Graph API to get all the information we needed from the user and then chose the features that we thought were the most important for the *training model*.

Training sets are generally tables or matrices where each row in the table represents a single training example. To create a training model for Google Prediciton API we need to create a CSV file, where the first column in the table is the label or the classification for that example, and the additional columns are features of the training example. As we want to design the appropriate training data for a good filtering model, we have to select the main labels and features so then we can upload this CSV file to Google Cloud Storage where it will be used as a path to create the *training model*.

6.2.1 Labels selection

This application is a filtering system, so when a query is sent to the prediction model, it will decide if that content is relevant or irrelevant and then return an answer with the corresponding categorization. However, as this model has to learn with the user's input, it has to be attributed two possible values to the pattern class, correct or incorrect, depending on whether the user has evaluated the recommendation as relevant or irrelevant. Therefore, the user's input should be natural as Facebook's normal usage.

So, as Facebook default page has already the "like" button, we decided to use this as the user relevant-feedback input. But we also want to know the irrelevant inputs from the user, and as Facebook does not have a "dislike" button, we added one next to the "like" button to get the irrelevant-feedback input.

Figure 6.2: Dislike button interface: before and after the dislike button is clicked

Figure 6.2 shows the two buttons we use to collect feedback from the user.

6.3 Experiments

The goal of our experiments is to analyse and evaluate the performance of the *prediction model*. Specially, we want to answer the following questions:

- Which features are the most useful for predicting the relevant posts?
- How effective is the Facebook Filter Extension compared to the Facebook filtering system?

In order to answer the first question, and to have immediate results before implementing the full application in Facebook, we have used Google APIs Explorer to make experiments with out experimental training models.

Experiments with real Facebook Users

To evaluate our solution, we conducted a detailed study of real users. Our experiments with real Facebook users involved colleagues from the University of Porto and also from Shortcut company. The objective was to active their training model and give many update inputs as possible to see if the prediction model would evolve. To be allowed to use the Facebook Filter Extension, it was necessary to add the user to the API Console, i.e, add manually the GMAIL address to the API Project Team to have access to the API.

6.4 Evaluation Measures

Many different measures for evaluating the performance of information retrieval systems have been proposed. All common measures assume a ground truth notion of relevancy, i.e., every content is know to be either relevant or non-relevant to a particular query [BYRN08]. We propose that the standard measures recall, precision, accuracy and F-measure are adequate measures to compute the effectiveness of our filtering approach.

Accuracy: In the fields of science, engineering, industry and statistics, the accuracy of a measurement system is the degree of closeness of measurements of a quantity to that quantity's actual (true) value.

$$accuracy = \frac{TP+TN}{TP+FP+TN+FN} \tag{6.1}$$

Precision: the precision is the degree to which repeated measurements under unchanged conditions show the same results. [33]

$$precision = \frac{TP}{TP+FP} \tag{6.2}$$

Recall: also called sensitivity, measures the proportion of actual positives which are correctly identified as such.

$$recall = \frac{TP}{TP + FN} \tag{6.3}$$

F-measure: A measure that combines precision and recall is the harmonic mean (average) of precision and recall.

$$F - measure = \frac{2 \times (precision \times recall)}{precision + recall} \tag{6.4}$$

K-fold Cross-Validation

Cross-validation is a statistical method for validating a predictive model. Subsets of the data are held out, to be used as validating sets, a model is fit to the remaining data (a training set) and used to predict for the validation set. Averaging the quality of the predictions across the validation sets yields an overall measure of prediction accuracy. In cross-validation, the original data set is partitioned into smaller data sets. The analysis is performed on a single subset, with the results validated against the remaining subsets. The subset used for the analysis is called the "training" set and the other subsets are called "validation" or "testing" sets [rag09].

K-fold Cross-Validation mimics the use of training and test sets by repeatedly training the algorithm K times with a fraction 1/K of training examples left out for testing purposes [BG04]. For instance, for each of K experiments, use K-1 folds for training and a different fold for testing .This procedure is illustrated in figure 6.3 for K=5.

