ALLAN DIEGO SILVA LIMA

S.O.R.M.: SOCIAL OPINION RELEVANCE MODEL

São Paulo
2015

ALLAN DIEGO SILVA LIMA

S.O.R.M.: SOCIAL OPINION RELEVANCE MODEL

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Doutor em Ciências

Área de Concentração: Engenharia de Computação

Orientador: Prof. Dr. Jaime Simão Sichman

São Paulo
2015

ALLAN DIEGO SILVA LIMA

S.O.R.M.: SOCIAL OPINION RELEVANCE MODEL

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Doutor em Ciências

Área de Concentração: Engenharia de Computação

Orientador: Prof. Dr. Jaime Simão Sichman

São Paulo
2015

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, 23 de julho de 2015.

Assinatura do autor Allan Diego Silva Lima

Assinatura do orientador

FICHA CATALOGRÁFICA

Lima, Allan Diego Silva
S.O.R.M.: Social Opinion Relevance Model / A.D.S. Lima. -- ed.rev. -- São Paulo, 2015.
163 p.

Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Recuperação da informação 2.Mineração de dados 3.Aprendizado computacional I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

Acknowledgements

To Professor Jaime Simão Sichman, for the tuition and constant incentive across four years of work.

To Milena Monteiro, for always supporting me during since the beginning of my project.

To Professor Lucia Filgueiras for all the time spent helping to improve our research methodology and preparing to my oral defense.

To Professors Anna Reali, Anarosa Brandão and Valdinei da Silva for important contributions to this project.

To my friends Gustavo, Felipe Leno, Nicolau Werneck and Allan Borim, not only for the many constructive discussions and suggestions to improve my work, but for the help in managing bureaucratic subjects as well.

And to all the people that collaborated directly or indirectly with the development of this work.

Exploration is in our nature. We began as wanderers, and we are wanderers still.

Carl Sagan

Resumo

Esta tese apresenta um modelo de relevância de opinião genérico e independente de domínio para usuários de Redes Sociais. O Social Opinion Relevance Model (SORM) é capaz de estimar a relevância de uma opinião com base em doze parâmetros distintos. Comparado com outros modelos, a principal característica que distingue o SORM é a sua capacidade para fornecer resultados personalizados de relevância de uma opinião, de acordo com o perfil da pessoa para a qual ela está sendo estimada. Devido à falta de corpus de relevância de opiniões capazes de testar corretamente o SORM, fez-se necessária a criação de um novo corpus chamado Social Opinion Relevance Corpus (SORC). Usando o SORC, foram realizados experimentos no domínio de jogos eletrônicos que ilustram a importância da personalização da relevância para alcançar melhores resultados, baseados em métricas típicas de Recuperação de Informação. Também foi realizado um teste de significância estatística que reforça e confirma as vantagens que o SORM oferece.

Palavras-chave: relevância da opinião; mineração de opinião; recuperação de informação; pesquisa social.

Abstract

This thesis presents a generic and domain independent opinion relevance model for Social Network users. The Social Opinion Relevance Model (SORM) is able to estimate an opinion's relevance based on twelve different parameters. Compared to other models, SORM's main distinction is its ability to provide customized results, according to whom the opinion relevance is being estimated for. Due to the lack of opinion relevance corpora that are able to properly test our model, we have created a new one called Social Opinion Relevance Corpus (SORC). Using SORC, we carried out some experiments on the Electronic Games domain that illustrate the importance of customizing opinion relevance in order to achieve better results, based on typical Information Retrieval metrics, such as NDCG, QMeasure and MAP. We also performed a statistical significance test that reinforces and corroborates the advantages that SORM offers.

Keywords: opinion relevance; opinion mining; information retrieval; social search.

Index of Illustrations

Index of Tables

Abbreviations and Acronyms

GA	Genetic Algorithms
IDF	Inverse Document Frequency
IR	Information Retrieval
MAP	Means Average Precision
MAS	Multi Agent Systems
NDCG	Normalized Discounted Cumulative Gain
OM	Opinion Mining
SORC	Social Opinion Relevance Corpus
SORC-G01	Social Opinion Relevance Corpus – Games 01
SORM	Social Opinion Relevance Model
SVM	Support Vector Machines

Summary

1 Introduction

When someone wants to find opinions about a subject (product, theme, person, etc.) how can he/she have access to third party opinions? Before the Internet age, he could ask his parents, relatives, friends, etc. Furthermore, other sources like television, magazines and newspapers were frequently consulted. However, due to the popularization of the Internet, nowadays it is possible to search for opinions about basically every subject. Following the growth of the Internet, tools like blogs, collaborative sites (like Wikis) and message boards (forums) have become more popular. Consequently, the number of people expressing their opinions in the Internet has grown significantly. Hence, if someone wants to search opinions about a given subject using general purpose search engines, like Google, the obtained results are lists of documents, where only part of their content is composed of opinions.

As an example, how someone interested in an electronic game called “Just Dance” can find opinions about it? One of the most common ways is to access a general purpose search engine (like Google or Bing) and search for “just dance review”. This query will return to the user around 6 million[1] pages and probably each of them containing many opinions. Another fact that is impossible to ignore is that, for many reasons, only some of the opinions in the documents are aligned with the user’s profile. In other words, just some of the opinions in the documents can be *relevant* for a specific user.

At this point we can see two different problems: (1) how to provide only opinions to the user, i.e. identify and retrieve only the opinions in documents, avoiding the user to waste his time reading all the document to identify them; (2) given a user, how to find out what makes an opinion relevant to him. The first problem has been vastly discussed in the literature as a text classification problem and it is one of the most important pillars of Opinion Mining (Liu B. , Opinion Mining, 2009), a recent research field that studies how to mine opinions, as we will see

[1] https://www.google.com.br/search?q=just+dance+review.

further in this thesis. However, the second problem has been rarely discussed, since most of the models that consider the existence of opinions in documents are not meant to estimate the relevance of an opinion, but to present documents that contain them. Consequently, those models do not make an in-deep study on what makes opinions relevant to someone. So, when it comes to this topic, usually those models have only a superficial approach, especially regarding the customization of the results.

In this work, we are interested in exploiting the second problem described. In the next section, will define our goal more precisely and discuss how related projects do not solve it properly. Thus in section 1.2 we describe our solution; in section 1.3 we address our research methodology; in section 1.4 we discuss the applications and limitations of our proposed solution; finally, in the last section we describe the content of the next chapters of this thesis.

1.1 Problem Definition and Limitations of Existing Work

As discussed earlier, our research project aims to model what makes an opinion relevant for a given user. Therefore, our model has to address two issues: (1) how to formally define the concepts that make an opinion significant for someone; (2) how to respect the differences between users that have distinct values while considering an opinion as relevant or not. Based in these issues, it is possible to define our main research question as:

Research Question – How to estimate the relevance of an opinion for a user with respect to his preferences?

Currently, **there is no model to estimate the relevance of an opinion that is based on an in-deep investigation of parameters that need to be taken into account to define this relevance.** In our literature review, we were able to find projects that have models to estimate the relevance of documents through a conjunction of parameters, including among them, an estimation of the relevance of the opinions on the documents (Zhang & Ye, 2008; Mishne, 2006; Attardi & Simi, 2006; Huang & Croft, 2009). But since those projects work with documents and not with opinions, their approaches to evaluate the importance of an opinion are relatively simple, commonly exploring only the text of the opinions. Following a similar

trend, recent projects (Orimaye, Alhashmi, & Siew, 2012a; Orimaye, Alhashmi, & Siew, 2012b; Xu, et al., 2012) started to explore the linguistic aspects of opinions while estimating their relevance. They proposed different ways to analyze the text of an opinion, thus refining and improving the main heuristics that guided the projects previously mentioned: “the more opinions a document has, the more relevant it is”. The project that was most aligned to our objectives was the one described in (Luo, Osborne, & Wang, 2013), but still it has important limitations such as the lack of customization on the results, which brings us to the next point of our problem definition. **This literature review also revealed that absence of domain independent modes to estimate the relevance of an opinion.** The models we could find on the literature were conceived to a specific domain or social network. Therefore, it is expected that they could not work properly on domains others than the ones they were originally meant for.

Opinions are not facts, then it is expected that an opinion can be relevant for a user but completely irrelevant to another; however, current models do not model this possibility. When a user is searching at the Internet for an answer to a question like, for example, “What it the currency conversion from dollars to euros?” he will find documents containing a fact (in this case a value) that answers his information need, whose relevance does not depends on the person who is searching for it. However, if he is looking for an answer to a question like, for example, “Does the Beatles song Eleanor Rigby have beautiful lyrics?” he is going to find sentences in documents representing opinions from different points of view on what makes a song’s lyrics beautiful or not. While he can agree with a subset of those opinions, another user can agree with a completely disjoint subset. The conclusion is that there is no consensus on what makes an opinion relevant: if it is relevant for a person it can be completely irrelevant for another. Current projects do not explore this issue properly, not allowing a different relevance value for the same opinion according to the person that made the search.

1.2 Proposed Solution and Contributions Overview

As a solution to our research question described in the previous section, we introduce the intuitive notion of a *model to estimate the relevance of an opinion with respect to the profile of the user who is searching for it*. To achieve this objective, we

have investigated the following issues: (1) what are the ways to estimate the opinion's relevance, respecting the user's profile? And (2) how is it possible to validate and refine the model through experiments?

Our approach to the first question was to search the literature and adapt document relevance techniques to estimate opinions' relevance. In other words, we need to find parameters to evaluate the relevance of an opinion, and gather them together in a pool of parameters that can be combined to form a suitable ranking function for opinions. Each parameter is a metric that can be computed using the information about the user, the opinion itself and its author.

This effort to define the parameters, however, would be pointless without a methodology to evaluate them. So, we also decided to research and adapt to our context some evaluation methods normally used by Information Retrieval systems.

Given that, we made an extensive literature review and adapted some techniques from the following research fields, and in particular from the corresponding works mentioned below:

- **Information Retrieval** (Manning, Raghavan, & Schütze, 2008)**:** this domain provides metrics to estimate the relevance of a document; inspired by them, we were able to define some of the parameters to our model. This research field also offered methodologies to test and refine our model, based on the metrics that this field produced to evaluate the results of Information Retrieval systems;

- **Opinion Mining** (Liu, 2012)**:** it offers definitions of what is an opinion, so we could translate it to a formal representation which became the opinion's definition present in our model. Moreover, this research field also provided all the support necessary to collect texts across the Internet and filter its content to create an opinion database for the evaluation of our model;

- **Social Search** (Morris, Teevan, & Panovich, 2010b)**:** it is a research field related to Information Retrieval, where we could find the answer to the problem of how to estimate the relevance of a document, while respecting the user profile. Specifically with the concepts of this research field, we were able to associate the user preferences and his social network to provide a personalized way to find the relevance of an opinion for every single user;

- **Multiagent Systems** (Wooldridge, 2009) in this domain, reputation models provide us the ability to model authors' reputation not only from a single user point of view, but from the point of view of a set of users as well. This is necessary in our model to concretize the estimation of the relevance of an opinion's author, which is also linked to Social Search concepts.

- **Evolutionary Algorithms** (Bäck & Schwefel, 1993)**:** they provided us means to instantiate an abstract opinion relevance model into a specific domain. This is archived by balancing and customizing the parameters of an opinion relevance model to a specific domain. Evolutionary Algorithms are domain and problem independent; two features that can be useful to create initial baseline benchmarks, especially when there are no in-depth studies on how to balance a social opinion relevance model in the literature.

Hence, given the multi-disciplinary character of our research project, during its development we were able to make the following contributions:

Main contribution: A formal and domain independent model to estimate the relevance of an opinion for a user, which is able to provide personalized results according to the user profile.

As a side effect of our work on our main contribution, we have also managed to make two other important contributions.

Other contributions:

1) A methodology, inspired by Information Retrieval methods, to evaluate models for opinion's relevance estimation;

2) A corpus created for the evaluation of our opinion relevance model. In other words, a dataset containing not only opinions, but also information associating users with the opinions that they consider relevant or not. Without this type of database, it would be hard to apply the previously cited methodology to evaluate an opinion relevance model.

3) Formal definitions to classify social and customizable aspects of an Opinion Relevance Model. They are fundamental to distinguish our project from the ones present on the literature.

At this point, it is possible to describe the main research hypothesis of our research:

Research Hypothesis – When estimating the relevance of an opinion to someone, it is important to customize the results based on his/her Social Network profile.

This hypothesis can be verified if we compare ways to estimate the relevance of opinions that are not customized with others that are. However, in order to do that, we first need to address many important steps, as seen next.

1.3 Methodology

In this section, we describe all the phases of our research methodology. First, we address our model to estimate the relevance of an opinion. Then we talk about how we have created our model and how we have created an Opinion Relevance Corpus in order to evaluate this model. Finally, we describe the steps involved in the evaluation of our model based on Information Retrieval Metrics and Statistical Tests.

1.3.1 Solution Overview

As discussed before, many opinion's aspects may influence its relevance; moreover, it is common for different people to have distinct views of the opinion relevance. Hence, it is important to customize the opinion relevance according to

whom it is being estimated for. We want to achieve this in a Social Opinion Relevance Model (SORM) by considering both the user's profile and the profile of the author of the opinion, as shown in Figure 1.1.

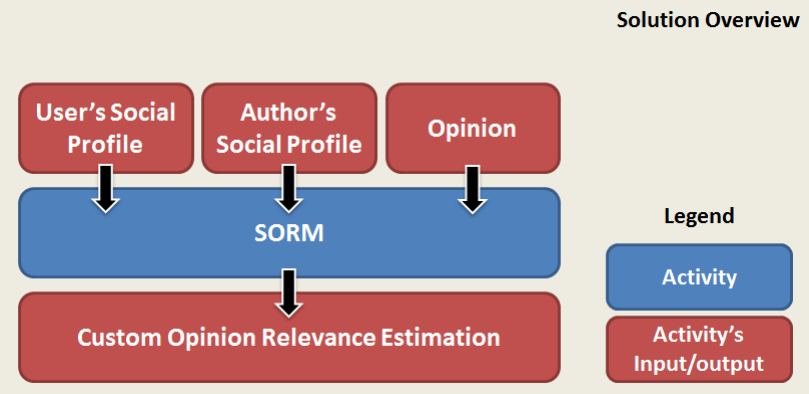


Figure 1.1 – Overview of our Methodology

1.3.2 Model Creation

In order to make the contributions described in the previous section we have worked in three main tasks: (1) the definition of a pool of parameters to compose an opinion ranking function; (2) the creation of an Opinion Relevance Corpus for testing our model; and (3) the execution of tests to compare the effect of social parameters in opinion mining strategies. The first two tasks were developed in parallel; however, in order to perform the last one we needed the other two to be finished before it started.

The parameters of our model were developed through many interactions of a simple process. First of all, we performed a literature review on different research fields (as seen in Figure 1.2), focusing whether we could use or adapt their concepts in order to make part of our pool of parameters. For example, with (Srinivasan, Teitelbaum, & Wu, 2006), we could identify their agent reputation models as a promising way to help in the estimation of opinion relevance, so we decided to adapt and include it in our pool of parameters.

Another important result of our literature review was the elicitation of ways to combine and weight the parameters of our model. Both are important because they allow the model to be instantiated in a specific domain. For instance, a specific parameter of our model may have a high importance to evaluate the relevance of an opinion on a given domain. However, the same parameter may have a smaller

importance (compared to the previous domain) on another domain. This phenomenon is modeled in our solution by allowing distinct ways to weight and combine the parameters of our model.

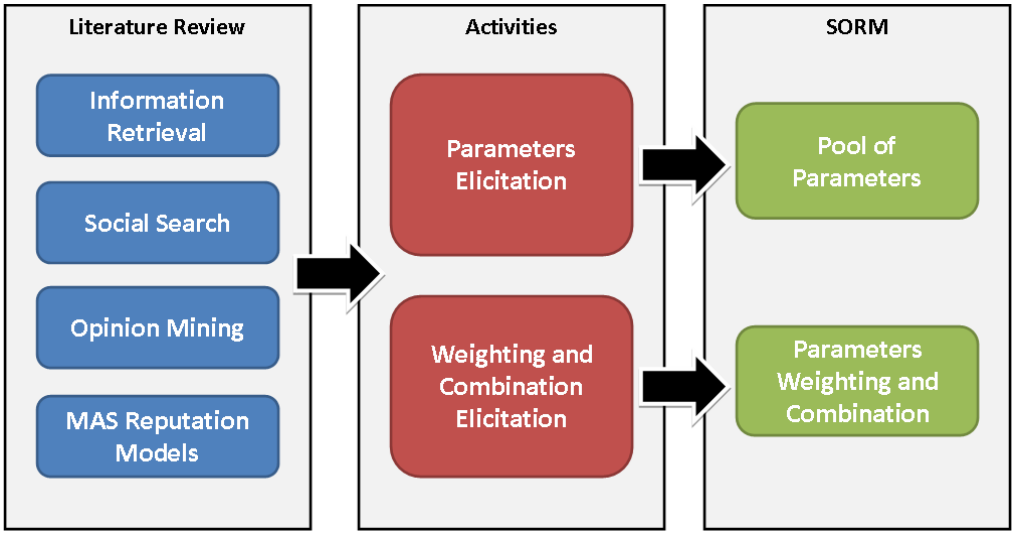


Figure 1.2 – Creating a Social Opinion Relevance Model.

1.3.3 Model Evaluation

In order to be able to test our model, we developed a methodology to create what we call a Social Opinion Relevance Corpus and use it to evaluate our model. This methodology is composed of three phases: (1) creation of a Social Opinion Relevance Corpus, (2) experiments using the corpus created previously to compare or solution with distinct approaches and test our research question; (3) comparison of different opinion relevance models, by conducting some statistical tests. Each is these phases is described as follows.

Creation of a Social Opinion Corpus

A Social Opinion Relevance Corpus can be created from web pages using an opinion classifier able to find the opinions in the pages. During this phase, first we convert Web Pages (non-structured data) into a Text Corpus (structured data), and then we extract reviews, sentences and opinions from the text corpus, as we can see in Figure 1.3.

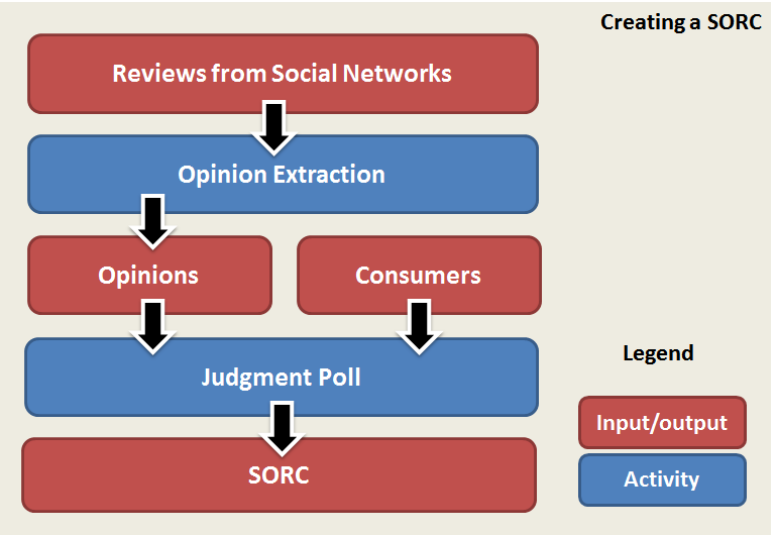


Figure 1.3 – Creating an Opinion Relevance Corpus from Social Network Web Pages.

Opinion Extraction

This first activity of this phase starts extracting reviews from one or more Social Networks, in order to compose a collection of Reviews. During this activity, non-structured HTML pages are converted in structured data with all its meta-data extracted. For instance, once created, a Review may contain information such as its author, its published date and its rate properly structured. By converting non-structured data in structured data, this step of our methodology prepares the data to be processed in the next step.

Once the reviews are extracted, we want split the reviews text into Text Fragments. A Text Fragment is text fragment is a sub-sequence of the Review text: for instance, it can be a sentence or a paragraph depending on the domain or the objectives of the project researchers. The output of this step is a set of Sentences that are used as input in next step.

Now we can finally find out which of the input Text Fragments are opinions. To achieve this, we used an opinion classifier. Such classifier was created by manually labeling samples of the Text Fragments in the set of Reviews created in the previously, and using them to train and test the classifier. After using this classifier, we could identify which of the input Text Fragments are opinions, and finally reach our objective in this activity: to build an Opinion Corpus.