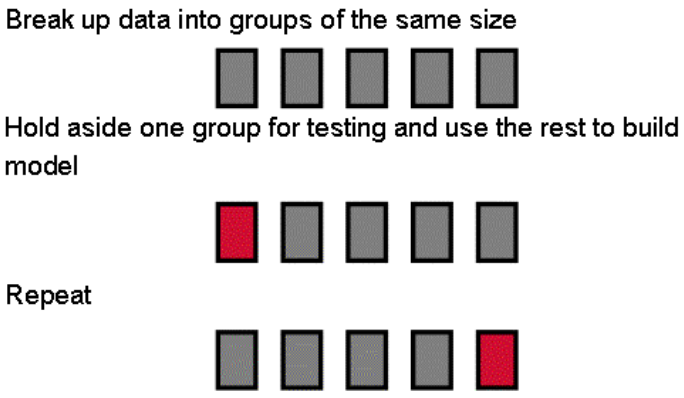

Figure 6.3: K-fold Cross-validation example with K=5

6.5 Results and Analysis

As the basis for our evaluation, we have made some experiments using real training models. Some of our colleagues, who are also common Facebook users, have tested the application for some time where they got the chance to create their own training model and as they constantly used

Facebook, they were giving iteratively more preference samples. This gave us an opportunity to use this training samples as real data for evaluation.

Therefore, for these evaluation experiments, it was used a five-fold cross validation: Training and Testing is performed 5 times. It was chosen two training samples from those Facebook users, each of them with 100 instances, i.e., 50 “Like” and 50 “Dislike”. The initial data were randomly partitioned into 5 mutually exclusive subsets: S1, S2, S3, S4 and S5, each of them with the same size, 20 instances each. So for an iteration i, partition Si is reserved as the test set, and the remaining four partitions are collectively used to train the model. So each sample is used five times for training and once for testing. There will be three different experiments, E1, E2 and E3, where some will differ in the number of features and others in number of training set samples:

- **E1:** training data comprises two post features: title and owner.
- **E2:** training data comprises three post features: title, owner and type.
- **E3:** training data comprises the same three post features as E2 but includes update samples, i.e., the training set contains an additional 50 instances - 25 “Like” and 25 “Dislike”.

To compute the effectiveness of our filtering approach we used the standard measures recall, precision, accuracy and F-measure for each experiment. We averaged each measure, to see how the predictions were evolving in each experiment.

From the *prediction models* that we get from Facebook users, we chose two for these experiments. We extract the training sets from those *prediction models*, where we collected 150 instances, i.e., 100 instances (50 “like” and 50 “dislike”) for experiments E1 and E2 and more 50 instances (25 “like” and 25“dislike”) for E3 as update samples.

Tables 6.1 and 6.2 show the results for each set of k-fold in each experiment, all with the same number of “like” and “dislike” samples. For instance, in E1 and S1, from the 100 instances, we had collected 80 instances for training and 20 instances for predictions.

6.5.1 Model 1

As the results shown in table 6.1 for the experiment 1 (E1), with only two chosen features (Post title and type) the average for all the evaluation measures was above 80% of success, i.e., an average of more than 8 of a set of 10 posts were correct. For the experiment number 2 (E2), with the addition of one more feature (Post type), the average evaluation increased to above 90%. And finally, for the experiment 3 (E3), with the same three features selected as E2 but this time with an update with 50 more instances (25 “like” and 25 “dislike”), the average almost reached the 100% of success. To understand the results better, Figure 6.4 illustrates very well this performance evolution when features and instances are added to training model.