Judgment Poll

In this activity, we want to associate relevance information to some of the Opinions in the set created in the previous phase. We can accomplish this by conducting polls with users.

A poll where the user (a volunteer) has to evaluate the relevance of opinions about a subject he has chosen previously. Each relevance evaluation made by a user represents a Relevance Judgment. A Relevance Judgment can be intuitively seen as a relevance value that was assigned by a user to an opinion about a certain subject. The result of this process is what we call a Social Opinion Relevance Corpus.

1.3.3.1 Evaluation

In this phase of our methodology, we have an important objective: we want to test different Opinion Relevance Models, i.e., models with different parameter values, for instance those including or not a Social parameter. In order to evaluate their performance, we use Information Retrieval Metrics, applying the models to the previously created Social Opinion Relevance Corpus. This process is illustrated in Figure 1.4.

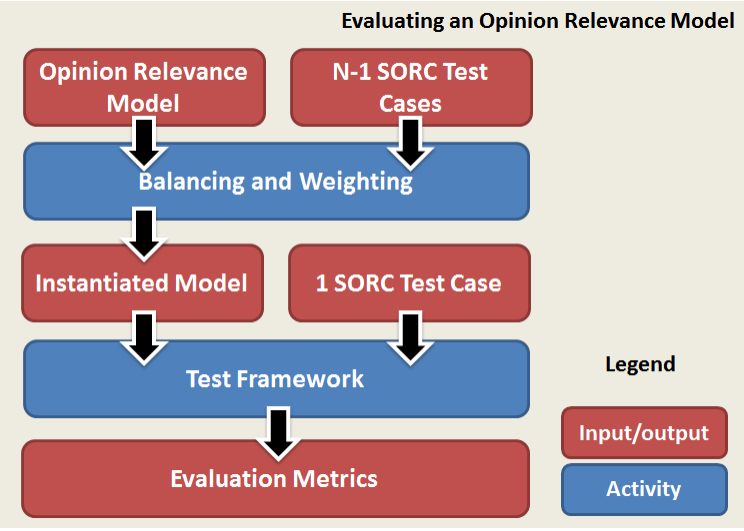


Figure 1.4 – Testing an Opinion Relevance Model.

Balancing and Weighting

This activity is the main responsible for instantiating a domain independent opinion relevance model to a specific domain. This is achieved by find how important each parameter of the model is for the domain. We call this process of weighting. The

responsibility of the activity is to find how to combine the multiples parameters of a model. We call this process Weighting.

Both weighting and balancing can be performed using machine leaning and cross-validation. We use as input N-1 test cases (were N is number of test cases of a SORC) to learn how to balance and weight a model form a domain. Then we use instantiated model and the remaining test case as the input the next activity: Test Framework.

Test Framework

In this activity, we tested distinct Opinion Relevance Models. Initially, we carefully chose the Information Retrieval metrics to evaluate the models. The metrics must be chosen based on the characteristics of the Opinions Relevance Corpus (which in one of the parameters of this phase) and the objectives of the comparison. For instance, we may want to find which model is better regarding the coverage of the relevant opinions. Therefore, we must choose an Information Retrieval metric that is able to give better values for models with higher coverage. The outputs of this step are the numeric values of the Information Retrieval metrics for each Opinion Relevance Model evaluated.

1.3.3.2 Significance Test

In this phase, we want to compare the significance of the results of two different Opinion Relevance Models that were tested on the previous phase. With this result, we want to complement the initial tests so the conclusion about which model is better can be based on statistical methods. This process is illustrated in Figure 1.5.

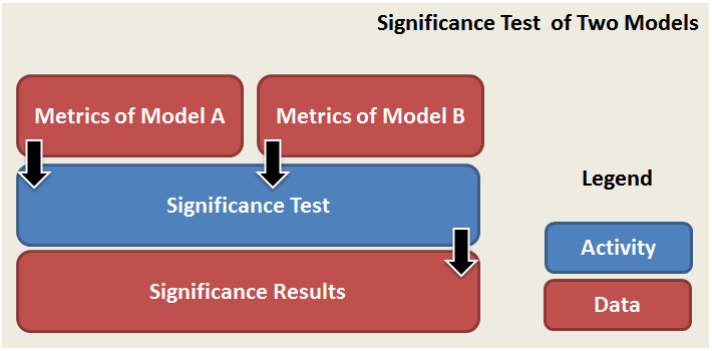


Figure 1.5 – Performing a significance test with two different Opinion Relevance Models.

Significance Test

In this step of our methodology, we want to be able to estimate the significance of the comparison of the results we computed in the previous step. To do that we need to choose a significance test, like for instance the Randomization Test (Fisher, 1935; Smucker, Allan, & Carterette, 2007). Thus we will be able to have a statistical background when we say that a system is better or not compared to other based in our tests. This is important especially because the Opinion Relevance Corpus used in the first step of this phase may not have many opinions or its relevance judgments may not cover all the opinions on the Text Corpus used to create it. However, even with those limitations we can still be able the have some significant and conclusive results, mainly due to this last step of our Evaluation Methodology. By conducting these significance tests, we can estimate the performance of the opinion relevance models using the inputs, and thus compare their results to evaluate which of them is best suited for the corpus.

1.4 Outline

The remaining of this thesis is structured as follows. The second chapter describes basic concepts that are needed to comprehend the techniques and strategies we chose to solve our problem. Chapter three provides an overview of the Opinion Mining research field, focusing on the concept of opinion relevance. Chapter four introduces our formal model, discussing how to compute its components and how to balance them in order to create a social and customizable Opinion Ranking function. Chapter five introduces the concept of a Social Opinion Relevance Corpus and shows how to use such a corpus to test our model. Chapter six explains the design of our experiments and the results we found by performing them. Finally, the last chapter presents our conclusions, the applications of our work to real-life problems, the scope limitation of our project, the challenges that we had to overtake during its development and how we intend to improve our research project.

2 Basic Concepts

In this chapter, we will discuss some key concepts that are necessary to understand our work. Therefore, here we will address research fields including their definitions, methodologies and techniques needed to comprehend the content of the next chapters.

We will start this chapter with an overview on Information Retrieval (Manning, Raghavan, & Schütze, 2008), focusing on Document Relevance Models and their evaluation. In section 2.2 we will discuss Social Search (Evans & Chi, 2010; Morris, Teevan, & Panovich, 2010b) , a relatively new research field that has its concepts used in many Internet services. In section 2.3, we will discuss Text Classification (Joachims, 1998) , by making an overview of its main strategies. In section 2.4, we will address Multi-Agent Reputation models, describing some of their distinct approaches. In section 2.5 we will review the concept of Evolutionary Algorithms, focusing on its most known strategy: Genetic Algorithms. Finally, in section 2.6 we discuss how each of the concepts addressed in this chapter has its applications on our work.

2.1 Information Retrieval

Information Retrieval (IR) can be defined as:

"Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers)" (Manning, Raghavan, & Schütze, 2008).

In order to understand this definition, we must first detail the differences between structured and non-structured data. A dataset is called structured when its information is explicitly delimited by fields containing a semantic value. A computer algorithm easily interprets such fields. Good examples of structured data are the rows of a table in a relational database. On the other hand, a dataset is considered as non-structured when there is no distinction among its different parts, which makes it hard to be interpreted by an algorithm. For example, a text in an Internet blog is considered a non-structured data.

Another important feature of the IR definition is related to the amount of data that is handled. When one wants to work with large volumes of data, it is necessary to promote many changes on the traditional algorithms and data structures. It is important to meet the requirements and constraints related to the time that an IR system has to match while retrieving data from a dataset. Hence, given a large amount of data and time constraints, traditional techniques and strategies usually cannot be used in IR problems: we have to adapt and modify them before we start to work on an IR problem.

Information Retrieval is a vast research field with many sub-areas and problems. However to properly understand our work we do not need to introduce them all. In the next subsections, we will focus on the four most important aspects of IR when it comes to understand our research project: (1) Document Retrieval Models, (2) Document Scoring (Relevance) Models, (3) Link Analysis, and (4) Evaluation of IR Systems.

2.1.1 Document Retrieval Models

There are basically two different types of document retrieval models: the boolean model and the bag of words model (Manning, Raghavan, & Schütze, 2008). Even if both handle the document as set of words, they have important differences..

The boolean retrieval model allows the retrieval of information through queries represented as Boolean expressions of terms. Hence, a query is composed of terms combined by Boolean operators such as AND, OR and NOT. For instance, the query “ball AND blue” has the terms “blue” and “ball” united by the operator “AND”. Semantically, this sample means that the IR system should search for documents containing both terms independently of the order, or of the number of times that each word appears in the documents.

While still handling the documents a set of words, the bag of words model takes into consideration the number of times a word appears in a document. This number is considered an indication of the document relevance. Therefore the more times a word in the query happens in a document, the more relevant the document is. For instance, a query containing the words “ball” and “blue” has as result a list of

documents where the top documents are the ones where they both appear more times.

2.1.2 Document Scoring Models

Given that IR problems have to deal with a large volume of documents, the result of a query is usually a list that may contain millions of documents. Unfortunately, such volume exceeds a human-being reading capability. Hence, Information Retrieval offers several strategies to score and generate numerical values for documents according to a query. Therefore, when a user submits a query to an IR system, it will return a small list of the documents that are supposed to be more relevant to that query.

The boolean ranking model estimates the relevance of a document by ranking its zones. Each zone of a document contains (meta) data about it. For instance, if the document is a blog post, its zones could be its text, its title, its date, its author(s), etc. Therefore, for each of these parts a relevance score is computed according to a query. Hence, if one of the words in the query appears on the title of a blog post, its score of this zone will be higher than a post that does not have any word of the query on its title.

Formally, let $w_1, \ldots, w_n \in [0,1]$ so that $w_1 + \ldots + w_n = 1$ and $1 \leq i \leq n$; let s_i be the boolean score of the $i\text{-}th$ zone of the document d, with and $0 \leq s_i \leq 1$; hence equation 2.1 shows how to compute the document score on the boolean retrieval model. On the equation, each w_i acts as the normalized weight of the $i\text{-}th$ zone on the model and the value of each w_i can be learned from samples using Machine Learning techniques.

$$Score(d, q) = \sum_{i=1}^{n} w_i s_i(d, q) \tag{2.1}$$

As discussed before, in the bag of words model, the document score is estimated based on the number of occurrence of the terms on it. However, in practice, the frequency of the terms is just one of the parameters of this model. One of the most popular ways to score a document in this model is called tf-idf weighting (Jones, 1972). It is an example of a document score function that does not rely only

on the content of the document to be ranked. The tf-idf score function is composed of two concepts (1) tf: term frequency, and (2) idf: inverse document frequency.

The term frequency is computed based on the number of occurrences of a term t in a given document d (equation 2.2(a)). The inverse document frequency (equation 2.2(b)) is computed by the logarithm of the division of the total number of documents in the set N by the number of occurrences of the term t in all documents in the collection $df(t)$. Notice that the logarithm is used on the equation to increase the values of the rare terms. The same principle can be applied to the tf equation, therefore instead of using just the number of times a term happens on a document, we can use the logarithm this value.

$$tf(d,t) = \#\ of\ times\ t\ happens\ on\ d \quad \text{(a)}$$

$$idf(t) = log\left(\frac{N}{df(t)}\right) \quad \text{(b)} \qquad (2.2)$$

Now that tf and idf are both defined, we can formalize the tf-idf score of a term t for the document d (equation 2.3(a)) as the product of both concepts. Furthermore, we can also formalize the score of the document d for a query q in this model (equation 2.3(b)) as sum of the tf-idf of each term in the query q.

$$\text{tf-idf}(\text{d},\text{t}) = \text{tf}(\text{d},\text{t}) \times idf(t) \quad \text{(a)}$$

$$Score_{\text{tf-idf}}(\text{d},\text{q}) = \sum_{t \in q} \text{tf–idf}(\text{d},\text{t}) \quad \text{(b)} \qquad (2.3)$$

2.1.3 Link Analysis

While the document scoring models discussed previously are able to rank documents based on their content and a given query, information retrieval also offers others models to estimate a document's relevance. In this context, the analysis of the links between documents is a solution to complement ranking models that are also based on the scoring models. Links between documents can have many distinct semantic values. For instance, in academic papers links are references to other papers; on web pages, links are references to others web pages (hyperlinks). Here

we will discuss two distinct types of strategies to analyze links: Hubs/Authorities (Kleinberg, 1999), and the PageRank (Brin & Page, 1998).

On the Hub/Authority strategy, each page has two different scores. One is called hub score and the other is called authority score. Intuitively, a hub is a document with multiple links to other documents, while an authority is a document that is referenced by multiple other documents. Formally, let x and y be members of a set of documents D, let $h(x)$ and $a(x)$ be respectively the hub and the authority score of the document x, and let $link(x,y)$ be a function that returns $True$ if the document x has at least one link to the document y and $False$ otherwise. Equation 2.4 shows how both $h(x)$ and $a(x)$ are computed. It is possible to see the dependence between them concepts. Usually, to compute $h(x)$ and $a(x)$ it is necessary an interactive algorithm where $h(x)$ and $a(x)$ are initially set, and then the authority and hub values are computed.

$$h(x) \sim \sum_{y \in D \wedge link(x,y)} a(y) \quad \text{(a)}$$

$$a(x) \sim \sum_{y \in D \wedge link(y,x)} h(y) \quad \text{(b)} \qquad (2.4)$$

On the other hand, PageRank is a technique that handles documents as a directed graph where the edges represent the links between them. Each edge has a semantic value defined by an associated link text. To compute this score, we simulate a user surfing between documents randomly using links between them. Thus, the higher the chance of a page being randomly found by the surfing simulation, the higher its PageRank is. In practice, this means that the more hyperlinks exist for a given page, the better its score is. Another important aspect of this model is its recursive nature. Given a link h from a document x to a document y, h has a weight proportional to the PageRank of x, so to calculate the PageRank of y we must first find the PageRank of all other documents that have one or more links to y. Initially, to compute the PageRank of a set of documents D, let P be a square matrix where each element p_{xy} on P represents the probabilities of a user to randomly go from a document x to the document y while surfing. Then we must find the

principal left eigenvector V (the one with eigenvalue equals to 1) of P, which is a single row vector where each v_i element represents the PageRank of the document i-th on D. The PageRank can also be updated based on its recursive definition. Given a document d, and the set of documents D_d containing links to d, Equation 2.5 shows how the PageRank can be updated once the initial PageRank value is computed for each document.

$$PageRank(d) = \sum_{x \in D_d} \frac{PageRank(x)}{|D_d|} \quad (2.5)$$

2.1.4 Evaluation of Information Retrieval Systems

In this subsection, we discuss how Document Ranking Models in Information Retrieval are evaluated. Our focus is the description of how to create and use a Document Relevance Corpus to estimate the performance of a model, since we are interested in how we could test our own model. Initially, we will discuss what an evaluation corpus is; then we will describe some of the most popular document corpus available in the Internet. Furthermore, we will discuss how to create these type of corpus by conducting polls, what are the main metrics used to measure the performance a document retrieval model and how to know if the obtained results are significant or not. Finally, we will also make a brief discussion on alternative methods to corpus based information evaluation of document ranking models and information retrieval systems.

In information retrieval, document ranking models are usually evaluated by tests conducted using document corpora (we will also use the expressions “Information Retrieval Test Sets” and “Information Retrieval Relevance Corpus” in this thesis to refer to this concept). Those corpora are composed by (Manning, et al., 2008):

1) A document collection;
2) A test set of information needs, which are expressible as queries;
3) A set of relevance judgments, standardly a binary assessment of either relevant or non-relevant for each query-document pair;

We can define an Information Retrieval Test Set and its elements as:

Definition 2.1 – Document Collection: a set of documents. The documents can be crawled from the Internet or from other sources depending on the objectives of the corpus and on the model to be evaluated. For example, if the objective is the evaluation of a model for ranking cooking documents, it is natural that the documents in the collection are about cooking recipes.

Definition 2.2 – Information Need: an abstract representation of an information wish. For example, a user may want to know about recipes for chocolate cake.

Definition 2.3 – Query: a String of characters representing the translation of an Information Need. Since many Information Retrieval Systems receive a query as input, an Information Need must be translated to queries by the users. For example, if a user wants to know about recipes for chocolate cake he may translate this Information Need to queries like: “chocolate cake recipes” or “how to make chocolate cakes”.

Definition 2.4 – Relevance Judgment: a tuple (qu, do, re), where qu is a query, do is a document, and re is the document’s judgment that can assume different document relevance values for the purpose of the query.

Definition 2.5 – Information Retrieval Test Set (Information Retrieval Relevance Corpus): a triple (Do, IN, RE), where DO is a set of Documents, IN is a set of Information Needs and RE is a set of Relevance Judgments.

Due to the amount of documents and possible queries, it is almost impossible to cover (i.e. judge) all the possible query-document pairs to compose a complete Information Retrieval Test Set. At the same time, the number of Documents in the test set and the set of Information Needs have to be of a reasonable size. In the literature, we can find a *rule of thumb*: 50 Information Needs has usually been found to be a sufficient minimum threshold for this type of set (Manning, et al., 2008).

There are many corpora for testing Documents Retrieval Models available in the Internet. Among them, some of the most known and used by researchers are: (1) Cranfield, (2) TREC, and (3) Reuters Corpora.

Cranfield (Manning, Raghavan, & Schütze, 2008) is composed of 1398 abstracts of aerodynamics journal articles collect at the United Kingdom, and it has a set of 225 queries. A particular characteristic of this corpus is that every pair (query, document) was judged which was possible because of its relatively small size. This project was considered a pioneer since it was one of the first corpora created for the evaluation of IR Systems; however currently it is considered dated due to its relatively small size.

Text Retrieval Experience – TREC (NIST-TREC, 2013), is a collection of corpora created to evaluate many types of distinct information retrieval systems. Those corpora are classified in thematic areas (called tracks) like web documents, chemical texts, spam documents, legal documents, microblog Tracks and medical documents. For instance, one of its sub-areas has 8 different corpora that together contain 1.89 million documents and 450 queries (information needs). Unlike the Cranfield track, this corpus usually does not have all the pairs (document, query) judged because of its size (Manning, Raghavan, & Schütze, 2008).

Finally, Reuters Corpora (RCV1, RCV2, TRC2) (NIST-Reuters, 2013; Lewis, Yang, Rose, & Li, 2004) consists of a collection of Reuters News used in language processing, information retrieval, and machine learning systems. It is composed of RCV1 (Reuters Corpus, Volume 1, English language), RCV2 (Reuters Corpus, Volume 2, Multilingual Corpus), TRC2 (Thomson Reuters Text Research Collection) containing about 810,000, 487,000, and 1,800,370 documents respectively. This corpus is the successor of the old Reuters-21578 corpus, providing more documents for better evaluation of IR Systems.

2.1.4.1 Polling Strategies

Relevance Judgments are an essential element of any Information Retrieval Relevance Corpus. In a Relevance Corpus, judgments are created by one or more individuals that are responsible for classifying documents of a given subject as relevant or not for them. The whole process of creating such judgments is usually performed by a poll, conducted by domain experts or potential users of an Information Retrieval System. Across the years many works have been developed about the best ways to conduct such polls. For instance, the basic methodology used in TREC corpus is composed of the following steps (Aslam, Pavlu, & Yilmaz, 2006):

1) Given a query q and a set of documents D, use X different models to extract from D the estimated top Y relevant documents by each system. For instance, use 10 models to retrieve 1000 documents each, for the query q;
2) Among all the retrieved documents, use a policy p to choose Z different documents. For instance, choose 100 random samples of the documents retrieved by the 10 models;
3) Let experts on the domain or potential users of IR System choose the relevant documents for the query q among the Z chosen previously;
4) Repeat the steps 1, 3, and 3 for at least 49 other queries totalizing a corpus of 50 test cases.

There are two important questions about this methodology: (1) how to avoid bias and (2) how many pairs (query, documents) should be evaluated. We could find guidelines in order to deal with these questions while creating a relevance corpus (Chris , Dimmick, Soboroff, & Voorhees, 2006):

- **Topic engineering**: design the queries to avoid problems that led to bias. For instance, queries containing words that are frequently present in the titles of the documents may introduce a bias to retrieve first the documents that contain these words on their titles. This tends to happen because some models give a high score to documents that contain the words on the queries on especial sections, such as on their titles;

- **Forming polls more efficiently**: use strategies to retrieve and judge documents that were not initially retrieved by the models used in the first step of the polling methodology described before. In some cases, it is also possible to try to use statistics or heuristics-based models to know how many pairs (query, documents) have to be judged;

- **Encouraging distinct retrieval approaches**: in order to complement the set of pre-defined queries, let users create their own queries so the diversity of the queries in the corpus can be increased;

- **Engineering the judgment set**: once finished the poll, only some subsets of the judgments are used, thus discarding those that may introduce bias. For instance, if a judged has evaluated many documents, compared to the others, we may want to decrease the number of documents evaluated by this judged to avoid creating a bias towards his/her own particular concept of relevance. This is an interesting technique when using an already pre-defined corpus like the Reuters. It may also significantly decrease the time needed to conduct the tests, if only a subset of the original corpus is used;

- **Use different evaluation methods**: the comparison of two or more metrics is useful when evaluating a document ranking model. For instance, if both two independent metrics shows that a system A has a superior performance compared to a system B, we can be more confident that A is in fact better than B for the domain of the corpus used during the tests.

In the literature, there are works that intend to minimize the effort of judging the relevance of documents in a large collection data. These works are based in statistical and heuristics strategies to select those documents that would be more important to increase the reliability of the results obtained when an Information Retrieval System is evaluated using a certain corpus.

Zobel (Zobel, 1998) introduces a metric to predict the number of relevant documents based in the judgments of the top k^2 documents for each query. On the next interaction of the poll we have to decide the queries that are going to have more j^3 documents judged. At this point, the queries that are less likely to have more relevant documents at the next j more deep documents are the ones that had less documents judged as relevant so far. Therefore, they are not prioritized in new interaction of judgments.

---

[2] 30, for instance.

[3] 10, for instance.

Aslam et al (Aslam, Pavlu, & Yilmaz, 2006; Aslam, Pavlu, & Yilmaz, 2005) propose a way to estimate the evaluation metrics based on the samples initially judged during the poll; after the estimation, new documents are selected and judged in order to reduce the variance of the evaluation metric. Another work (Carterette, Allan, & Sitaraman, 2006) explains that the more similar two models are, the more relevance judgments we will be needed to meet a difference criteria among the evaluation metrics. Hence, they propose an algorithm that can choose the documents to be judged in order to maximize the difference between two models evaluated by the same metric.

Furthermore, Cormack et al. (Cormack, Palmer, & Clarke, 1998) describe two different strategies. The first is called Interactive Searching and Judging, where the top documents for a query q are judged until the number of non-relevant documents reaches a threshold. The second is called Move-to-front Polling and prioritizes the judgment of documents retrieved by the query that returned more relevant documents in the top k documents. For instance, suppose that we want to judge 30 documents and we have two different queries *q1* and *q2*; among the top 10 documents, *q1* has returned 5 relevant documents, while *q2* has returned only 2; then this strategy will prioritize the judgment of next 10 (top 11 to 20) documents retrieved by *q1*.

The strategies listed here are applied essentially when we have a lot of documents to be judged. When judgments can cover only a few documents for each query (for instance, less than 5%), those strategies can be useful to increase the statistical significance of the obtained results. Their drawback is that all of them require a very well controlled environment to be used. For instance, (Cormack, Palmer, & Clarke, 1998) require the poll to be split in a few steps and on each of them the judges need to be fully accessible, which makes it difficult to be used in contexts such as those based on volunteers randomly recruited on the Internet.

2.1.4.2 Evaluating Document Rankings

Ranks of documents generated to satisfy Information Needs have to be evaluated; this is the reason why there are many different metrics to measure how good an information retrieval model is. In this section, we will cover some of them, particularly describing how they can be applied to estimate the performance of a

Document Ranking Model. The metrics to be discussed are: (1) Precision, (2) Coverage, (3) FMeasure, (4) Means Average Precision, (5) RPrecision; and the gain based evaluation metrics: (1) Normalized Discounted Cumulative Gain and (2) QMeasure.

2.1.4.2.1 Evaluation Metrics

Precision (Equation 2.6) is computed by the division of the number of relevant items retrieved by the number of retrieved items. In other words, precision represents the conditional probability of an item to be relevant, given that it was retrieved. An important variation of this metric is the K-Precision (also known as P@K) where only the first k retrieved items are considered.

$$Precision = \frac{\#(relevant\ items\ retrieved)}{\#(items\ retrieved)} = P(relevant|retrieved) \quad (2.6)$$

Coverage (2.7) is computed by dividing the number of relevant retrieved items by the number of relevant items. In other words, it is the conditional probability that an item is retrieved given that it is relevant.

$$Coverage = \frac{\#(relevant\ items\ retrieved)}{\#(relevant\ items)} = P(retrieved|relevant) \quad (2.7)$$

For combining both precision and coverage, there is a metric called FMeasure. It unites both metrics by the harmonic mean, as shown in Equation 2.8, where P represents Precision, C represents Coverage and α represents the weight of the combination. For example, if $\alpha > 0.5$ it means that precision is more important than coverage.

$$FMeasure = \frac{1}{\propto \frac{1}{P} + (1 - \propto)\frac{1}{C}} \quad (2.8)$$

Another important metric used to evaluate Information Retrieval systems is the Means Average Precision (MAP). This concept is computed by the average precision for the first K items in each query during the tests. Hence, let $Q = \{q_0, \ldots q_j, \ldots q_n\}$ be the

set of all test queries; let $I_j = \{i_{1j}, \ldots, i_{mj}\}$ be the set of the mj relevant items for a query q_j; and let R_{jk} be the set retrieved items at the first k positions for q_j, Equation 2.9 shows how to compute the Means Average Precision.

$$MAP(Q) = \frac{1}{|Q|}\sum_{j=1}^{|Q|}\frac{1}{m_j}\sum_{k=1}^{m_j} Precision(R_{jk}) \quad (2.9)$$

RPrecision (Equation 2.10) is a metric that requires a set of known relevant items R and computes the precision of $/R/$ items. R can be incomplete (i.e. does not necessarily contain all the relevant items) which makes it good to use this metric if the relevant items for the queries in the test set were compiled, using relevance judgments for the polled top k results of a particular model.

$$RPrecision = \frac{\#(relevant\ items\ retrieved\ at\ the\ first\ |R|\ items)}{|R|} \quad (2.10)$$

In contrast to the previous listed metrics, bpref (Buckley & Voorhees, 2004) is a function that shows the number of times that items judged as non-relevant are retrieved before relevant items. For a query that has a set of $R = \{r_1, \ldots, r_n\}$ relevant items and mi is the number of non-relevant items presents in the top $/R/$ items that were retrieved by a system before a relevant item ri, Equation 2.11 shows how bpref can be computed. Like RPrecision, bpref is a metric created to deal with incomplete data, i.e. corpus where not all the items were evaluated. An important bpref variation is called bpref-k, where instead of dividing mi by $/R/$ it is divided by $k + /R/$. This variation is used when we want to ensure that at least k items are used in the computation of its value.

$$bpref = \frac{1}{|R|}\sum_{i=0}^{|R|} 1 - \frac{m_i}{|R|} \quad (2.11)$$

2.1.4.2.2 Gain based Evaluation Metrics

Unlike other metrics studied so far, gain based metrics were developed with two important priorities in mind (Järvelin & Kekäläinen, 2002):

1. Highly relevant items are more valuable than marginally relevant items;
2. The higher[4] the ranked position of a relevant item, the less value it has for the user, because it is less likely that the user will ever examine it.

Given this assumption, the concept of relevance of an item is not a binary value anymore: it becomes a multilevel value that may vary according to the corpus that the researchers are using to test a ranking model. For instance, given a relevance corpus, its items may be evaluated as highly relevant, relevant, marginally relevant and non-relevant. Notice also that while this new way to look at the concept of relevance provides richer evaluation options, it is not possible to use it properly if researchers have only access to binary relevance judgments.

Now we can formalize the Gain function for an item, as shown in Equation 2.12, that gives the relevance value of an item in a Relevance Corpus.

$$Gain: I \rightarrow \mathbb{R}, where\ I\ is\ a\ set\ of\ Items \tag{2.12}$$

Another important feature of this paradigm is that instead of looking only if relevant items were retrieved or not, now it is possible to look at the position they were retrieved. Hence it is also possible to decrease the performance of a ranking model in a gain-based metric if it returns too many relevant items at a high rank.

Let us now introduce the concept of Gain (G) Vector, shown in Equation 2.13:

$$G[j] = Gain(i_j), where\ i_j\ is\ the\ item\ ranked\ at\ the\ index\ j \tag{2.13}$$

The gain vector is the key concept behind the computation of the Normalized Discounted Cumulative Gain (NDCG) (Järvelin & Kekäläinen, 2002). NDCG is a metric that computes a vector of gains and gives as result the sum of its all elements. This vector is calculated based on the gain of each retrieved item. It also aims to

[4] We mean that most important documents are ranked in a lower position, like 1st., 2nd or 3rd.

decrease the item score as its rank increases (thus the “discounted” on its name). For instance, a relevant item is retrieved at the 5th position has a relevance value superior to an equally relevant item retrieved at the 6th position.

The first step to compute the NDCG is the definition of the Cumulative Gain (CG) Vector:

$$CG[j] = \begin{cases} G[1], if\ j = 1 \\ CG[j-1] + G[j], otherwise \end{cases} \quad (2.14)$$

To clarify this concept, assume that we have an item Ranking Model that retrieved 10 items for a given query, but only the items at rank 3, 5 and 9 are relevant. Moreover, the Gain of each item is respectively 2, 3, and 1. Then our Gain Vector (G) would be $G = [\ 0, 0, 2, 0, 3, 0, 0, 0, 1, 0\ ]$, and our Cumulative Gain (CG) Vector would be $CG = [\ 0, 0, 2, 2, 5, 5, 5, 5, 6, 6\ ]$.

The next step to compute the NDCG is defining the Discounted Cumulative Gain (DCG) Vector. DCG decreases the item score by dividing the Cumulative Gain elements by the log of their rank. If b is the base of the logarithm, we can define recursively the Discounted Cumulative Gain (DCG) Vector as shown in Equation 2.15:

$$DCG[j] = \begin{cases} CG[j], if\ j < b \\ DCG[j-1] + \left(\frac{G[j]}{log_b j}\right), if\ j \geq b \end{cases} \quad (2.15)$$

Continuing our example, let us take b = 2. In this case, our Discounted Cumulative Gain (DCG) Vector would be $DCG = [\ CG[1], CG[2], DCG[2] + (G[3]/log_2 3), DCG[3] + (G[4]/log_2 4), DCG[4] + (G[5]/log_2 5), DCG[4] + (G[6]/log_2 6), DCG[6] + (G[7]/log_2 7), DCG[7] + (G[8]/log_2 8), DCG[8] + (G[9]/log_2 9), DCG[9] + (G[10]/log_2 10)\ ]$ which ultimately led us to the following vector: $DCG = [0.00, 0.00, 1.26, 1.26, 2.55, 2.55, 2.55, 2.55, 2.87, 2.87\ ]$.

In order to continue to present Gain metrics, we need to introduce two new concepts: the Ideal Gain (IG) Vector and the Ideal Cumulative Gain (ICG). The first

one is the Gain vector that would be the result of a perfect Ranking Model. For instance, looking at our previous sample again, if there is only three relevant items and we need to retrieve ten items then, the Ideal Gain Vector would be: $IG = [\,3, 2, 1, 0, 0, 0, 0, 0, 0, 0\,]$. Furthermore, the Ideal Cumulative Gain Vector is the Cumulative Gain Vector of a perfect Ranking Model, in our example It would be: $ICG = [\,3, 5, 6, 6, 6, 6, 6, 6, 6, 6\,]$.

At this point we are finally able to introduce the Normalized Discounted Cumulative Gain (NDCG) equation. This metric is calculated by computing the DCG vector, and dividing each element of it by the element at the same position on the ICG vector, which generates a Normalized Vector (NV). Once the NV is computed, we just need sum all its elements and divide the result by the size of the vector to reach the value of the NDCG metric. Equation 2.16 shows how to compute this metric for the top-k retrieved elements by a ranking model.

$$NV = \left[\frac{DCG[1]}{ICG[1]}, \ldots, \frac{DCG[k]}{ICG[k]}\right] \quad \text{(a)}$$

$$NDCG(k) = \frac{\sum_{j=1}^{k} NV[j]}{k} \quad \text{(b)} \qquad (2.16)$$

Adopting once more our example, the NV would result in $[\,0.00, 0.00, 0.21, 0.21, 0.43, 0.43, 0.43, 0.43, 0.48, 0.48]$. As a consequence, NDCG would result in 0.31.

QMeasure (Sakai, 2004) is an alternative metric to NDCG; according to its creators, is better than the former because its denominator does not “freeze” after a specific rank r (notice that the value of the elements of the ICG vector stops increasing after a specific rank r). Here we will introduce the Blend Ratio Vector (BR), which is based on the cumulative gain vector (CG), its ideal version (ICG), the vector $R[i]$ containing the number of relevant items at the rank i, and a constant β. Given that, we must sum all the blend ratios for 1 to k and divide the result by the number of relevant items at the rank k to compute QMeasure as shown on equation 2.17 (a) and (b).

$$Blend\ Ratio[i] = \frac{(\beta * CG[i]) + R[i]}{ICG[i] + i} \quad \text{(a)} \qquad (2.17)$$

$$QMeasure(k) = \frac{\sum_{i=1}^{k} BlendRatio[i]}{R[k]} \quad \text{(b)}$$

2.1.4.3 Significance Test

When Information Retrieval researchers publish the results of their experiments showing that a ranking particular model is better than another based on a specific metric, statistical significance testes are often used to complement the results obtained (Sanderson & Zobel, 2005).

Once the evaluation metric for the comparison of two IR systems is estimated, we need to define two additional important concepts to perform a statistical test. The first is called the null hypothesis; for instance, that both IR systems are actually the same. The second is the significance level that is computed by using the results of two models on an evaluation metric previously defined. Then, we have to specify how that value could have occurred under the null hypothesis, which is usually called the p-value. The null hypothesis is rejected when the p-value is low, otherwise the difference between two IR systems maybe just consequence of noise or bias at the experiment (Smucker, Allan, & Carterette, 2007).

The research community of Information Retrieval uses many distinct tests of statistical significance. According to (Sanderson & Zobel, 2005; Smucker, Allan, & Carterette, 2007), the most important ones are the Student's paired t-test (t-test), the Wilcoxon signed rank test, the sign test, the bootstrap and Fisher's randomization. Researchers found evidence that there is little practical difference between t-test, bootstrap and randomization; however the randomization test is preferred in IR experiments for some reasons; for example, the bias that the bootstrap test may introduce in the evaluation in some specific scenarios could be higher when compared to the randomization test (Smucker, Allan, & Carterette, 2007).

2.1.4.3.1 The Randomization Significance Test

For applying the Fisher's Randomization test (Fisher, 1935; Smucker, Allan, & Carterette, 2007) in IR experiments when comparing two systems A and B, the null hypothesis is defined as: A and B are both identical. Hence, system A has no difference when compared to system B on the evaluation metric, queries, corpus, and relevance judgments.

To explain how this test works, imagine that we have 100 different queries used on the experiment and that the metric selected for the evaluation was the average of the RPrecision on all the 100 queries. Let the average RPrecision of the systems A and B be 0.223 and 0.154 respectively. The difference between A and B is 0.069. We have also 200 different values of RPrecision (100 for each system). Each value has a label that identifies the system that generated it. There are 2^{200} possible ways of labeling the 200 results. Now it is possible to test all the 2^{200} forms of labeling the results to know how many of them actually show a difference equals or greater than 0.069. This number divided by 2^{200} is what is called single sided p-value. However, if instead of using the difference we actually use the absolute value of the difference, we would have computed what is called the two-sided p-value.

Unfortunately, sometimes computing all the possible labeling permutations takes a lot of time, thus information retrieval researchers usually create a limited number of random samples of all possible permutations. In our previous sample, suppose that we generate $500\,000$ permutations and then we compute the difference in the RPrecision average for all of them. If 1050 are ≤ -0.069 and 785 are ≥ 0.069 our two-sided p-value would be $(1050+785)/500000 = 0.00367$. This low value shows that the difference of 0.069 on our evaluation metric is unlikely, hence we can reject the null hypothesis. Therefore, it is possible to conclude that A is not equal to B whenever it comes to its performance on the RPrecision metric.

2.1.4.4 Alternative Strategies to Test Information Retrieval Systems

The standard relevance corpus based tests that we focused so far are aimed to test the ability of a ranking model to retrieve many relevant items on a list of a specific size. However, many other different aspects of a ranking model or even a full information retrieval system can be evaluated as well.

One important aspect of Information Retrieval systems is how fast it is when it comes to retrieve items. Given that in real life IR applications have to deal with a large amount of data, if they are not efficient enough to process a query and give the results, their users may stop using it. In a nutshell, test regarding this aspect are made by giving a set of queries as input to a system and then measuring the average time that it takes to process and retrieve the items for all of them.

Moreover, another aspect linked to the large amount of data that IR systems have to handle is the storage of an item. This leads to compacting data strategies to significantly decrease the space required to store all items. Tests focusing on this aspect are made by compacting corpora of items and the indexes used by a system with different compacting strategies, and then comparing the amount of space that each strategy was able to save.

Finally, the last aspect we will discuss here is related to the usability of an information retrieval system. This is related to how its interface is easy to use and can be tested in many distinct ways. Paper prototyping, polls and interviews with users among many others approaches are applied to improve the usability of Information Retrieval systems.

Since on this thesis we are focusing on opinion ranking models, we made just a brief overview of the most important issues of Information Retrieval systems; we focused mostly on how to test models, without detailing the other aspects.

2.2 Social Search

Social Search is expected to be the future of search engines. Academic researchers (Evans & Chi, 2010) and companies like Google (Dupont, 2008; Google, 2014) and Microsoft (Morris, Teevan, & Panovich, 2010a) have been studying in the last years how social components could help a user to fulfill its information needs.

Experiments developed by Microsoft researchers showed that people usually have more confidence in answers from friends. They also have discovered that people frequently believe that social networks are better than search engines when answering subjective questions like products recommendations (Morris, Teevan, & Panovich, 2010a). Moreover, there is evidence that the experience of information searching could be improved by the integration of search engines and social resources (Morris, Teevan, & Panovich, 2010b).

We can understand the meaning of social search by looking two complementary definitions:

Definition 2.6 – Social Search by Morris et al: The term social search means the process of finding information on-line by the help of social resources, like, for

example, asking to friends or unknown people at the Internet for answers (Morris, Teevan, & Panovich, 2010b).

Definition 2.7 – Social Search by Evans & Chi: Social Search is used to describe search acts that make use of social interactions with other individuals. These interactions may be explicit or implicit, local or remote, synchronous or asynchronous (Evans & Chi, 2010).

Both definitions are complementary. Definition 2.6 stresses on the interactivity of social search, while Definition 2.7 discusses interaction types in social search. To continue our exploration of this relatively new research field, we will introduce some samples of its application on Internet services and applications.

2.2.1 Social Search and Internet Services

Across the years, Social Search has been used on many distinct applications on the Internet. From innovative web tools to classical search engines, the social aspect can be introduced to improve the user's experience. In this subsection, we will show and analyze distinct applications of Social Search. This discussion will enable us later to introduce the requirement that a document retrieval model needs to be considered also a social document retrieval model.

2.2.1.1 The Aardvark Project

The Aardvark (Horowitz & Kamvar, 2010) project was one of the pioneers to apply social search on the Internet. This project had the objective of finding the best person to answer a question to another one. To do so, they used the profile of both the person who is searching for the answer and the person that will answer the question. The dynamics of the social search engine developed by them can be summarized as follows:

1. The user a enters a question to the system;
2. The system searches on its users database to find another user b (the best one) to answer the question;
3. The question is redirect to the user b;
4. b answers the question;
5. The answer is send to a;

In order to find who is the best user to answer a certain question, the search engine looks into social aspects such as the experts in a subject and direct peer-to-peer evaluation (a user can evaluate another). The team responsible for the project development was later hired by Google, so currently the platform is not available anymore.

2.2.1.2 Google Search

Google started to explore social aspects on its services in 2008. The first notable public experiment that Google made to promote social search aspects on its search products was named SearchWiki (Dupont, 2008). In this effort, they wanted to provide to their users a way to customize search by re-ranking, deleting, adding, and commenting on search results (Figure 2.1). This was made possible by adding few new options to the search engine interface, such as buttons to remove results from the list of documents or to increase the rank of a document. Thus, they were able to collect data from users to customize the results. The changes made by a user changed only its own results. However, it was also possible to share the changes with other users (i.e. a user was able to see the changes that others made).