		S1	S2	S3	S4	S5	Average
E1	Accuracy	0.90	0.75	0.80	0.85	1.00	0.86
	Precision	0.83	0.83	0.83	0.90	1.00	0.88
	Recall	0.86	0.67	0.67	0.82	1.00	0.80
	F-Measure	0.85	0.74	0.74	0.86	1.00	0.84
E2	Accuracy	0.95	0.95	1.00	0.95	0.95	0.96
	Precision	0.91	0.91	1.00	0.90	0.91	0.93
	Recall	1.00	1.00	1.00	0.90	1.00	0.98
	F-Measure	0.95	0.95	1.00	0.90	0.95	0.95
E3	Accuracy	0.95	1.00	0.95	1.00	0.95	0.97
	Precision	1.00	1.00	0.91	1.00	0.91	0.96
	Recall	1.00	1.00	1.00	1.00	1.00	1.00
	F-Measure	1.00	1.00	0.95	1.00	0.95	0.98

Table 6.1: Model 1 statistics

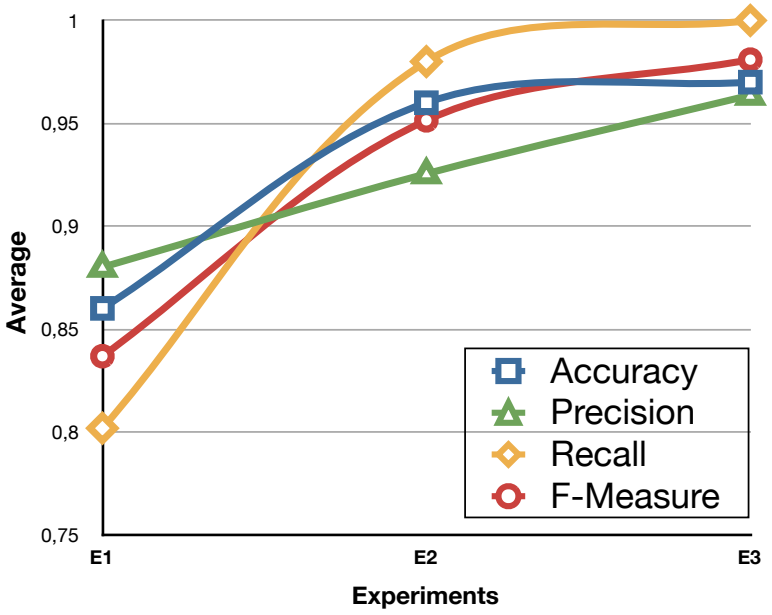

Figure 6.4: Model 1: results chart

6.5.2 Model 2

For the Model 2, the results for E1 were a little bit under 80%. Adding one more feature to the training model, E2 showed a big performance evolution, with an average near 95%. Making an update to the training model in E3, three of the four evaluation measures decreased but the Recall measure actually increased to 100%.

		S1	S2	S3	S4	S5	Average
E1	Accuracy	0.75	0.50	0.60	0.95	0.95	0.75
	Precision	0.86	0.50	0.56	1.00	1.00	0.78
	Recall	0.60	0.60	0.90	0.90	0.90	0.78
	F-Measure	0.71	0.55	0.69	0.95	0.95	0.77
E2	Accuracy	0.95	1.00	0.90	0.95	0.95	0.95
	Precision	0.91	1.00	0.83	1.00	0.91	0.93
	Recall	1.00	1.00	1.00	0.90	1.00	0.98
	F-Measure	0.95	1.00	0.91	0.95	0.95	0.95
E3	Accuracy	0.85	0.95	0.90	1.00	0.95	0.93
	Precision	0.77	0.91	0.83	1.00	0.91	0.88
	Recall	1.00	1.00	1.00	1.00	1.00	1.00
	F-Measure	0.87	0.95	0.91	1.00	0.95	0.94

Table 6.2: Model 2 statistics

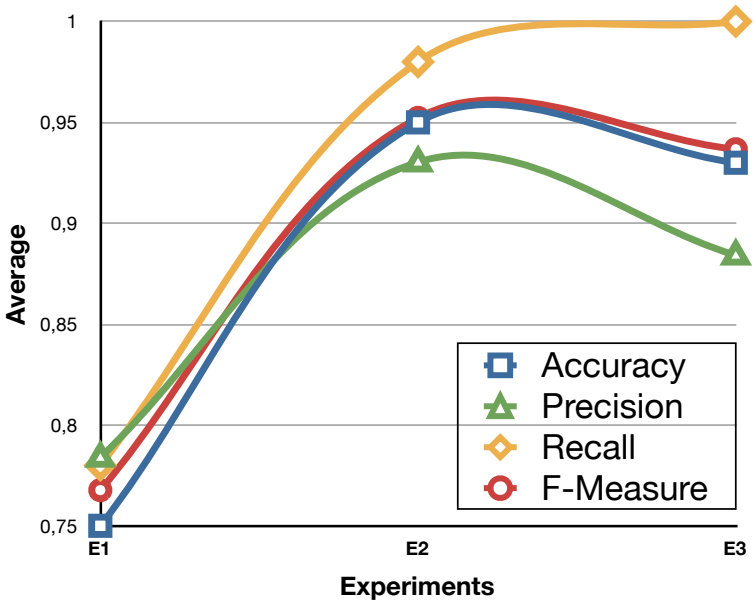

Figure 6.5: Model 2: results chart

6.6 Discussion

We assessed the performance of the proposed K-fold Cross Validation methods on both training models. From figures 6.4 and 6.5 it is clear that with adding more features and instances, the prediction model improves its performance.

For the experiment 1 (E1) with only two features and 80 instances for creating the training

model, in both models the initial average is nearly 80%. We can say that the prediction performance is very effective for such few inputs. In E2, there's a big evolution in both models with adding just one more feature. This proves that the features we selected are very effective for predictions. For E3, the results for Model 2 were no much different from the results of Model 1, but with a curious difference: for the Model 2, the results curiously decreased in *accuracy*, *precision* and *F-mesure*. Actually as the user provides more inputs, the accuracy of the resulting predictions should improve and that actually happened in Model 1. But we can say that Model 2 is not proving this wrong, because, besides the updates give more examples, this can give more sparse results, i.e, if the prediction results are near 50%, with more information, the prediction results can lean to both labels, "like" or "dislike".

The results are promising and positively support that this Google Prediction API is indeed effective for making predictions.

Besides the great results that we obtained using Google Prediciton API with Facebook, there are some limitations that need to be described.

As Facebook already uses a filtering system, there are many information that is hidden from the user's News Feed, i.e., if the user don't interact with a friend for a long time or if he actually had hidden all posts from a friend, this information cannot be analysed and therefore be filtered as relevant or not. As we described in our research, it is possible that some information can be relevant for a user besides his close relationship with other friend. That is why the combination of our selected features (Post title, owner and type) are so important for the relevancy of a post.

Another limitation from Facebook is that this Social Network is basically a data base of "Like" history, i.e., the information that we can get from the user is only the "positive" preferences. So, this lack of "dislike" history makes this training model to suffer from "cold start", but as the user gives more updates, the predictions will evolve and become more effective.

From Google Prediction API there is also a limitation: it is needed more than 100 update samples for the Prediciton API actually updates the training model.

6.7 Summary

This chapter presented a full description of the Prediction Model, i.e., how it was built and which labels and features were selected to make this prediction model more effective. To confirm that we have chosen the best post features, we also made a survey to Facebook users where we asked which features influence them to read, comment or put a "Like" in a post.

It was also described the different experiments that were made to the Facebook Filter Extension. To do this, we have used real data that was extracted from Facebook users. The results from these experiments were very positive and provided strong support for the great Google Prediction's efficiency in predictions. With the user interaction and consequently more update samples, it is expected that it might considerably increase the efficiency of the Facebook Filter Extension.

Besides the good results, there are some limitations that do not make this application perfect. Due to the Facebook's filtering system, this hides many posts that could be analysed by the Facebook Filter extension and which could be relevant for the user.

Chapter 7

Conclusion and Future Work

7.1 Work Summary

Social networking sites like Facebook have an abundance of information that is constantly growing in users News Feed. Facebook also thought of a filtering method so it created its own filtering system, which is unnoticed to the common user, but this filtering system exists. This filter gives to much weight to people the user interacts the most and it uses that to show the majority of information only from those people. But this filtering method reveals much information that is irrelevant for the users.

So, the goal of this dissertation was to create a filtering model that would analyse all this information and filter it as relevant or irrelevant for the user. Our research focused on developing an effective filtering system, capable of finding the real user preferences. In order to achieve the proposed goals, a study of many filtering systems were made to better decide which one would suit better for this project.