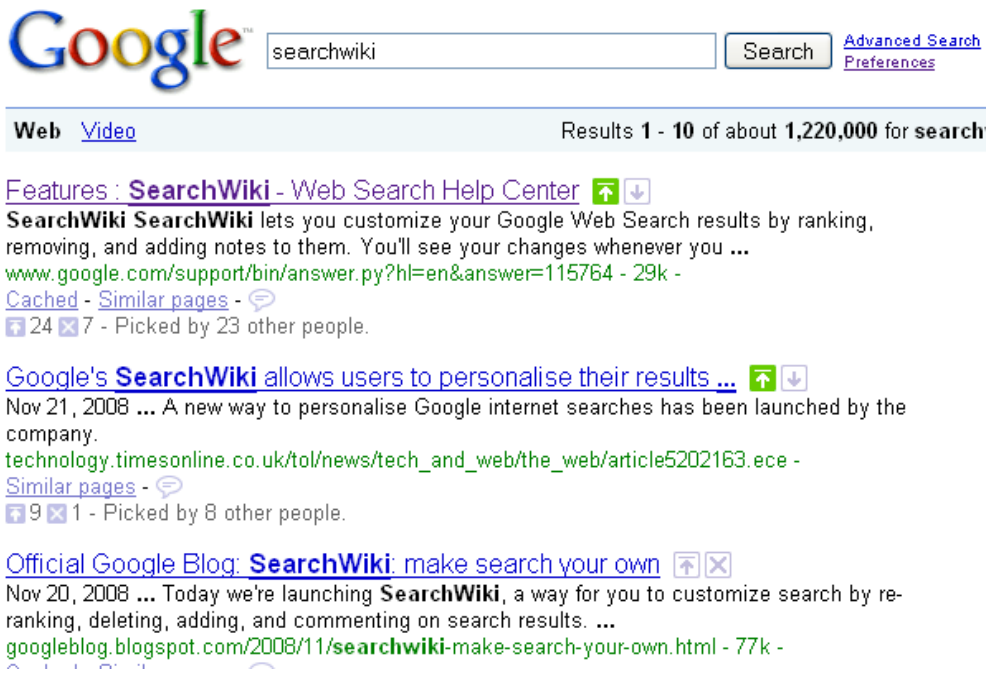


Figure 2.1 – Google SearchWiki.

A few years after the end of the SearchWiki experiment, Google started to integrate their search engine and their new social network, called Google Plus. The result of this new integration is exemplified in Figure 2.2, which shows how the system shows the results for the query “just dance review”. It is possible to see that the third result, the review from the site IGN.com, has as picture of its author, in this case Keza MacDonald. The picture has a link to Keza’s profile on the social network Google Plus, which according to Google (Google, 2014) “help users discover great content”.

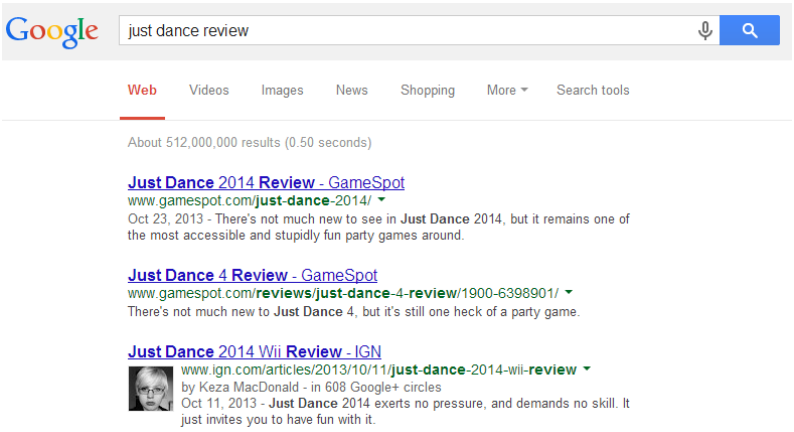


Figure 2.2 – Results for the query “just dance review” on the Google search engine.

The addition of the author information (Google Plus Profile) in the search engine, coupled with the fact that many Google users have also a Google Plus profile, means that their search system now has access to a complete social network. This opens space to the development of many customization strategies. For instance, Google now has the information needed to find blogs of the friends of a user, and so when this user performs a search, Google can increase the rank of those blog posts in the results. Following the social search trending, in next few years we expect not only Google but also others search engines to make more use of the social profile of their users and authors of the web pages to provide customized search results.

2.2.1.3 YouTube Comments

Since November 2013, Youtube started to explore social aspects of their users (including the authors of their videos) on the comments posted on its videos (Google, 2013). The comments were initially listed from the newest to the oldest with few popular comments in the top (the ones that others users liked more). However, with

the introduction of the new comment ranking system, comments are now linked the Google Plus account of the user that made them (Figure 2.3).

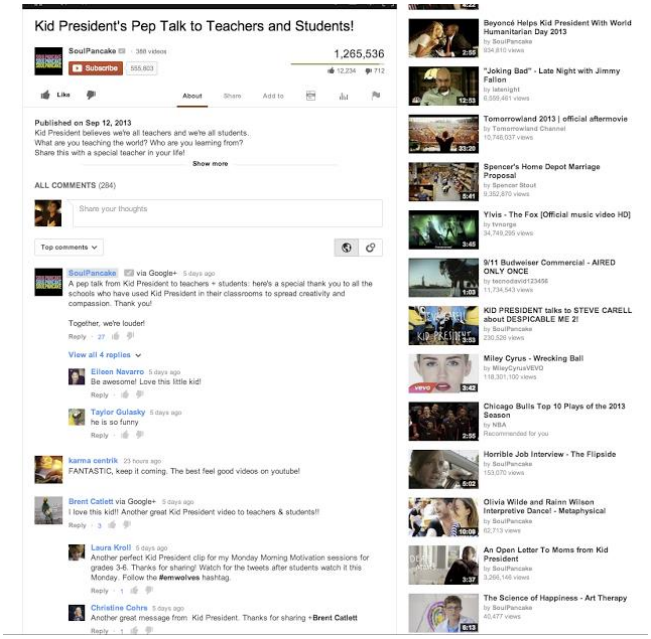


Figure 2.3 – How comments look like in Youtube with the new ranking model.

The link created to the social profile of the users allows them to increase the rank of the comments following heuristics such as:

- Comments from the video author;
- Comments from popular personalities;
- Engaged discussions about the video;
- People in the Google Plus Circles on whom is browsing the video (i.e. somehow related to the user).

By introducing this new ranking model for comments, YouTube expects them to become conversations that matter to their users. Moreover, we can see that especially the last heuristic presented in the list would not be feasible without the integration to the Google Plus profile of the user that is browsing the video comments, and the authors of the comments.

2.2.1.4 *Bing Social Search*

In a similar fashion to the Google, Microsoft started to integrate their general purpose search engine called Bing to the social network Facebook. Once a Bing user connects its Facebook profile with the site, the main results are complemented by an additional column showing posts from friends on Facebook and from experts across other major social networks (Microsoft, 2014).

Figure 2.4 – Bing Social Search for the query "Penguin".

Despite the similarities, while the Google Search engine currently introduces social information about the profile of the authors of the documents, the Bing search engine offers two distinct lists of results. The first is the list of document that it would traditionally display. The second is a list of profiles and content form social networks. We believe that this distinct approach is a result of the challenges of integrating distinct types of content into a single list of ranked entries. For instance, how can Bing know if a blog post (containing many words) is more relevant or not than a single sentence on a Facebook post? This type of problem is not our focus here, but it represents how complex social search can be when it comes to estimate the relevance of different types of documents.

2.2.1.5 *Facebook Graph Search*

Finally, one of the most promising projects on the field of social search is the Facebook's new search system called Graph Search (Facebook, 2014). Their new approach to the search on social networks aims to be able to interpret natural

language queries and return as result content from others users of the network (Figure 2.5).

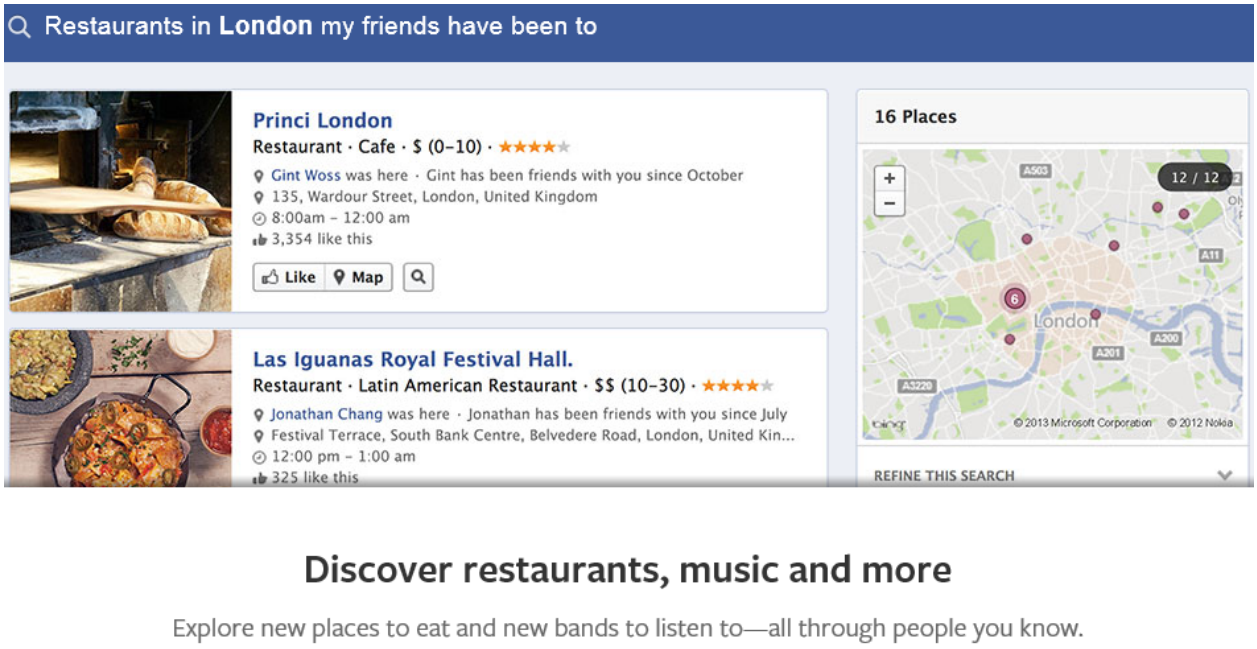


Figure 2.5 – Preview of the Facebook social search.

A few samples of queries that their new search has the ambition to correctly process for a certain user include:

- Peoples who like cycling;
- Music his friends like;
- Photos of his friends in New York;
- Cities his family visited;
- Photos before 1900.

By looking into the samples of queries, we can see that all of them rely on two specific features: (1) the profiles of the Facebook users; (2) the meta-data of the content of the social network (dates, places, etc.). By the time this text was written, unfortunately the graph search service was still on a restricted beta state. However, even with the limited information released about the service by Facebook, we were

able to see the evidences of how important social aspects were to the feasibility of this new search tool.

2.2.2 Towards a Social Document Ranking Model

So far, we have made an overview of the social search and its application on many services across the Internet. However, when compared to Information Retrieval, Social Search is still relatively new. Its definitions and applications are still on the beginning of their life.

By compiling this knowledge, we are ready now to present our vision of what can be considered a Social Document Ranking Model. This is vision that we will use during the remaining of this thesis.

A standard Document Ranking model can be represented by function that has two inputs: a query (representing an information need) and a document. The output of the function is a real number representing the estimation of the document relevance for the query.

Our vision of a Social Document Ranking Model, while keeping the same characteristics of a document model, must have the following important characteristics as well:

1. It must use the profile of the authors of the documents while estimating its relevance;
2. It must use the profile of the person who is searching for documents while estimating their relevance;
3. It must use the documents' metadata (for instance, their publishing date) while estimating their relevance;
4. It must be able to provide customized results, according to who is searching for the documents.

Given this characteristics, we are now able to formalize what we consider a Social Document Ranking model as shown on Equation 2.18. The definition is based on four parameters: an information need, represented by the set of queries Q; two users represented by the set U, respectively the authors of the documents and the person that is searching for the documents; and a document, represented by the set

D. Also, in order to be fully considered as social, a model's formal definition of document must not be limited to the set of words that it contains, but must reveal important metadata such as its date and the place where it was written.

$$Social\ Document\ Relevance: QxUxUxD \rightarrow \mathbb{R} \quad (2.18)$$

Notice that the fourth characteristic of what we see as social ranking model (i.e. the customization of the results) becomes a natural consequence of the use of the set of users and of the document metadata. Notice also that one could provide customized results only using the information about the person that is searching for the documents. However, intuitively, we believe that ignoring the author of the document would limit the relevance estimation, as it would be impossible to know if the author and the person are friends or even if they have interests in common. As we could see on the examples of social search applications just presented, most of them make use of both profiles (user and author), besides the metadata of the documents.

In the rest of the work, if a document ranking model does not have all the four listed characteristics, we are not going to classify it as a social model. For instance, if the model uses only the profile of the authors of the documents, we will use the expression "partially social" to label it.

Moreover, it is possible to classify as customizable or/and social not only a whole model, but its parameters as well. For instance, if a document ranking model uses tf-idf as one of its parameters, we can say that this specific parameter is not social and not customizable, since its equation only takes into consideration the text of the document. To be classified as customizable, a ranking model must have at least one parameter classified as customizable. Likewise, to be classified as social, a ranking model must have at least one parameter classified as social.

As we are interested in ranking opinions, a first step is to classify a certain text as an opinion, as discussed next.

2.3 Text Classification

Given the fast growth of the amount of information in the Internet, text classification became a very popular strategy to organize data (Joachims, 1998). Formally speaking, we can define the main task of the text classification as:

Definition 2.8 – Text Classification: Given a set of documents D, a set of classes C and a document d in D, find out to which class c in C the text of d fits better.

Strategies for solving this problem have multiples applications such as (Manning, Raghavan, & Schütze, 2008):

- Automatic Span Detection;
- Automatic Detection of sexually explicit document;
- Automatic E-mail labeling according to the folders at the user inbox;
- Topical Search.

There are several approaches to solve classification problems. One of the simplest is to classify documents manually by experts that would be able to perform this task with a high precision. However, the higher the amount of documents, the higher is the number of people needed to categorize the data. Furthermore, this choice also increases the chance of errors, since the experts may not share always the same view.

Consequently, whenever there is a large volume of data it is necessary to define an automatic or semi-automatic process for the classification of documents. A possible solution could be to create a set of rules able to define which class a given document belongs to. Again, this strategy is unfeasible on a large scale, as it requires consulting several experts to assemble the set of rules, and a lot of time to test and adjust the model to the problem.

Therefore, to solve the problem of text classification it is common to use different learning algorithms. With this approach, given a set of previously classified documents, also known as the training set, new documents are categorized according to the classes of highest similarity with them (Li & Jain, 1998). Thus, through sample documents, the algorithms are able learn the pattern of each class and then use this pattern to define whether a document belongs to a given class or

not. Following this approach, several classification techniques were developed; in the next subsections, we will briefly introduce some of them: (1) Naïve Bayes; (2) K-Neighbors (3) Logistic Classification; (4) Support Vector Machines.

2.3.1 Naïve Bayes

This technique works by computing the conditional probabilities of a feature (usually a word) being present in a document of a given class. Each feature has an estimated probability for each possible class (Manning, Raghavan, & Schütze, 2008). For instance, for a feature f a higher probably for a class c means a high indication that a document containing that feature will be a sample of the class c. In this algorithm, there is the assumption that for each class, the probability of a feature to be part of a document is independent of the others (Li & Jain, 1998). Equation 2.19 (a) shows how the class of a given document $d = (f_0, \dots, f_n)$ is chosen based on the $arg\,max$ function, where $n \in \mathbb{N}$ and $f_i\ (0 \leq i \leq n)$ is a feature of d,. For each class $c \in C$, where C is the set of all classes in problem, the class with the highest value in equation is chosen as estimated class by the algorithm. In Equation 2.19 (a), two probabilities are defined: the probability of any document of being a sample of class C, $P(c)$, and the probability of the presence of a feature f, given a certain class c. The first probability is estimated by Equation 2.19 (b), where N_c is number of documents of the class c, and N is the total number of documents in the training set. The second probability can be estimated by Equation 2.19 (c), where F_{cf} is the number of occurrences of a feature f in documents of the class c and V is the set of all features in the training set.

$$class(d) = \arg max \left[logP(c) + \sum_{1 \leq k \leq n_d} logP(f_k|c) \right] \quad \text{(a)}$$

$$P(c) = \frac{N_c}{N} \quad \text{(b)} \qquad (2.19)$$

$$P(f|c) = \frac{F_{cf}}{\sum_{f' \in V} F_{cf'}} \quad \text{(c)}$$

2.3.2 K-Neighbors

This technique labels an unclassified document d as a member of the class c if the closest (most similar) document to d in a previously classified document set D (samples) was labeled as a member of the class c (Li & Jain, 1998; Manning,

Raghavan, & Schütze, 2008). To measure similarity between two documents d_1 and d_2, both are converted into a vector containing the weight of each term present on both of them. These weights are defined using the tf-idf metric, previously discussed in section 2.1. Hence, each element on the vector represents the tf-idf score of a specific word on the document. Intuitively, this strategy represents each document as a n-dimension vector, and then computes the similarity between the document to be classified and the documents on the sample set.

Formally, given the vectors V_1 and V_2 with the weights of each term in common on the documents d_1 and d_2, respectively, the cosine similarity is used to measure the similarity (also called proximity) between them as shown on Equation 2.20, where the operator // // denotes the norm of a vector and V_1^T is the transposed of the vector V_1.

$$S(d_1, d_2) = \frac{V_1^T V_2}{||V_1|| ||V_2||} \quad (2.20)$$

There are a few variations of this strategy that take into consideration more than one close document (to the one to be classified) while estimating a class. For instance, given a document d we could define that given the five closest documents on the samples, we will estimate the class of d based on the most common class of the five closest documents. The specific variation of the K-Neighbors classification (1 neighbor, 2-neighbors etc…) should be defined according to the domain that the classifier is created for. In some cases, researchers may decide to test a few variations before deciding which one should be used.

2.3.3 Support Vector Machines

This technique is based on the principle of structural risk minimization (Joachims, 1998; Hearst, Dumais, Osman, Platt, & Scholkopf, 1998). The algorithm seeks for a hypothesis h such that it is possible to ensure the best true error. This strategy tries to achieve such hypothesis by testing different possible hypothesis and, for each of them, computing the probability that it will make an error in one instance not previously seen and randomly selected. For example, a document d present in the training set is considered as a point in a space of dimension D, where D is the

number of different features present in all documents in the training set. This technique assumes that the points (documents) present in each of the classes are linearly separable, and then tries to find a hyper plane that can separate them. Equation 2.21 (a) shows how a hyper lane is computed. This equation aims to find a hyper lane composed of for the vectors $x = [x_0, \dots, x_n]$ that satisfy its equality. In addition, when we want to find the class of a vector $x = [x_0, \dots, x_n]$ Equation 2.21 (b) presents how this process is made (in a two class problem).

$$\{ x \mid (w \cdot x) + b = 0 \}, \quad where\ w \in R^N\ and\ b \in R \qquad (a)$$

$$c(x) = sign\left(\sum_{i=1}^{n} v_i \cdot k(x, x_i) + b\right) \qquad (b) \qquad (2.21)$$

$$where\ v = [v_0, \dots v_l,]\ is\ parameter\ vector$$
$$and\ k : R^N x\ R \rightarrow R\ is\ called\ kernel\ fuction$$

2.3.4 Logistic Classification

This classifier tries to satisfy constraints by modeling them using uniform models (Nigamy, Lafferty, & McCallum, 1999). For example, in a two-class problem where 75% of the documents containing the word “feeling” are opinions, intuitively when given a sentence contains “feeling” it is possible to say that it has a 75% chance of being an opinion. However, if the sentence does not have the word “feeling”, it still has a 25% chance of being an opinion. Hence, this algorithm sees each feature in the training set as a possible constraint, like “feeling” in our example, and tries to build a model that satisfies all of them. In a typical two class problem Equations 2.22 (a) and 2.22 (b) show how this algorithm computes the probabilities of two classes ($c=0$ and $c=1$) of a vector $x = [x_1, \dots, x_n]$ where each element is a feature of the document; in both equations $w = [w_0, \dots, w_n]$ is a weight vector computed using the training samples.

$$P(c = 0|x) = \frac{exp(w_0 + \sum_{i=1}^{n} w_i x_i)}{1 + exp(w_0 + \sum_{i=1}^{n} w_i x_i)} \qquad (a)$$

$$P(c = 1|x) = \frac{1}{1 + exp(w_0 + \sum_{i=1}^{n} w_i x_i)} \qquad (b) \qquad (2.22)$$

2.4 Multi-Agent Reputation Models

Reputation can be defined as the opinion that an entity has about another one, but can also be seen as the reliability of an entity, commonly called agents or individuals (Srinivasan, Teitelbaum, & Wu, 2006). However, the same entity may have different reputations on distinct domains making this concept multidimensional (Zacharia O. , 2000). Thus, each entity may have a distinct reliability for each domain, which means that each domain can be seen as a dimension. For instance, give the domain quadratic equations, it is expected that a math teacher should be more reliable to provide feedback and resolve problems than an English teacher.