The result was the Facebook Filter Extension, a web-service that used four APIs: Facebook API, Google Prediction API, Google Cloud Storage and Google Chrome Extensions. We decided to use Google Prediction because it is a new technology created by Google and which experiments showed us great results. To prove its effectiveness, we made some experiments using some features and also adding input samples for the training model. As mentioned before, our experimental evaluation assessed the performance of the proposed K-fold Cross Validation methods using training inputs from real Facebook users. The results indicated that those *prediction models* are very effective in filtering posts according to the user preferences. The results indicated that the prediction efficiency actually improve when adding more features.

Additionally, one of the main challenges was to give the lowest user effort and design a friendly user interface. As Facebook already has a simple form of interaction, if we wanted the user to use our application and also improve its training model, the user's input would be simple and intuitive. We created a "dislike" button next to the default "like" button, so this two buttons would both "work" to give the users inputs to update their training models.

Having the training model complete, when the user accessed the News Feed's posts, he hould see them with the corresponding colors, "green" for relevant and "red" for irrelevant. In the case of a post being wrongly labelled as relevant or irrelevant, the user could indicate the opposite, i.e., he would indicate that post his preference by clicking the "like" or "dislike" buttons, giving this post information as an input update for his training model.

Therefore, the results from the Facebook Filter Extension proves that, with the user preferences and his interaction to improve the filtering model, it can help the user to show only the relevant posts as long he keeps improving its training model.

7.2 Future Work

Despite the main objectives have been implemented in the functional prototype, there are still further steps that could be improved in future work to have a more mature application:

- Improve the user interface: the current interface only shows the colors to the corresponding prediction result, i.e., greed for "like" and red for "dislike".
- Some features are not supported in prediction queries like the number of times that our friends have shared a post. So it would interesting that these features would be integrated as a major factor.
- There are some application limitations. The current version only lets users use Facebook Filter Extension with full permissions to the API, i.e., it is necessary to add the GMAIL address to the API Team to have complete access to Google Prediction and Google Could Storage.
- As Facebook only has "likes" from the user, this makes the training model suffer from "cold start" effect. So, to have a more accurate prediction in the classifier construction phase, it would be better to only create the filtering model after some "dislike" samples to have balanced samples of both labels. This way, the initial predictions would be more accurate.

Referências

[BCCO08] A. Bellogín, I. Cantador, P. Castells, and A. Ortigosa. Discovering Relevant Preferences in a Personalised Recommender System using Machine Learning Techniques. In *Preference Learning Workshop (PL 2008), at the 8th European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2008)*, pages 82–96, September 2008.

[BG04] Yoshua Bengio and Yves Grandvalet. No unbiased estimator of the variance of k-fold cross-validation. *J. Mach. Learn. Res.*, 5:1089–1105, December 2004.

[BHC98] Chumki Basu, Haym Hirsh, and William Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *In Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 714–720. AAAI Press, 1998.

[BJDA11] Johannes Behr, Yvonne Jung, Timm Drevensek, and Andreas Aderhold. Dynamic and interactive aspects of x3dom. In *Proceedings of the 16th International Conference on 3D Web Technology*, Web3D '11, pages 81–87, New York, NY, USA, 2011. ACM.

[BS97] Marko Balabanovic and Yoav Shoham. Fab: Content-based, collaborative recommendation. *Communications of the ACM*, 40:66–72, 1997.

[BYRN08] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Publishing Company, USA, 2nd edition, 2008.

[CNN+10] Jilin Chen, Rowan Nairn, Les Nelson, Michael Bernstein, and Ed Chi. Short and tweet: experiments on recommending content from information streams. In *Proceedings of the 28th international conference on Human factors in computing systems*, CHI '10, pages 1185–1194, New York, NY, USA, 2010. ACM.

[Coe08] Fabrício Luis Coelho. Classificação semi-automática de monografias, 2008.

[CYM08] Ming-wei Chang, Wen-tau Yih, and Christopher Meek. Partitioned logistic regression for spam filtering. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 97–105, New York, NY, USA, 2008. ACM.