The concept of reputation has been applied in many distinct fields of Computer Science. Within the Internet, on many online shopping services like e-bay, Amazon[5] and PayPal[6] users can buy and sell products or services. In order to help their users, those sites provide a reputation model, so the users that want to buy something are able to know if the seller is reliable and vice-versa. They usually allow their users to evaluate each other, composing a global reputation measure for members of the site. Thus, such sites have their own ways of evaluating users regarding the reliability of the transactions they made on their services.

On the other hand, academic researchers in multiagent systems have been creating reputation models with many different objectives and applying them in different contexts, such as partner choice (Sabater & Sierra, 2005). Focusing on these academic agent reputation models, on the remaining of this section we will briefly introduce the main foundations of four reputation models: (1) Sporas, (2) Histos, (3) Liar, and (4) Repage.

Sporas (Zacharia O. , 2000; Zacharia, Moukas, & Maes, 2000) is a reputation model created to be simple and capable of working well independently of the number of interactions and evaluations among the entities in a community. It provides a global reputation value for each member in a community. The model constructs the

[5] www.amazon.com

[6] www.paypal.com

reputation of the members of a community based on several principles, among them we can highlight the fact that the reputation of an old member in a community never falls below the reputation of a member who has just entered in the community, independent of how unreliable the old member is. Another important feature of it is that the computation of the reputation between two entities in this model takes into consideration only the last rating, thus ruling out any evaluation that has been made in the past.

Histos is a model (Zacharia O. , 2000; Zacharia, Moukas, & Maes, 2000) that was designed to be more complex when compared to Sporas. Therefore, it has several features that differentiate it from Sporas. For instance, Histos does not deal with direct evaluation among members, however it allows the existence of witnesses who can evaluate members even without being directly involved with them in an interaction (also called transaction). But this also augments the model complexity: for example, a reliable seller is not necessarily a reliable witness (Sabater & Sierra, 2005). From a formal point of view, peer evaluations in this model are represented as a graph, where nodes and directed edges represent the respective weights ratings among members. So, as long as there is a path for the member A to the member B, it is possible to calculate the reputation of B with respect to A.

The Repage (Image and Reputation Among Limited Autonomous Partners) model (Sabater, Paolucci, & Conte, 1996; Sabater & Paolucci, 2007) proposes an approach based on a peculiar view of two concepts: image and reputation. According to the authors, given two individuals A and B in an environment, the image A has of B is precisely how it individually evaluates B. However, the reputation of B is viewed as partially dependent on A, because this is how B is evaluated by all members in a community. In other words, reputation is a social value, independent of each individual opinion. Another important feature of this model is the use of fuzzy logic to represent rankings of the reputation or image of individuals. Because of this choice, the numerical values of the evaluations are translated into four possible discrete values: very poor, poor, neutral, good, and very good.

Finally, the L.I.A.R. (Liar Identification for Agent Reputation) model (Muller & Vercouter, 2010) was designed to control iterations of individuals. Therefore, it allows the formal expression of the communication rules that should compose a system. By

using tools based on the model, one can develop agents that are able to understand interactions between other agents, and eventually to detect which of them are violating the community rules, thus reflecting this detection on the agent reputation. The model is based on the main premise that communication rules are homogeneous and known to all agents participating in the simulation. Therefore, agents who are not aware of the rules, but are still part of the simulation, can possibly be evaluated as harmful as they can break the rules without notice.

2.5 Evolutionary Algorithms and Genetic Algorithms

Evolutionary Algorithms (EA) simulate the natural evolution in order to solve complex problems. Across the years, researchers have been applying evolutionary algorithms to solve problems related to areas like search, optimization and machine learning (Whitley, 2001). After decades of basic research and applications, despite the limitations and simplifications of the simulations developed, EA is considered very robust (Bäck & Schwefel, 1993).

There are three main sub-areas of the Evolutionary Algorithms: (1) Evolution Strategies; (2) Evolutionary Programming; and (3) Genetic Algorithms (Bäck & Schwefel, 1993). All three of them share many similarities. They are based on a population where each individual represents a possible solution to a problem. This population evolves through a non-deterministic process of selection, mutation and recombination.

Selection can be intuitively viewed as a process of filtering the individuals of a population. Just like in the natural evolution, EA tends to select the best (the finest) individuals to continue to live and eventually be replicated or combined. The evaluation of each individual in the population is computed by a fitness function that is used during the selection phase to find out which are the best individuals in the population.

Mutation is a process that tries to simulate the similar natural evolution phenomena. Among the individuals of a population, a few are randomly selected and have some of their features randomly changed on the next generation of individuals.

Recombination usually involves two or more individuals. In this process, analogously to sexual reproduction in natural evolution, features of distinct individuals are switched to create new individuals in the next generation.

In a nutshell, evolutionary algorithms can be summarized in a pseudo-code adapted from (Bäck & Schwefel, 1993), as shown in Figure 2.6 :

```
t := 0; // time
p := { } // population
initialize p0 ; // random initialization
while ( objective not reached ) do
begin
     p': = recombine pt;
     p'' = mutate p';
     p''' = evaluate p'';
     pt+1:= select p''';
     t := t + 1;
end
```

Figure 2.6 – Evolutionary Algorithms pseudo-code.

While sharing similarities, the three paradigms also have important differences. For instance, while evolution strategies and evolutionary programming focus on the mutation process this latter has a secondary role on genetic algorithms when compared to recombination (Bäck & Schwefel, 1993). However, these paradigms have influenced each other and new implementations freely borrow ideas from each other (Whitley, 2001). Since Genetic Algorithms are considered the most recognized technique among EA, on the remaining of this section we will focus on it and, more specifically, on its canonical variation.

Genetic Algorithms are a specific type of Evolutionary Algorithms where the individuals of a population are represented as strings over a finite alphabet (Whitley, 2001). Usually, the alphabet chosen is composed of only 0's and 1's, generating individuals that are strings of bits. In a generation, individuals are combined in a process that can be seen as a simplified version of the sexual reproduction,

eventually generating a different individual. Individuals can also be mutated, by flipping some of their bits (Whitley, 2001). Each individual is evaluated by a fitness function, which gives higher values for better individuals. (Whitley, 1994)

"Canonical" is the term used to refer to the original version of genetic algorithms (Bäck & Schwefel, 1993). In this version the fitness value for the individual i is defined by f_i/f_{avg}, where f_i is the fitness of the individual i and f_{avg} is the average of the fitness of all individuals in the population. The execution of the algorithm starts with the current population. Then selection if performed to reach the intermediate population. Finally, recombination and mutation are performed creating the current population. The whole process of creating the next population from the current population constitutes one generation in the execution of genetic algorithms (Whitley, 2001). This process is exemplified in Figure 2.7.

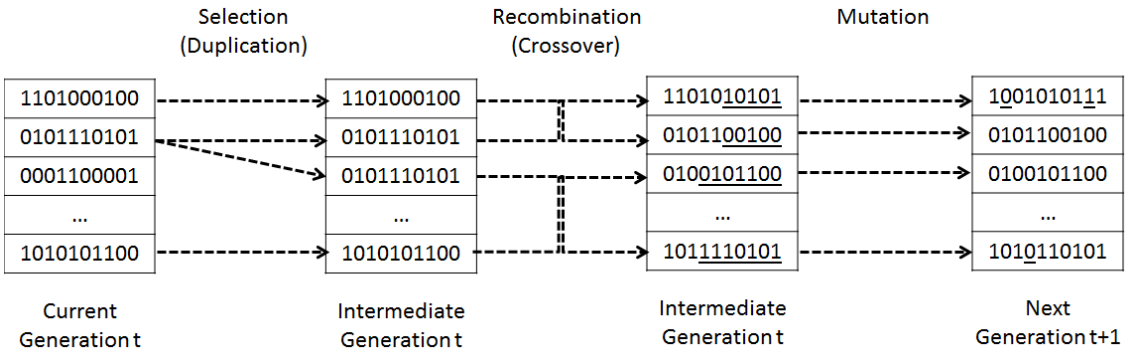


Figure 2.7 – Overview of how a canonical Genetic Algorithm generation, based on (Whitley, 2001)

In canonical genetic algorithms, the probability of an individual from the initial population to be copied to the intermediate population is proportional to its fitness. For instance, while modeling the selection process on Genetic Algorithms, we may define that an individual with fitness value equal or higher to 1.0 will have at least one copy in the intermediate population For instance an individual with fitness value 1.70 will have one copy plus 70% of chance for a second copy in the intermediate population. However, an individual with fitness value equal to 0.45 will have only 45% of chance for a single copy on the intermediate population.

Furthermore, the canonical implementation uses a concept called crossover to perform the recombination. Give two randomly chosen strings $s1$ and $s2$, we first

select randomly a recombination point p, which is a number between 1 and the size of the strings. Then, we split both strings on that point, generating two strings from the beginning of both to point p ($bs1$ and $bs2$) and two other strings from the point p to the end of both ($es1$ and $es2$). Finally, we perform the recombination by concatenating the beginning of a string with the end of the other. For instance, given two strings 1110100111 and 0000111100, if the combination point selected was 3, the final strings after the crossover would be: 1110111100 and 0000100111.

Once the recombination is finished, the mutation phase can be finally performed. This is achieved by mutating each bit of each individual with a small probability p_m, which is typically lower than 1% (Whitley, 1994). If a bit was chosen to be mutated, it can have a new randomly value generated for it (notice that it may not change) or it can be just flipped (which will always generate a new value). .

2.6 Conclusions

In this chapter, we discussed many concepts and requirements for the proper understanding of the next chapters of this thesis.

Each one of the research fields discussed here will be used in the further chapters of this thesis. Now we can summarize their importance on the remaining of this thesis:

1) **Information Retrieval**: one of the most important pillars our work. This area provided to us mainly document relevance metrics and evaluations methodologies that could be adapted and used on our model, described in Chapter 4;
2) **Social Search**: a relatively new research field where we could find how to customize the results of a document ranking model, as well as an opinion relevance model, as we will see in Chapters 4, 5 and 6;
3) **Text Classification**: An important approach to classify and extract opinion from documents, as illustrated in Chapter 5. This is fundamental to differentiate our work, which deals with opinions, from others that work with sentences or documents.
4) **Multi-Agent Reputation Models**: A research field that inspired some of the key concepts of our model as we will see further in Chapter 4. For

instance, reputation enables us to estimate how users of a social network may perceive the author of an opinion;

5) **Evolutionary Algorithms**: techniques that can be useful to learn the balance of elements of an equation. For instance, given two members a and b of an equation E, by using EA we can learn that the best way to combine both members is by summing them, and that member a should have twice the value of member b. So, equation E may end up looking like E = … 2a + b … GA have an important property that is the nonexistence of assumptions about the problem that they are solving. This property makes it a good starting point (i.e. baseline) to test the effectiveness of more specific (i.e. problem specific) techniques. Therefore, if we have a problem we can initially use Evolutionary Algorithms to solve it. In a second step, we may try more problem specific strategies and compare their solutions with the baseline performance established by the Evolutionary solution. In Chapter 6, we address how EA can be applied to learn the balance of the equation that compiles distinct aspects an Opinion Relevance Model. By using EA in this context, we are also creating a new baseline to test new strategies to balance our model in future experiments.

Now that the fundamental concepts are established, we are finally able to discuss Opinion Mining, its applications and review the various opinion corpus found in the literature. We are also able analyze the concepts of Opinion Relevance and Social Opinion Relevance. All these subjects will be the focus of our next chapter, when we will discuss Opinion Mining and Opinion Relevance.

3 Opinion Mining

Textual information can be categorized basically in two types: (1) facts and (2) opinions. Facts are objective declarations about entities or events, while opinions are declarations reflecting subjective fillings of people or perceptions about entities or events (Liu B. , Opinion Mining, 2009). Since the popularization of the Internet, especially regarding blogs and social networks, the way people express and have access to opinions has been changing. For example, when someone wants to make his mind about a polemic subject, he can use not only the traditional ways (friends, relatives, television, newspapers or magazines), but he can consult the information present in the Internet (Liu, 2010) as well.

Information Retrieval (Manning, Raghavan, & Schütze, 2008), briefly discussed in Section 2.1, is a computer science research field that studies how to mine information from the Internet. However, researchers in this field, in most cases, do not handle facts and opinions differently, both are considered as information. Consequently, those techniques are not able to properly help users demanding primary for opinions. This is one of the reasons why researchers created a research field called Opinion Mining (OM) that can be defined as follow:

"Sentiment analysis, also called Opinion Mining, is the field of study that analyzes people's opinions, sentiments, evaluations, appraisals, attitudes, and emotions towards entities such as products, services, organizations, individuals, issues, events, topics, and their attributes." (Liu, 2012)

In order to distinguish opinion mining from information retrieval, let us analyze a hypothetical piece of information about a camera: "it has a good image resolution, but its zoom isn't as good as other cameras already in the market". It's possible to notice the existence of two different opinions in the text, the first one is positive and the second one is negative. Moreover, the first opinion is directly related to the image resolution. The second refers to the zoom feature by a comparison with other cameras. While information retrieval will handle that sentence as a text document piece, opinion mining will try to identify and eventually index opinion aspects present on it.

The vision of this research filed can be clarified by the formal definition of what an opinion is (Liu, 2012):

Definition 3.1 – Opinion: a quintuple $<en, ta, se, ho, ti>$ where en is an entity, ta is a target, se a sentiment about the target, ho is the opinion holder, and ti is the time when the opinion was expressed.

On the definition above, the entity en is the subject on which the opinion is given. The target ta can be any feature of the entity about as opinion was expressed. Furthermore, se can be a positive, neutral, or negative sentiment, or a numeric rating value expressing how intense the sentiment is. This concept is equivalent to what we also refer as the orientation of an opinion. The holder ho is the person the expressed the opinion, i.e. the author of the opinion. Finally, ti is a representation of the time which the opinion was published.

Using our previous sample, let us supposed the sentence “it has a good image resolution, but its zoom isn’t as good as other cameras already in the market”. This opinion was about a camera called CamPixPro expressed by an user of a social network called Jeff Miles on July 25, 2014. Now let us analyze how the formal definition of an opinion can be used on the first opinion of Jeff’s sentence. First, CamPixPro is the entity the opinion is about; “image resolution” if the target of the opinion; positive is the sentiment expressed by him; he is the holder of the opinion which was published on July 25, 2014. Therefore we can represent his opinion formally by the quintuple (CamPixPro, “image resolution”, positive, Jeff Miles, July 25, 2014). This sample can be generalized to a task of extracting all the quintuples representing opinion in a document. This mission is considered the key task of opinion mining (Liu, 2012).

3.1 Task and Applications

So far, we intuitively introduced some of the opinion mining targets. But there are other research objectives and task related to this research field, among them we can highlight (Liu, 2010; Liu, 2012):

- **Problem formalization:** in order to properly (mathematically and scientifically) handle a problem it is important to correctly define the main concepts involved. Hence, limitations, restrictions, goals and many other

aspects usually need to be formally described. Formalization of a problem gives more credibility for its results, avoiding ambiguity and misconceptions which also is useful for other researchers evaluating or studying the project;

- **Sentiment and subjectivity analysis:** given a document, first identify if it has opinions on it, and then, if it has, classify them using a taxonomy such as: positive, negative and neutral. This is the most basic fundament of Opinion Mining. Usually OM researchers have to deal with this problem as an intermediate step to solve a more complex problem. Hence, this is perhaps the most explored subarea of OM with many different strategies and applications.

- **Characteristic oriented sentiment analysis:** usually, a characteristic of a product or a subject is an attribute that he possesses, and in the OM context it is possible to extract this information from an opinion. For example, on the domain of the computers, we may say that their characteristics are CPU type, hard drive size, amount of RAM and so on. Thus, characteristic oriented sentiment analysis is essentially the process of discovering the attributes that have been cited by the author on the text of the opinion;

- **Sentiment analysis in comparative sentences:** the use of metaphors or comparisons to better exemplify an idea is not rare when someone expresses his/her opinion about a subject, for example, in the sentence "the notebook battery lasts, on the average, as much as any other ordinary notebook battery". Thus there are sentences that do not directly express an opinion, interpreting them and treating them appropriately according to the problem’s context is one of the most complex challenges of OM;

- **Search engine and information retrieval:** just as general purpose search engines (such as Google) are very popular and useful nowadays, opinions oriented mechanisms can be as well. However, they would deal with just a more limited range of queries. An opinion search engine is an information retrieval system able to receive as input a query representing a subject and to produce a list of documents (or opinions) about the subject as result. In order to create such a system, researchers have to deal with many sub-problems specially regarding opinion storage, retrieving algorithm, interface usability and utility (relevance);

- **Spam detection and opinion utility (opinion search and retrieval):** Just like in information retrieval, spam is also a huge problem in opinion mining. For instance, fake opinions may be expressed for commercial purposes: a company may pay someone to write good/bad reviews about its own/competitors' products. Meanwhile, the opinion utility, i.e. its relevance, is related to the quality of the opinion. In theory, each opinion could have an associated relevance value. This value would be an estimation of how important the opinion is. Hence, the utility of an opinion can be viewed as a score that represents its importance degree, and therefore it could be used to create opinion ranking functions.

- **Opinion Summarization**: Most applications need to analyze a large number of opinions. Complementary, while looking for opinions about a subject, someone usually wants read many opinions about its features. The task of summarizing opinions can be useful for both problems since it decreases the amount of data that a program or a person needs to deal with. Summaries of opinions may be a fraction of the opinions in a single review or even from distinct documents. Another important aspect of this task is its quantitative perspective. For instance, 80% of authors being positive about a product may be key information for someone that is searching for the opinions.

During the last couple of decades, researchers have been studying and developing many projects on the opinion mining field. In this sub-section, we will provide a high-level view of projects in this area, focusing on their application on real-life problems. It is important to reinforce that, while we are focusing on a very specific OM problem on this thesis, there are many other researchers on this field working on projects related to the challengers previously discussed.

Regarding the application areas of OM, through a literature review we can find that its concepts are applied in the composition of vendors' reputation in Internet shopping sites (Ghose, Ipeirotis, & Sundararajan, 2007); Extraction and classification of opinions expressed about politicians in online forums (Esuli & Sebastiani, 2006); Mining of consumer reviews about products on shopping sites through the Internet (Lee, Jeong, & Lee, 2008); Ranking blogs by adding opinions as one of the parameters of the function score of sites (Attardi & Simi, 2006). These works showed the versatility of the application of the concepts of OM. It may consist of the main area related to the project (like the project on opinions about politicians) or, on the other hand, it may be applied to complement and improve previous results (like the project where opinion were used to help to rank blogs) and used in conjunction with other techniques to reach the final goal.

3.2 Opinion Corpora

In order to conduct OM experiments, researchers must use Opinion Corpora. If we look at the challengers discussed in the section 3.1, almost all of them need to an opinion corpus to be studied or solved. For instance, if researchers want to create a new opinion classification algorithm, they will need a corpus to test it and measure how good it is.

When it comes to the granularity, we can categorize an opinion corpus essentially in two types: (1) corpus of documents where opinions are annotated, where the sentences were manually labeled as opinions or not; they also may contain information regarding the classification of the opinions such as their orientations (positive, negative, neutral, etc.); (2) corpus of opinions, where opinions are represented as paragraphs or sentences; they may also offer information about the orientation of the opinion. Relevance judgments are another type of information that both types of opinion corpora may have. This information is essential if

researchers want to use a corpus to test an opinion relevance model, or a document relevance model that uses information from opinions on the documents. Depending on the corpus, the granularity of the relevance judgments may be documents or opinions.

Among the opinion corpora available and well documented at the Internet, we can highlight:

- **The Wal-Mart Corpus** (Yu, Diermeier, & Kaufmann, 2009) – Self defined as "a multi-granularity corporate opinion corpus for opinion retrieval, classification and aggregation", this corpus is composed of a compilation of 1080 documents (business news articles) about the Wal-Mart company. The paragraphs of the documents were manually judged as relevant or not, as opinions or not and finally the orientation of opinions were also annotated. Paragraphs that referred to Wal-Mart were considered relevant. The opinion orientation classification was based on four categories: positive, negative, neutral or mixed;

- **MPQA (New Multi-Perspective Question and Answer Corpus)** (Stoyanov, Cardie, & Wiebe, 2005) – This corpus is composed of 98 documents retrieved from the blog platform WordPress. Each document had its sentences classified manually as opinions or not. This corpus also contains 30 questions that its documents may contain an answer to. The answers to some of the questions are facts, while others are opinions. For each question, the researchers annotated the region on the documents where the answer was found. Moreover, they also represented the confidence associated to the annotation , i.e. how confident the judge was that the document region, in fact, contains the right answer to the question;

- **NTCIR-7 (NII Test Collection for IR Systems) Opinion Corpus** (Seki, et al., 2008) – A corpus created for the Multilingual Opinion Analysis Task at the NTCIR-7 workshop. It is composed of 72 topics (test cases) in languages such as Japanese, English, Traditional Chinese, and Simplified Chinese. The documents were retrieved from newspapers, composing a total of 971

documents. The relevance judgments were made on two granularities. First, they manually labeled the documents as relevant or not to the topics on the corpus; then, among the relevant documents, they judged all their opinions (sentences expressing opinions) as relevant or not to the topics.

- **TREC 2008 Blog Corpus** (Balog, Serdyukov, & Vries, 2010) – A corpus created from the TREC 2008 blog track. This corpus contains 50 new topics, as well as 100 other topics from the previous years. While the authors did not specified the total number of documents in the corpus, they provided the information that the collection has 148 GB in size and that for each topic approximately 1000 document had they relevance judged. The documents (blogs posts including its comments) were judged as relevant or not. Furthermore, they also had their opinion polarity judged. This corpus allows tests to be made in two distinct forms: (1) retrieval and ranking of the documents with positive opinions; (2) retrieval and ranking of the documents with negative opinions.

- **TREC 2011 Microblog Track** (Ounis, Macdonald, Lin, & Soboroff, 2011) – This corpus is composed of 16 million tweets (small sentences with at most 140 characters) retrieved from the social network Twitter. It has a total 50 topics (test cases), each of them representing an information need for a specific point in time. A topic in this corpus can be summarized as: “at time t, find tweets about topic X”. The tweets were judged as non-relevant, relevant and high relevant. Given a topic, the authors retrieved 30 tweets that were related to it and then judged all of them as relevant or not. Intuitively, we can say that many of the tweets in this corpus are opinions; however, the corpus itself does not offer information if a tweet is an opinion or not. However, since most of the relevant projects related to our work were based on this corpus, we decided to include it here.

Table 3.1 summarizes the main aspects of the opinion corpora that we discussed so far. We can see similarities and variations of the multiple aspects of them (domain, relevance and granularity) that overall do not interfere directly in our

objective, but can be used as reference when we will further introduce our own corpus. However, we can also see that from a social point of view, *almost all of them lack information about the author of the opinions/documents and especially about the judges that decided what is relevant or not*. The only exception is the TREC Micro-blog 2011, which provides information about the author of the tweets. Unfortunately, as we will see in the section 3.3, the information that this social network contains about a user profile is very limited (which is in-line with the Twitter objectives), and hence it does not allow many aspects of the author to be easily explored by an opinion ranking model. The limitations of those corpora also became a limiting factor to projects that used them, as we will see in the next section where the concept of opinion relevance will be studied.

Table 3.1 – Features of the Opinion Corpora available on the Internet.

Corpus	Domain	Relevance	Granularity	Author Profile	Judge Profile
Wal-Mart	News on Wal-Mart	Binary	Paragraph	No	No
MPQA	News Blogs	Multi-level	Sentence	No	No
NTCIR-7	News Paper	Binary	Document and Opinion	No	No
TREC Blog Task 2008	Blogs	Binary	Document	No	No
TREC Micro-blog 2011	Micro Blogs	Multi-level	Document	Yes	No

3.3 Opinion Relevance: Evolution and State of Art

Now that the Opinion Mining challenges, applications and corpora were exemplified, we will discuss the specific problem that we studied during our project: Opinion Relevance. We will address how this concept evolved through the years, so we will be able to compare and show the limitations of the previous projects in other to make clearer the motivation behind our own work.

Despite the lack of projects directly related to our objective, especially regarding the social aspects, it's still possible to find in the literature some projects using opinions in the development of ranking functions. The concept of opinion relevance started to be explored by researchers as one of the parameters of a document relevance model. They used it mainly to rank documents. They also used a document corpus to perform experiments and Information Retrieval metrics to see if they could achieve better results.

Usually, their research hypothesis is somehow similar: by introducing parameters related to the opinion relevance, instead of only using the traditional topic relevance approaches from Information Retrieval, it is possible to achieve better result on Information Retrieval Metrics while ranking documents.

Later, a second generation of projects started to go even further on the exploration of opinion relevance, by investigating how the whole text could be used as an opinion, instead of only sentiment-related words in a document. We could also find in this generation a few social aspects being used to estimate the relevance of an opinion. However, they still share characteristics with the first generation such as (i) using a document corpus to perform the experiments, (ii) applying the opinion relevance concept as a parameter to rank documents and (iii) absence of customized results.

In the following subsections, we describe some projects from both generations, giving a high level description and describing their opinion relevance estimations, when this was available in the project sources. At the end, we discuss the limitations of those projects, and the possible reasons for these limitations.

One of the first works in this field was published in 2006 (Mishne, 2006). It describes a ranking function associated with aspects (parameters), like sentiment analysis, spam detection, and link-based authority estimation, in order to rank blog posts. The function presented in this project is composed of a linear combination of those aspects. The ranking function has seven parameters in total. Among them, two are based on the opinions present in the documents. These parameters are: (1) Post Opinion Level; and (2) Feed Opinion Level. Both are categories of words defined on The Inquired Lexicon Dictionary (Stone, Dunphy, & Smith, 1966). The author computed the opinion-based parameters by extracting all the sentences about a subject in a document. He then counted the number of occurrences of sentiment-related words on the sentences, and divided it by the total number of words in the sentences. The corpus used on the experiments was the TREC 2006 Blog Task (Macdonald & Ounis, 2006). Moreover, the author has applied MAP (*Mean Average Precision*), RPrecision and bpref metrics to evaluate the weight of the proposed parameters in the ranking function. At the end of the experiments, the conclusion was

that the link-based authority estimation value was not relevant, as the other aspect parameters, to improve the ranking function.

Another related project (Zhang & Ye, 2008) proposed a ranking function for documents composed by two main values: opinion relevance and topic relevance. Both values are linked by a quadratic combination, because the authors believed that this type of combination is superior to linear interpolations. Their opinion relevance function is based on the occurrence of sentiment-related words near to the words on the query. The opinion score of a document do given a query qu and a set of sentiment-related words SE is computed according to Equation (**3.1**), where $co(t,qu/wi)$ is the frequency of the sentiment word t which is co-occurred with any word on the query qu within the window wi (the number of the window's words) and $c(qu,do)$ is the query term frequency in the document (number of words in the query and the also in the document). Both λ and wi are constants that should be balanced during the experiments. The sentiment-related word dictionary was based on a compilation of works from six different dictionaries including WordNet (Miller, 1995), the General Inquirer (Stone, Dunphy, & Smith, 1966) and the SentiWordNet (Baccianella, Esuli, & Sebastiani, 2010).

$$Socore_{op}(do, qu, SE) = \sum_{t \in SE \wedge t \in do} (1 - \lambda) \frac{co(t, qu \mid wi)}{c(qu, do)wi} + \lambda \qquad (3.1)$$

Based on that, the authors developed a raking function and conducted experiments with the TREC Blog06 (Macdonald & Ounis, 2006) and Blog07 corpora (Macdonald, Ounis, & Soboroff, 2007). In those experiments they applied as evaluation metrics MAP, RPrecision, and precision at top 10 (P@10) results in order to evaluate different variations of their ranking function. At the end, they have concluded that the quadratic combination together with logarithm normalization was the best way to implement their ranking function.

A third project (Huang & Croft, 2009) presents a probabilistic ranking function for documents. The ranking function is also based on the number of occurrences of sentiment-related words. It was used together with others techniques, like a query expansion based on a synonymous dictionary to build a score function for

documents. The values were combined by linear interpolation. The parameters introduced in this document ranking model were: (1) topic relevance; (2) query-independent sentiment expansion; and (3) query-dependent sentiment expansion.

The first parameter in this model is related the occurrence of query words in the document, as every other ordinary Information Retrieval topic relevance metrics. The two remaining, on the other hand, try to estimate the relevance of opinions in a document from two complementary perspectives: one that is static from all queries and other that takes the words in the queries into consideration. The first is called query-independent sentiment expansion parameter, and it is based on an estimation of the weight of all sentiment-related words in a document. This estimation is learned by comparing the Average Precision of the original query qu with the Average Precision of an expanded query qu'. The new query qu' contains the same words in qu, plus a sentiment-related word wo. At end of the learning phase, it is possible to know what the most relevant sentiment-related words are (those that generated higher increments on the Average Precision) and, consequently, give higher ranks to the documents that contain these words. The third and final parameter of this model is called query-dependent sentiment expansion. It is proportional to the probability of the words in a query being not only present in the document, but being sentiment-related as well. Hence, given a query qu, a word of qu qu_i, a document do, and a word wo the query-dependent sentiment expansion parameter is computed as shown in Equation (3.2):

$$P(qu \mid do, wo) \approx \prod_{i=1}^{n} P(qu_i|do, wo) \quad \text{(a)}$$

$$P(qu_i|do, wo) = \begin{cases} \frac{count(qu_i, do)}{|do|}, if w\ occurs\ in\ do \\ 0, otherwise \end{cases} \quad \text{(b)} \qquad (3.2)$$

In the project, the authors performed experiments in the TREC Blog06 (Macdonald & Ounis, 2006) and COEAE08 (a document dataset for the Chinese language compiled by the Chinese Academy of Sciences) databases to discover the weight of each component in their scoring function that maximizes the results. The main metric used is this project was MAP. However, other metrics like RPrecision and P@10 were also computed and presented in the paper.

Following our historical evolution of the Opinion Relevance, another project (Attardi & Simi, 2006) presents a ranking model that uses, as one of its parameters, a relevance estimation function for opinions in documents. To achieve that, subjective words considered to be carrying an opinion bias were tagged in the documents. Then the authors build a system that receives as one of its inputs the subject on which the user is searching for opinions. To interpret those queries, their system looks for words carrying an opinion bias near to the words representing the subject. Documents with more sentiment-related words that are close to the query words have their rank increased. In order to evaluate the results, they made experiments in the TREC Blog 06 (Macdonald & Ounis, 2006) using the P@5 (precision at top 5) as the main evaluation metric. Their experiments showed an increase on performance of the metrics when exploring annotation on sentiment related words.

So far, we can see the most of these works presented two important aspects in common increase the relevance of a document: (1) the number sentiment-related words the document has; (2) the number of sentiment-related words the document and the query have in common. Recently, a new generation of projects proposing new methods to estimate the relevance of an opinion has started. This new generation keeps exploring the sentiment-related words on the documents, but exploring new ideas such as the lexical aspects of the opinions. These new approaches try to explore the text of the opinion (usually a sentence) and find characteristics on it that can make an opinion relevant or not.

The first sample of this generation that we will discuss here is (Orimaye, Alhashmi, & Siew, 2012a). It proposes a model for document ranking based on opinion relevance. In this model, a document is considered as a set of sentences, and the authors use Natural Language techniques to extract information about them. For instance, one of the information they want to extract is what they call “Entities”, which are the words which other words expressing subjective feelings refer to them. Their score function is the average of the relevance of all sentences in a document do for a query qu, as shown in Equation (3.3) :

$$DocumentScore(qu, do) = \frac{1}{n} \sum_{se \in do} OpinionScore(qu, se) \quad \text{(a)}$$

$$OpinionScore(qu, se) = P(se)P(E|S) \quad \text{(b)} \qquad (3.3)$$

In this model, the OpinionScore function is based on two distinct concepts: the Opinion Likelihood, *P(E/S)*, and the Prior Relevance Belief, *P(se)*. The first means that the more close words referring to the "Entities", the more relevant an opinion is. The second, Prior Relevance Belief, is computed based on the numbers the words in common between the query and the opinion (sentence); the more words in common, the more relevant an opinion is. The experiment conducted in this project focused on testing the impact of the proximity between words (a key aspect of this model) compared to non-proximity based methods. The authors performed experiments using the TREC Blog 2008 dataset using the metrics MAP, R-precision and P@10. Their results showed that syntactic and semantic-based opinion search, using deep Natural Language Proceeding techniques, are achievable and helpful.

The relatively new project proposed in (Xu, et al., 2012) studied the concept of Opinion Relevance on blog feeds. According to them, a feed is a set of documents from the same blog. The motivation was to find blogs that tend to express opinions about a given subject, so those interested on the subject could follow those feeds, and to track influential and interesting opinions in the blogosphere. The authors proposed a unified framework for feed retrieval of opinions in blogs, which combines topic relevance and opinion scores. They introduced a topic-specific variable ex_{qu} to denote opinion expressions (i.e. a sequence of words) about the query qu. Using this variable, they formalized their relevance model as the probability of a feed fe being generated, given a query qu and the opinions expression related to it ex_{qu} as shown in Equation (**3**.**4**).

$$P(fe|qu, ex_{qu}) = P(fe)P(qu|fe)P(ex_{qu}|qu, fe) \quad (3.4)$$

The equation is divided in two major components: (1) *P(fe)P(qu/fe)*, which deals with the topic relevance by multiplying the probability of feed being generated by conditional probability of feed being generated given a query; and $P(ex_{qu}/qu,fe)$,that deals with the opinion score. Since topic relevance has already been studied by many other researchers, they focused the work on the opinion relevance aspect of the model. Therefore, they concentrated their resources on modeling $P(ex_{qu}/qu,fe)$ that is presented in Equation (3.5) where wo is a word on the vocabulary of all words V.

$$P(ex_{qu}|qu, fe) \propto \sum_{wo \in V} P(wo|qu, fe)P(wo \mid ex_{qu}) \quad (3.5)$$

They called their solution to the estimation of $P(wo/ex_{qu})$ as Topic-Specific Model (TOM) and their estimation of $P(wo/qu, fe)$ as Topic-biased Feed Model (TFM). TOM reflects language usage of topic-related opinion expressions, which helps identifying topic relevant opinions. TFM, on the order hand, was estimated to capture salient content information in the blog feed with bias towards the topic, and, consequently, help reflecting whether the feed shows principal inclination to expressing opinions about the topic. Intuitively, this model tried to identify whether a feed express opinions about a topic and if the opinions use expressions known by the model. On the experiments they compared different TOM and TFM different estimation approaches using the TREC Blog 2009 and 2010 corpus computing MAP, P@10 and R-precision.

Finally, the last project of our state of art review explores both the textual and the so-called "social" aspects of the opinions on the social network Twitter[7] to build an Information Retrieval System (Luo, Osborne, & Wang, 2013). Using a subset of the TREC Tweets 2011 Corpus (Ounis, Macdonald, Lin, & Soboroff, 2011), the authors conducted relevance judgments with volunteers creating their own Opinion Relevance Corpus. The data they gathered was used as an input to a ranking learning algorithm, and so they could create an Opinion Ranking Function. This whole process, which represents most of their research methodology, is illustrated at Figure 3.1.

[7] www.twitter.com

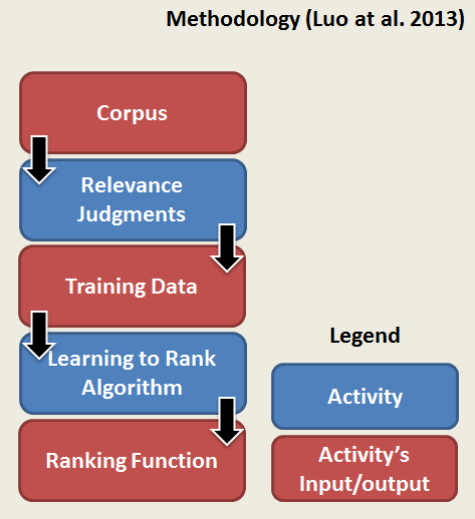


Figure 3.1 – Methodology used by (Luo, Osborne, & Wang, 2013) on their project.

This project is perhaps the work with most similarities with our own work, with respect to the adopted methodology, the obtained solutions and the experimental settings. All these similarities justify why this particular project is described in more details in this section. However, despite these characteristics in common, both approaches have important distinctions, as we address in the next few paragraphs.

As inputs for their relevance function, they used many parameters, including Social Network aspects and Twitter specific aspects. The parameters are: (1) the social features; (2) and what the authors defined using the expression "Opinionatedness Feature".

The social features are the following:

- **URL** – 1 if a tweet (twitter post) contains at least on a link on it, 0 otherwise;
- **Mention** – 1 if a tweet refers to at least one other user of the network, 0 otherwise;
- **Hashtag** – 1 if a tweet contains at least one hashtag (i.e. contains a word started with "#"). 0 otherwise;
- **Recency** – the age (in seconds) of a tweet;
- **Statuses** – the number of tweets an author has, intuitively, the more tweets he/she has the more relevant their opinions are;

- **Followers and Friends** – the number of followers and friends of the author of an opinion, intuitively, the more followers and friends an author has the more relevant he/she opinions are.
- **Listed** – Twitter users can be listed in groups. Intuitively, if a user is listed many times, it means that his/her opinions are relevant to many people. This parameter measure as many times a user was listed.

The Opinionatedness Feature of a tweet is presented in Equation (3.6), and it is computed by the sum of relevance of each word on the tweet/text (sentence) that meets a criterion. In the equation, $p(wo|do) = count(wo, do)/|do|$ and it represents the frequency of the word wo in the document do divided by the number of words on do. Notice that on the domain of this project documents are tweets, which are in most cases are sentences with a very few words.

$$Score(do) = \sum_{wo \in do \wedge x^2(wo) \geq m} p(wo|do).Opinion(wo) \tag{3.6}$$

The last part of the equation, $Opinion(wo)$, required the authors to manually label a subset of the their tweet corpus as subjective (opinion) or not (non-opinion). Once they finished this task, they were able to use the frequency of the words on the subjective and the objective tweets to compute $Opinion(wo)$ for each wo in a tweet text. This is shown in Equation (3.7) (a), while Equation (3.7) (b) details how to compute the chi-square x^2 of a word wo. On both the equations, swo is the number of subjective tweets containing wo; $snwo$ is the number of tweets not containing wo; owo is number of tweets objective containing wo; $onwo$ is the number of objective tweets containing wo; awo is the number of tweets containing wo; $anwo$ is the number of tweets not containing wo; st is the number of subjective tweets; ot is the number of objective tweets; and, finally, t is the total of number of tweets that they manually labeled.

$$Opinion(wo) = sng\left(\frac{swo}{awo} - \frac{owo}{ot}\right).x^2(wo) \quad \text{(a)}$$

$$.x^2(wo) = \frac{(swo.onwo - snwo.owo)^2.ot}{st.ot.awo.anwo} \quad \text{(b)} \tag{3.7}$$

Regarding the experiments, the authors compared variations of their model and also the BM25 (Robertson, Walker, Jones, Hancock-Beaulieu, & Gatford, 1995), concluding that the so-called social features of their model were able to improve the MAP results for the corpus they have created for the experiment. They also performed statistical tests using the Students t-test (Seize, 1977).

3.4 Discussion

During our literature review, we could observe the evolution of the concept of Opinion Relevance. Something that started as a single parameter for document ranking evolved to a lexical approach, trying to extract opinion information from the text and also to explore many other opinion relevant aspects. The main features of those projects are summarized in Table 3.2. Additionally, we have inserted our work in the table, in order to highlight its difference with the previous ones found in the literature. The features of our model and how it was tested are respectively the subjects of Chapter 4 and 6.

Table 3.2 shows that most of the work used blogs as an application of the models (column Domain), and since blogs are composed by posts (documents) they usually rank documents (column Ranks) as well. If we look at the Opinion Relevance concept proposed by them, we notice that in most of the cases it is estimated by exploring the text of the opinion. Only one project explored the meta-data of the opinions (column Metadata), hence opinion's aspects like its author and its publishing date were almost not explored yet. Still regarding the opinion relevance, we could not find in deep studies on the profile of the opinion consumer (column Customized).