[DPDM09] Toon De Pessemier, Tom Deryckere, and Luc Martens. Context aware recommendations for user-generated content on a social network site. In *Proceedings of the seventh european conference on European interactive television conference*, EuroITV '09, pages 133–136, New York, NY, USA, 2009. ACM.

[dSB10] Vagner Figuerêdo de Santana and M. Cecília C. Baranauskas. Summarizing observational client-side data to reveal web usage patterns. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, pages 1219–1223, New York, NY, USA, 2010. ACM.

[EHL11] Ed E. Hammer-Lahav. Oauth v2. http://tools.ietf.org/html/draft-ietf-oauth-v2-22, 2011.

[Fac11] Facebook. Facebook authentication. https://developers.facebook.com/docs/authentication/, 2011.

[FGC+97] Armando Fox, Steven D. Gribble, Yatin Chawathe, Anthony S. Polito, Andrew Huang, Benjamin Ling, and Eric A. Brewer. Orthogonal extensions to the www user interface using client-side technologies. In *Proceedings of the 10th annual ACM symposium on User interface software and technology*, UIST '97, pages 83–84, New York, NY, USA, 1997. ACM.

[FL10] Lujun Fang and Kristen LeFevre. Privacy wizards for social networking sites. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 351–360, New York, NY, USA, 2010. ACM.

[Goo11a] Google. Google oauth2. http://code.google.com/intl/pt-PT/apis/accounts/docs/OAuth2.html, 2011.

[Goo11b] Google. oauth2-chrome-extensions. http://smus.com/oauth2-chrome-extensions, 2011.

[Goo11c] Google. Prediction black box. http://www.technologyreview.com/computing/26093/, 2011.

[Joa98] Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pages 137–142, Heidelberg et al., 1998. Springer.

[Joa01] Thorsten Joachims. A statistical learning learning model of text classification for support vector machines. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '01, pages 128–136, New York, NY, USA, 2001. ACM.

[KB06] Arnd Christian König and Eric Brill. Reducing the human overhead in text categorization. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 598–603, New York, NY, USA, 2006. ACM.

[KLR+10] Janne Kätevä, Perttu Laurinen, Taneli Rautio, Jaakko Suutala, Lauri Tuovinen, and Juha Röning. Se-155 dbsa: a device-based software architecture for data mining. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, pages 2273–2280, New York, NY, USA, 2010. ACM.

[Par11] E. Pariser. *The Filter Bubble: What the Internet Is Hiding from You*. Penguin Group USA, 2011.

[PNH08] Xuan-Hieu Phan, Le-Minh Nguyen, and Susumu Horiguchi. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *Proceeding of the 17th international conference on World Wide Web*, WWW '08, pages 91–100, New York, NY, USA, 2008. ACM.

[rag09] raghava. Evaluation of bioinformatics methods. http://www.imtech.res.in/raghava/gpsr/Evaluation_Bioinformatics_Methods.htm, 2009.

[RBSC] Sistemas De Recomendação, Eliseo Berni, Reategui Sílvio, and César Cazella. Chapter 1.

[Sas02] Manabu Sassano. An empirical study of active learning with support vector machines for japanese word segmentation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 505–512, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[Set09] Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[SFD+10] Bharath Sriram, Dave Fuhry, Engin Demir, Hakan Ferhatosmanoglu, and Murat Demirbas. Short text classification in twitter to improve information filtering. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 841–842, New York, NY, USA, 2010. ACM.

[SK10] Kazunari Sugiyama and Min-Yen Kan. Scholarly paper recommendation via user's recent research interests. In *Proceedings of the 10th annual joint conference on Digital libraries*, JCDL '10, pages 29–38, New York, NY, USA, 2010. ACM.

[SR06] John Stamey and Trent Richardson. Middleware development with ajax. *J. Comput. Small Coll.*, 22:281–287, December 2006.

[TK02] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *J. Mach. Learn. Res.*, 2:45–66, March 2002.

[ZL06] Dell Zhang and Wee Sun Lee. Extracting key-substring-group features for text classification. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 474–483, New York, NY, USA, 2006. ACM.