When we look at the experiments that were carried on, we will see that MAP, R-Prec and P@10 are the most popular metrics (column Metrics), which is a requirement of the TREC task that most of the projects followed. Rarely the experiments offered any type complementary tests, especially regarding the significance of the results achieved (column Significance). Finally, most of the projects used the multiples versions of the TRAC Blog corpus (column Corpus), which is a well-established and well-known source of data for information retrieval experiments regarding the document ranking functions.

Table 3.2 – Features of the related work.

	Domain	Ranks	Opinion Relevance			Combination	Metrics	Significance	Corpus
			Metadata	Customized	Social				
(Mishne, 2006)	Blogs	Doc.	No	No	No	Linear	MAP R-Prec P@10	None	TREC Blog 2006
(Attardi & Simi, 2006)	Blogs	Doc.	No	No	No	Not available	P@5	None	TREC Blog 2006
(Zhang & Ye, 2008)	Blogs	Doc.	No	No	No	Quadratic	MAP R-Prec P@10	None	TREC Blog 2006 2007
(Huang & Croft, 2009)	Blogs	Doc.	No	No	No	Linear	MAP R-Prec P@10 bpref	None	TREC Blog 2006; COEAE08
(Orimaye, Alhashmi, & Siew, 2012a)	Blogs	Doc.	No	No	No	None	MAP R-Prec P@10	Negative KL-divergence	TREC Blog 2008
(Xu, et al., 2012)	Blogs	Doc.	No	No	No	None	MAP R-Prec P@10	None	TREC Blog 2009 2010
(Luo, Osborne, & Wang, 2013)	Micro-blogging	Optioned Tweets	Yes	No	Partial	Linear / SVM_{rank}	MAP	t-test	Own corpus based on TREC Tweets 2011
Social Opinion Releve Model	Independent	Opinions	Yes	Yes	Yes	Linear, Multiplication, Division, Exponential	RPrecision, MAP, bpref, QMeasure, NDCG	Fisher Randomization Test	Social Opinion Relevance Corpus

Among all the projects, we can see that the most recent we could find (Luo, Osborne, & Wang, 2013) offered a very distinct approach compared the previous ones cited here. This project explored multiple aspects an opinion, while estimating its relevance by crossing a borderline that the others did not. However, its innovations had some important side-effects when compared to the other projects previously described, since they are just the corner stone of a new and not standardized type of estimation of the opinion relevance. First, its formalizations are not polished as some of the other projects. Second, many of its parameters for estimating the opinion relevance were directly coupled to its domain, thus making it

hard to be applied on other social networks. Models proposed by other projects usually are much more generic and easy to apply on others domains. Furthermore, the researchers had to develop their own corpus, since the previous corpora did not offer enough information to compute all the parameters that they have idealized. Also, their experiments were not well rounded as most of the others projects. They did not compute some of basic information evaluation retrieval metrics, their relevance judgments had a low coverage compared the TREC corpus. They did not explore the evaluation metric created for incompleteness (low coverage corpus). Finally, and most importantly, they were not able to provide ways to customize the proposed model to the opinion consumer. Therefore, their approach to the opinion relevance estimation does not meet the four requirements we have proposed in chapter 2, section 2. The lack of customization of this model is the most important evidence of its inability of matching the requirements we defined for a model to be labeled as Social. Therefore, we classified their approach as “partially social” as we also discussed on chapter 2, section 2.

Looking specifically into the corpus that these projects used, we can see that while the TREC corpus represented a quality and standardized source of data for the researchers, it was also a limitation factor for the projects that used it. Basically, the TREC corpus was designed to rank documents (blog posts) and not opinions. Then, together with documents and relevance judgments, TREC corpus did not provide information regarding social aspects, such as the profile of the people that judged the opinions as relevant or not. This is the reason why we could not find in-depth studies on the customization of the opinion relevance and only one and very recent project explored the meta-data and some social aspects of an opinion. Even if we look into the other available “opinion corpus” to perform experiments concerning opinion relevance, besides the TREC corpus, we will find similar limitations. Corpora like the Wal-Mart, MPQA Opinion Corpus, NTCIR 7 and TREC 2010 Entity Track do not provide information about the profile of neither its author nor about the opinion consumer. Furthermore, as their test cases (i.e. topics) were created to evaluate documents relevance, the experiments had their performance purely based on relevance metrics computed by ranking documents.

3.4. Conclusions

In this chapter, we made a basic introduction of the Opinion Mining Research Field. We also described some of the opinion corpus available at the Internet, and then made an in-depth study of the concept of Opinion Relevance relating its state-of-art evolution and current limitations.

Given the context of our project, the first relevant difference is about our objectives when compared to most of the projects we could analyze: we want to rank opinions, while they wanted to rank documents. Thereby, let us look into the “Just Dance” review example once again. We aim to consider every single opinion about the subject as a single entity, and then generate a score for each one of them. However, most of the related work would handle each review (document with many opinions and facts about the subject) as a single entity and then generate a score for it based on the number of opinions on it. The objective of our model is to rank opinions, so all parameters that we consider are focused on the estimation of the opinions relevance. On the other hand, other projects aiming to rank documents use other concepts like topic relevance. So, the score computed by our model is exclusively linked to opinion’s relevance, but the scores of other projects use opinion’s relevance estimation as a parameter of a document relevance estimation function. The only exception to this was the project described in (Luo, Osborne, & Wang, 2013), which has its own limitations, as we addressed before.

Another important difference between our objective and the previous models in the literature is the possibility to customize results. We believe that Social aspects (Morris, Teevan, & Panovich, 2010a) are a key to handle different kinds of users, just like on Information Retrieval systems. When it comes to opinions, it is expected that the same opinion can be relevant for a user A, while it cannot be relevant for a user B. In these models, it was impossible to provide different results for the same query. However, we want to use concepts inspired in Social Search, where we can allow customized results according to the user profile. For example, a sentence written by an author A can be on the top 10 best scored opinions for a user U1, because A has a profile similar to U1. However, for a user U2, the same sentence cannot achieve the top 10 because B has not a profile similar to U1.

Finally, the different projects found in the literature developed their experiments using corpora composed mainly by blog posts, without the selection of a specific domain (for example, blogs about books). We believe that final result of a score function can vary according to its domain. However, when it comes to the evaluation metrics, the related work has an important contribution to our objectives because they showed to us that it is possible to use information retrieval techniques in order to evaluate opinion mining models. Therefore, we are able to apply traditional metrics like MAP to measure the results of our experiments. Hence, what we need it to adapt and customize them for the context of our project and the corpus used on the experiments.

4 Social Opinion Relevance Model (SORM)

This chapter presents the Social Opinion Relevance Model (SORM). SORM is an abstract and domain independent model that aims to estimate the relevance of an opinion to a consumer (person). SORM offers twelve distinct parameters, that were defined based on a literature review of distinct research fields, discussed in Chapter 2. Since the importance of each parameter is expected to change according to the domain being considered, SORM offers distinct ways to combine and to weight them. By using both the consumer and the author profiles, SORM enables customized results. Furthermore, it also provides ways to evaluate the content of an opinion (its sentence) and its meta-data (i.e., its publishing date).

The remaining of this chapter is organized as follows: section 4.1 formalizes a few important concepts that are necessary to comprehend how SORM parameters are defined and computed; section 4.2 introduces a formal way to classify the multiples aspects of an opinion relevance model; section 4.3 presents a brief discussion about SORM's parameter categories; then section 4.4 formalizes and discuss all SORM parameters; section 0 contains a discussion about how to weight and combine the parameters in our model; section 4.6 addresses a study we made to verify the feasibility of instantiating SORM on distinct social networks; finally, section 4.7 presents the conclusions and the link to the rest of the thesis.

4.1 Basic Formal Definitions

Before detailing our model, we need to introduce some basic concepts. They are needed to properly understand our problem, our model and its formalization. These concepts are the following: (1) Feature; (2) Interest; (3) User; (4) Opinion. In our formal model, capital letters denote sets. Some of the definitions may have elements in common **Definition 3.1**. For instance, similarly to **Definition 3.1**, the definition of an opinion in our model contains a date to represent when it was published. Furthermore, the nomenclature adopted in this chapter is also slightly distinct, so the name of concepts in our model can closer to ones presents on Social Networks.

Definition 4.1 – Feature: a string that always semantically means a characteristic of a domain. For instance, given the domain of books, two of its features are “story” (which is related to the book’s narrative) and “writing” (which is related to how the book was written);

Definition 4.2 – Interest: string representing a concept or a subject that a person has interest in. For instance, an interest may be the name of a book (e.g., “Odyssey”) or the name of a movie (e.g., “Titanic”);

Definition 4.3 – User: an ordered quadruple $us = <na, FR, IN, ag>$, where na is a string representing the name of the user; FR is a set of users representing the friends of the user in a social network; IN is the set of interests of the user; and ag is a positive natural number representing the age of the user;

Definition 4.4 – Opinion: an ordered quintuple $op = <in, au, se, FE, da>$, where in is the interest that the opinion is about; au is a user representing the author of the opinion; se is a string representing the sentence where the opinion is expressed; FE is the set of features that the sentence se refers to; and da is the date when the opinion was published.

Given a Social Network, we call U the set of all its users and O the set of opinions written on the network by the elements in U. Given an opinion op, we will refer to the user that has produced it as “author”, and we will refer the user whom the opinion relevance is being estimated for as “consumer”.

As an example, let $us1$ be a user such that $us1 = <$ *“John”, { “Mary”, “Carl” }, { “Odyssey”, “Hamlet” }, 25* $>$. Hence, $us1$ represents a user called John who has Mary and Carl on the set of his friends; Odyssey and Hamlet are two books on the set of his interests; and he is 25 years old. Now let $op1$ be an opinion, such that $op1 = <$ *“Odyssey”, ”John”, “The book has a very impressive plot”, { “story” }, “December 25th, 2013”*$>$. In this new sample, $op1$ is an opinion written by the user John (the same user of the first example) about the book Odyssey; its sentence refers to the story of the book and it was written on December 25th, 2013.

4.2 Three Aspects of the Opinion Relevance Concept

The most basic way to define Opinion Relevance is through a function that, given an opinion, estimates its relevance by returning a natural number. Therefore, we suppose that the relevance estimation is independent of the person whom it is been estimated for. We present a formalization for this intuitive concept in Definition 4.5.

Definition 4.5 – Opinion Relevance Function: given a set of opinions O, a function OR: O→ℝ. The semantic of the value computed by OR is the estimation of an opinion's relevance.

One way to improve this definition is by allowing the result to be *customizable.* This can be achieved with the introduction of a new parameter in the opinion relevance function: *the opinion consumer*. The opinion consumer is the person whom the opinion relevance is being estimated for. Given this new parameter, the function is now able to take into account many of the consumer's characteristics. For instance, the function may be able to take into consideration the age of person on even its country when estimating the opinion relevance. This intuitive notion is formalized in Definition 4.6.

Definition 4.6 – Customized Opinion Relevance Function: given a set of opinions O and a set of users U, a function COR: OxU→ℝ. The semantic of the value computed by COR is the estimation of an opinion's relevance for a specific user.

We believe that we can still evolve the Opinion Relevance concept by adding a new parameter to its definition. This new parameter is the *profile of the author of the opinion*. Assuming that both the opinion author and the consumer are members of a social network, both can be represented by the same type of entity: users. Therefore, the estimation of opinion relevance can explore new issues. For instance, now it is possible to know if the author and the user are of the same age group or even from the same country, while estimating the relevance of an opinion. We will use the word Social to describe this new type of opinion relevance, and it is formalized in Definition 3.4:

Definition 4.7 – Social Opinion Relevance Function: given a set of opinions O and a set of users U from a social network, a function SOR: OxUxU→ℝ. The semantic of

the value computed by SOR is the estimation of an opinion's relevance for a specific user (the opinion consumer) using also information about the author of the opinion.

4.3 Parameters Categorization

Each parameter on our model has its own formal definition. The definitions are usually expressed by equations, showing how to compute them. These equations make explicit the inputs needed to calculate the parameters. Based on the inputs, we can classify SORM parameters into two distinct types: the social/non-social categorization and the customizable/non-customizable categorization, as discussed on chapter 2, section 2.

Non-social parameters are the ones that can be computed without using the author and the consumer. Social, conversely, are the parameters that make use of the author of the opinion or the consumer on its equation. Essentially, non-social parameters are the paradigm followed by most related work, except (Luo, Osborne, & Wang, 2013). Meanwhile, social parameters are only computable if we have a social network available relating opinions and users (authors or consumers).

An alternative view is the categorization of the parameters onto customizable and non-customizable ones. Customizable parameters are those that may change according to the consumer. In other words, they are the parameters that have the consumers on their equations. Non-customizable parameters are the ones that do not use the consumer on their equations.

4.4 Parameters Formalization and Discussion

In this section, we formalize each parameter of our model. However, before the formalization we describe the meaning of each parameter, which is important for the understanding of the motivations behind them. In the following equations: op is an opinion; the function $sentence(op)$ returns the set of words in the sentence of an opinion op; au is a user representing the author of an opinion; and co is the opinion consumer.

Furthermore, notice that equations presented in this section are relatively simple and in most cases do not contain measures to prevent their value to grow indefinitely. This is a limitation that is addressed on section 0. For instance, in some of the parameters we could add a logarithm to the members of their equation to avoid

the final result to grow. However, we believe that this decision is domain specific. Hence, it should made only when the model it being instantiated to a domain.

4.4.1 Author Experience

This parameter represents how experienced the author of an opinion is when compared to the other authors of opinions in O. The more opinions the author expressed, the higher the value of this parameter. Formally, given the subset of O in which au is the author of the opinions, Oau, the Author Experience is shown in Equation 4.1.

$$Author\ Experience(au) = \frac{|O_{au}|}{|O|} \tag{4.1}$$

4.4.2 Feature Experience

It represents how experienced the author of an opinion op is about a feature f. The more opinions about f he has expressed, the more relevant are his opinions about f. Intuitively, it means that an author with many opinions about a given feature may be a specialist on it. Formally, given a feature f and an author au; let Of be a subset of O such that each opinion $op \in Of$ refers on its sentence se to the feature f; and let $Oauf$ be the subset of Of such that each opinion $op \in Oauf$ has au as its author: hence, the Feature Experience is defined in Equation 4.2.

$$Feature\ Experience(au, f) = \frac{|O_{auf}|}{|O_f|} \tag{4.2}$$

4.4.3 Opinion Inverse World Frequency (OIWF)

This parameter was inspired by the Inverse Document Frequency (IDF) (Jones, 1972) metric from Information Retrieval, presented in section 2.1.2. Intuitively, it means that the rarer the words in an opinion sentence are, the more relevant the opinion is. This parameter, shown in Equation 4.3, is computed by the sum of the inverse frequency of each word of the *sentence*(op).

$$OIWF(op) = \sum_{w \in sentence(op)} log\left(\frac{|O|}{|O_w|}\right)$$
$$where\ O_w = \{\, o \mid o \in O \ \wedge\ w \in sentence(o) \,\} \tag{4.3}$$

4.4.4 Feature Score

Feature Scores follows a very straightforward heuristics: the more features an opinion refers to, the more relevant it is. Intuitively, the more features an opinion talks about the richer its text is. Formally, given the set of features F present in O, and the set of features that the opinion op refers on its sentence, Fop, the Feature Score is defined in Equation 4.4.

$$Feature\ Score\ (op) = \frac{|F_{op}|}{|F|} \tag{4.4}$$

4.4.5 Linear Longevity

This parameter can be interpreted as: the newer an opinion is, the more relevant it is. In some domains, the relevance of an opinion may change as the time evolves. For instance, a notebook processor may be cutting edge when launched, but it may already be outdated a year after. Consequently, old opinions may be outdated as well. Formally, given the functions *date(op)* that returns the date of an opinion op and *currentDate()* a function that returns the present date, the Linear Longevity is expressed in Equation 4.5.

$$Linear\ Longevity(op) = \frac{date(op)}{currentDate()} \tag{4.5}$$

4.4.6 Network Distance

Network Distance is the shortest path (distance) between two users (an author and an opinion consumer) in a Social Network graph. The closer they are, the more relevant the author's opinions are. This parameter follows the heuristic that opinions from friends are more relevant than opinions from other people. Formally, given a directed graph g where the users in U are its nodes and its edges link two users that are friends in a Social Network, i.e. there is an edge from the node a to the node b, if and only if $b \in friends(a)$, where *friends(u)* is a function that returns the set of friends of the user u, the Network Distance is shown in Equation 4.6.

$$\begin{gathered} Network\ Distance: UxU \rightarrow \mathbb{N} \\ The\ shortest\ path\ between\ two\ Users\ in\ the\ graph\ g \end{gathered} \tag{4.6}$$

4.4.7 Similarity

Similarity is an estimation of the amount of interests in common between an author and an opinion consumer. The more they have in common, the more relevant the opinions of an author are for a consumer. Formally, given the function *interests(u)* that returns the set of interest of a user u, this parameter is modeled in Equation 4.7.

$$Similarity(au, co) = \frac{|interests(au) \cap interests(co)|}{|interests(co)|} \quad (4.7)$$

4.4.8 Image and Reputation

Inspired by the multi-agent Repage reputation model proposed by (Sabater, Paolucci, & Conte, 1996), presented in section 2.4, we adopted two different concepts, Image and Reputation, as distinct parameters. In our model, Image means how a user perceives another one based on past direct experiences, while Reputation means how a user is viewed by all other users in U. Intuitively, we can estimate both by looking into the number of opinions previously judged as relevant by the users in U. Formally, given the set of opinions from the author au judged as relevant previously by the opinion consumer co, *ROau-co*; and the set of opinions from au that were evaluated (as relevant or not) by co, *JOau-co*, these parameters are defined in Equation 4.8.

$$Image(au, co) = \frac{|RO_{au-co}|}{|JO_{au-co}|} \ (a)$$
$$Reputation(au) = \frac{\sum_{co \in U} Image(au, co)}{|U|} (b) \quad (4.8)$$

4.4.9 Positive Orientation and Negative Orientation

These parameters reflect the estimations of how positive and how negative an opinion is. Both can be computed by using optioned word dictionaries like the SentiWordNet (Baccianella, Esuli, & Sebastiani, 2010). Intuitively, each orientation is the sum of positive or negative weight of each word on the sentence of an opinion. Instead of unifying them into a single generic view of orientation, we decided to compute both separately because intuitively they may have distinct weights. By doing this, we can analyze how each of them is important for the model in a given domain. Formally, given the functions *positiveWeight(w)* and *negativeWeight(w)* that return

the positive real number representing the orientation of a word w, we can model these parameters like shown in Equation 4.9.

$$Positive\ Orientation(op) = \frac{\sum_{w \in Sentence(op)} positiveWeigth(w)}{|sentence(op)|} \quad \text{(a)}$$

$$Negative\ Orientation(op) = \frac{\sum_{w \in Sentence(op)} negativeWeigth(w)}{|sentence(op)|} \quad \text{(b)} \qquad (4.9)$$

4.4.10 Age Similarity

Age Similarity is the difference of age between an author and an opinion consumer. Intuitively, the smaller the difference is, the more relevant the author is. For instance, we expect that people from the same generation (around the same age) think similarly. Let $age(u)$ be a function that returns a number representing the age of the user u in years; and k be a constant defined according to the domain. If au and co have the same age the result must be 1, however if the difference between their ages is equal or larger than k the similarity is 0. The Age Similarity is described in Equation 4.10.

$$AgeSimilarity(au, co) = \max\left(0,\ 1 - \left(\frac{|age(au) - age(co)|}{k}\right)\right) \qquad (4.10)$$

4.4.11 Categorization

Table 4.1 shows how SORM parameters are classified according to the social/non-social categorization and to the customizable/non-customizable categorization. As discussed before, the categorization was based on the inputs of the equation of each parameter.

Table 4.1 – Classification of SORM parameters.

Parameters	Customizable	Social
Author Experience, Feature Experience	no	yes
OIWF, Feature Score, Linear Longevity, Positive Orientation, Negative Orientation	no	no
Network Distance, Similarity, Image, Reputation, Age Similarity	yes	yes

Parameters that are both social and customizable are the core of our model. They represent our main distinction and innovation when compared to the models found in the literature, as shown in Table 4.2. However, since we also recognize the other parameters as important ones, SORM provides non-customizable and non-

social parameters to complement the model, making it more robust. Notice that the categorization showed here respects the discussion about social document ranking models we have made in section 2.2.2.

Table 4.2 – Comparing SORM classification with the related work.

Model	Parameters	
	Social	Customizable
(Mishne, 2006) (Zhang & Ye, 2008) (Huang & Croft, 2009) (Orimaye, Alhashmi, & Siew, 2012a) (Xu, et al., 2012) (Orimaye, Alhashmi, & Siew, 2012b)	no	no
(Luo, Osborne, & Wang, 2013)	partially	no
SORM	yes	yes

4.5 Parameters Weighting and Combination

Combining and weighting the parameters is one of SORM's most import features. While defining exactly how to perform the combination and the weighting would make SORM less flexible, we can still define what types of weighting and combination the model supports. Furthermore, the exactly weight and combination is expected to vary according to the domain where the model will be instantiated.

Let x be a parameter of our model and a and b be two constants, we can compute x's weight in the following ways:

- Multiplication: $weight(x) = a.x;$
- Division: $weight(x) = a/x;$
- Logarithm: $weight(x) = a.logb(x+1),$ notice that the $"+1"$ on the equation ensures that result is greater or equals to zero;
- Exponential: $weight(x) = a.xb$.

Notice that the exponential weight with b = 1 is similar to the multiplication weight; however, since the multiplication weight is the base of the linear combination of two parameters, which is used by many projects in the literature, we decided to keep explicitly both of them in our model.

Let x and y be two distinct parameters of the model and let weight(x) and weight(y) be their respective weights. On the experiments x and y can be combined in two distinct ways:

- Sum: *weight(x) + weight(y)*;
- Multiplication: *weight(x).weight(y)*.

Since our model has more than two parameters their combination can be made by generalizing these two options. For instance, let us say we want to combine three parameters *x*, *y*, and *z*. A sample of valid way of combining them is: *weight(x) + (weight(y) * weight(z))*;

The way the parameters will be combined or composed can be learned from examples. Thus, in order to instantiate SORM on a domain it is necessary to make use of a Social Opinion Relevance Corpus on the same domain. The leaning algorithm used to optimize can follow a generic optimization approach or one specific for learning ranking functions. The decision of which learning strategies should be used to balance the model should be made according to the domain where the model is being instantiated.

4.6 Study and Discussion about SORM's Domain Independence

SORM was created to be domain independent. In fact, this is one of its most important features. However, we believe that the generic and formal definitions of its parameters are not enough to prove it. Hence, after the definition of SORM parameters, we performed a small feasibility study in order to evaluate how each of SORM's parameters could me computed as if it they were to be instantiated on distinct social networks on the internet.

To perform this study we selected three distinct social networks: (1) Facebook, one of the most popular general purpose social networks on the internet; (2) Twitter, another popular general purpose social network, but it this case with minimalist

features; (3) My Anime List[8], a domain specific social network, focusing on Animes and Mangas.

During this study we decided to focus on how feasible is the computation of each SORM parameter from the perspective of the owner of the social network. This means that, in theory, we would have access to all the information that a social network stores about its users. Using this information, we try to find what among the date stored by the social network is useful while computing the SORM parameters.

Furthermore, we also admit the existence of an Ontology of Features and a function capable of mapping opinions on member of this ontology (i.e. mapping opinions on features) for each social network. We believe that this assumption is reasonable given that both can be constructed using knowledge from domain specialists.

4.6.1 Facebook

Facebook is a general purpose social network on the internet. The profile of its users can have plenty of information, such as their age, friends, and interests. Users can express their opinions on their time line (which is similar with a micro blog) and also on pages and communities created by other users. Figure 1.1 shows one type of information available on the profile of their users, in this case, interests on the domain of music.

[8] http://myanimelist.net/

Figure 4.1 – Samples of interests of a Facebook profile on the Music domain.

While evaluating the feasibility of computing SORM parameters using the information available on Facebook, we reached in the following conclusions:

Author Experience: It can be estimated using the number opinions of an author about the same domain of the opinion. We retrieve all of his/her opinions published on Facebook about a specific domain. Then we can compute how experienced he/she is in this domain. On Facebook users can express their opinions in distinct ways like on its own profile, on its timeline, in page, and in a community. Therefore, collecting all opinions from a user may not seem to an easy task. However, since we are looking to this problem from the Facebook's owner perspective, this task is expected to be easier than it looks since the databases of the social network probably have indexed all the information published by a user.

Feature Experience: It can be estimated using the number of opinions of an author about the same domain of the opinion and a specific feature of that domain. The set of opinions from a user necessary to compute this parameter is a subset of the one presented on the Author Experience. Once we retrieve all the opinions on a

domain from an author on Facebook, we just have to filter the ones about a feature to compute this parameter.

OIWF: It depends only on the text of the opinion, so it can be computed directly for each opinion on the Facebook.

Feature Score: Number of features of a given domain, which an opinion refers to can also be computed from the opinion text and using a feature ontology. Therefore, like OIWF this parameter can be computed without major problems.

Linear Longevity: This parameter can be computed using the date and time that an opinion was published. Facebook stores and shows this information on each post (which includes the opinions) published on the site. Consequently, this parameter can be computed using the data present on the site.

Network Distance: Facebook supports the concept of friends. Each user on the social network has its own friends. Using this information, we can construct the graph necessary to compute this parameter. Hence, using the graph, this parameter can be computed on Facebook too.

Similarity: Facebook shows on the profile of its users a list of the interests that they had linked to their profiles (Figure 4.1). Let us imagine we want to compute the similarity among two Facebook users on the domain of music. Using the sets interests about music on both profiles we can use the insertion between both sets to estimate this parameter.

Image and Reputation: Both can be estimated from an Opinion Relevance Corpus, a subject addressed on chapter 5. On Facebook, users can judge opinions by performing an action called “like”. If a user finds an opinion published by another user relevant, he/she can add it to the set of these opinions that he/she likes. This information is the corner stone of an Opinion Relevance Corpus. Using it, we can know how many people like the information of a user, and how many opinions from an author a user likes. Both are needed to compute Reputation and Image, respectively.

Positive and Negative Orientation: It can be estimated by using dictionaries of opinion related terms such as SentWorldNet. Given an opinion, we want to

estimate the Positive and Negative Orientations we just need to use the dictionary chosen to find how positive or negative is each word on the opinion.

Age Similarity: Facebook stores information regarding the age of the users on their profiles. One of the information asked to the users when they are creating their profile on the network is their birthday. Using this information this parameter can be computed directly.

Therefore, 12 out of 12 SORM parameters can be computed using Facebook data.

4.6.2 Twitter

Twitter is also a general purpose social network. However, unlike Facebook, it is a minimalist network. This means that it stores less data about its users, and what users publish on the site has some restrictions related to the size of its content. As shown in Figure 4.2, there is no age, interests or friends. A twitter profile is composed mainly by the people that a user follows, the users that follow him and the small texts that the user published.

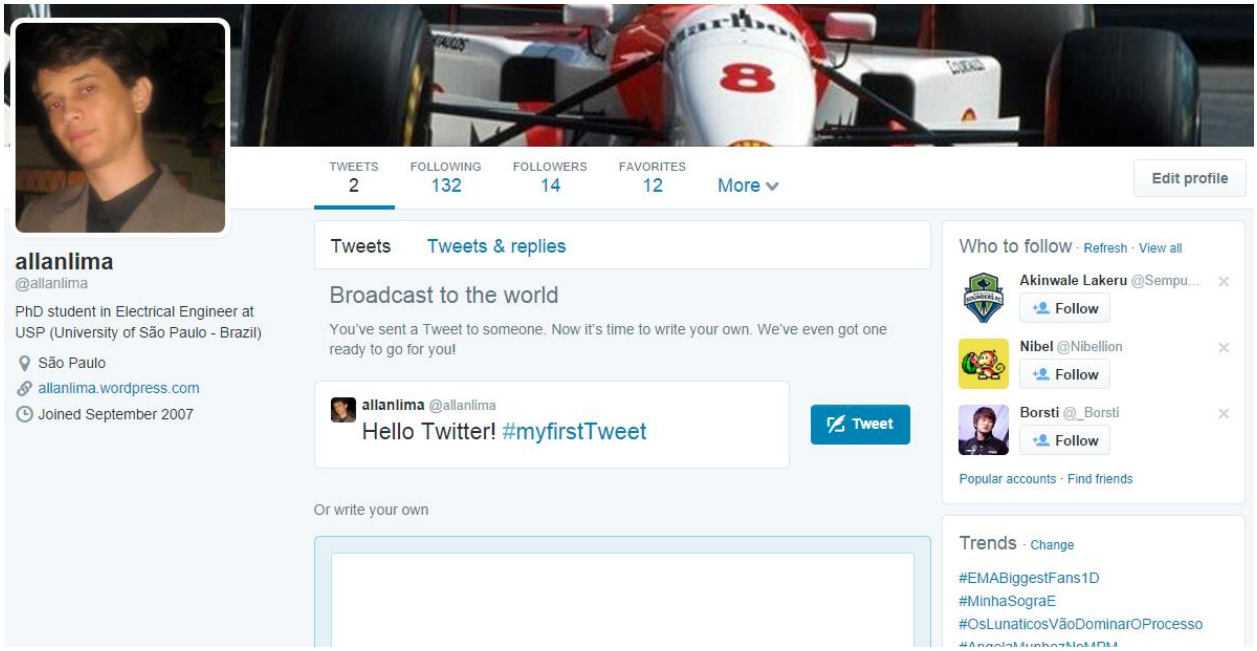


Figure 4.2 – A typical Twitter profile.

After evaluating the feasibility of computing SORM parameters using the information available on this network, we reached in the following conclusions:

Author Experience: Based on the number of twits of an author about the same domain of the opinion. Twitter associates to its users' profile all their publications on the social network. It is possible to filter all their opinions using an opinion classifier and it is also possible to filter among them the opinions that are about a given subject by using a topic relevance function. Hence we can estimate how expert an author is about a certain subject.

Feature Experience: Based on the number of twits of an author about the same domain of the opinion and a specific feature of that domain. In order to compute this parameter, we need to filter all the opinions from an author about a subject feature. Once we have these opinions filtered we can estimate the Feature Experience of an author.

OIWF: It depends only on the text of the tweet, so it can be computed directly from the words on the tweet.

Feature Score: Number of features of given a domain that a tweet refers to. Assuming that we have an ontology that allows us find the features that a tweet refers to, we just need to count them to compute this parameter.

Linear Longevity: Date and time when a tweet was published. Twitter stores this information and it is shown on each tweet published on the site. Therefore, we can use the date associated to each tweet in order to compute this parameter.

Network Distance: The graph of users needed to compute this parameter can be created using the feature of the service called followers. Each user has its followers and is followed by other users. Therefore, we can create a directed graph in which the users are the nodes, and there is an edge from a user A to a use B if A follows B.

Similarity: Unlike Facebook, there is no list of interests associated with the profile of users from twitter. So, this parameter cannot be computed properly.

Image and Reputation: Similarly to Facebook, both can be estimated from an Opinion Relevance Corpus. The relevance judgments on the corpus can be used to estimate the image and the reputation of an author in a similar fashion we addressed while analyzing how to compute this parameter on Facebook.

Positive and Negative Orientation: Similarly to Facebook, this parameter can be estimated by using dictionaries of opinion related terms such as SentWorldNet.

Age Similarity: Twitter does not provide information regarding the age of the users on their profiles. Hence, this parameter cannot be computed properly.

Therefore, 10 out of 12 SORM parameters can the computed using Twitter data.

4.6.3 My Anime List

My Anime List is social network focused on the domain of Animes (Japanese cartoon) and Manga (Japanese comic books). They offer to their users the possibility to discuss Animes and Mangas with others users and also review both types of media. A typical profile (as shown in Figure 4.3) on this network is as rich in information as the ones on Facebook, containing links to its posts on forums, its reviews and the Animes and Mangas that the user has interest on.

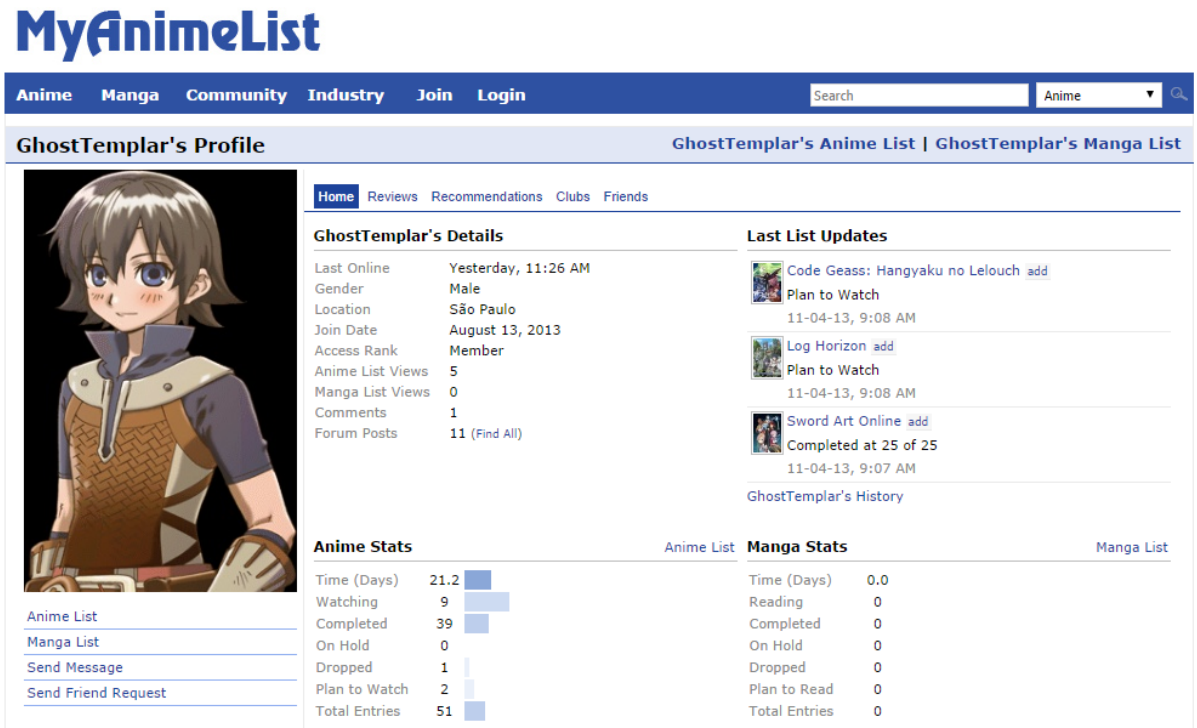


Figure 4.3 – Profile of a user on the social network My Anime List.

While evaluating the feasibility of computing SORM parameters using the information available on My Anime List, we reached in the following conclusions:

Author Experience: Can be estimated using the number opinions that an author expressed on the reviews which he/she wrote. My Anime List allows its users to write reviews about the amines presents on the site's database. If we want to estimate the experience of an author, we can use the opinions he expressed in his/her reviews.

Feature Experience: Similarly to Facebook and Twitter, it can be estimated using the number opinions about a feature of an author expressed on the reviews which it wrote. However, in this case, the source of the opinions is the reviews that the users published on the site.

OIWF: Once again, this parameter depends only on the text of the opinion, so it can be computed directly. Once we filter the opinions, we just need count the occurrence of its words to be able to estimate their OIWF.

Feature Score: Like Facebook and Tweeter this parameter can be estimated using the number of features of given a domain that the opinions from an author refer to.

Linear Longevity: My Anime List displays the date and time when each review published on the site. Since opinions are extracted from the reviews, we can use this information for all opinions in a review to compute their Linear Longevity.

Network Distance: My Anime List supports the concept of friends. Consequently, this parameter can be computed in a similar way as discussed while Facebook was evaluated. This means that we can use the friendship information available on the profiles of the users on the network to create the graph necessary to compute this parameter.

Similarity: My Anime List shows on theirs users profile lists of the Animes and Mangas that they had linked to their profiles. So a similarity measure among two users of the network can be computed using this data. For any two users of the network, we need to compute the intersection of their interests to estimate this

parameter. This estimation can be made likewise the one we described while studying this parameter feasibility on Facebook.

Image and Reputation: Both can be estimated from an Opinion Relevance Corpus. The relevance judgments on the corpus can be used to estimate the image and the reputation of an author. Once more, there is no difference in the way these both parameters could be estimated on My Anime List compared to the previous networks we discussed so far.

Positive and Negative Orientation: It can be estimated by using dictionaries of opinion related terms such as SentWorldNet, similarly to Facebook and Twitter

Age Similarity: My Anime List does not ask information regarding the age of the users. Hence, this parameter cannot be computed due the lack of this information on the social network.

Therefore, 11 out of 12 SORM parameters can be computed using My Anime List data.

4.6.4 Discussion

The results of our study are summarized in Table 4.3, at the end of our study we could find that most of SORM parameters can be computed on the social networks we evaluated. Given the sensible difference in focus, features and scope between those networks we can now be more confident of SORM's independence of domain. Furthermore, this conclusion is complemented by the fact that none of those networks was used on our experiments. This means that we evaluated the feasibility of computing SORM parameters on another network before performing our experiment, as we address on chapters 5 and 6.

Table 4.3 – Summary our study about computing SORM parameters on three distinct social networks on the Internet.

Parameter	Social Network		
	Facebook	Twitter	My Anime List
Author Experience	yes	yes	yes
Feature Experience	yes	yes	yes
OIWF	yes	yes	yes
Feature Score	yes	yes	yes
Linear Longevity	yes	yes	yes
Network Distance	yes	yes	yes
Similarity	yes	no	yes
Image	yes	yes	yes
Reputation	yes	yes	yes

Positive Orientation	yes	yes	yes
Negative Orientation	yes	yes	yes
Age Similarity	yes	no	no
Total	12 (out of 12)	10 (out of 12)	11 (out of 12)

4.7 Conclusions

In this chapter we introduced SORM, a Social Opinion Relevance Model that was conceived to be domain independent. Each of SORM's parameters was discussed, formalized using mathematical equations, and classified as customizable/non-customizable and social/non-social. Furthermore, we also discussed how to create an opinion ranking function and evaluated if SORM parameters could be computed on distinct social networks on the Internet.

We are now able to explain in details how it is possible to instantiate SORM on a specific domain. In the next chapter, we will introduce a methodology to translate, balance and test SORM parameters on a domain. In the sequence, we will show how we have instantiated our model on the electronic games domain.

5 Social Opinion Relevance Corpus (SORC)

This chapter describes the Social Opinion Relevance corpus (SORC), which is a key element to understand how to translate and test our domain independent model in a specific domain. However, before discussing our corpus important steps must be addressed. The first step is to discuss how social aspects can be included in an opinion relevance corpus. The second one is to investigate how to evaluate a social opinion relevance model based on such a social opinion relevance corpus. Then, we can finally introduce the Social Opinion Relevance Corpus (SORC).

SORC is a customized corpus created specifically for testing opinion relevance models. In order to achieve this goal, SORC provides all the necessary information to compute social and non-social parameters. As explained in Chapter 4, SORM contains metadata (such as the opinion's publishing date) besides the profiles of both the author and the consumer of the opinions. Given the limitations of the currently available corpora, SORC had to be created. The domain we have chosen to populate SORC was electronic games. This decision was made mainly because there is a reasonable amount of social networks on the Internet on this domain. Hence, this corpus, called SORC-Games01, is the first SORC corpus we have created.

This chapter is organized as follows: on section 5.1, we explain how to create a social opinion relevance corpus; section 5.2 describes how we can evaluate a social opinion relevance model using such a corpus; in section 5.3 we introduce the corpus we have created for our experiments; finally, in section 0 we present our conclusion of the chapter.

5.1 Social Aspects on an Opinion Relevance Corpus

An Opinion Relevance Corpus can be designed for a domain specific evaluation. For example, it can be composed by opinions about books. However, it can also consist of a multi-domain collection, for example it can be composed by opinions from customers of an on-line shop which may include opinions about electronic appliances, movies, electronic games, furniture etc. The corpus composition will have to match the purposes of the researchers.

The introduction of social aspects into an opinion relevance corpus is fundamental for testing any social opinion relevance model such as SORM. Originally, any opinion relevance corpus must provide relevance judgments of opinions. However, since we want to introduce social information inside the corpus we must provide with it not only with relevance judgments of opinions, but also with information about the judges (that played the role of opinion consumers) and the authors of such opinions. Such information will allow social relevance models to compute their parameters properly and even to provide customized results.

We can formalize the intuitive notion discussed so far as the definition of a Social Opinion Relevance Corpus:

Definition 5.1 – Social Opinion Relevance Corpus: a quintuple $<OP, US, IN, JU, TC>$ such that

1. OP is a set of opinions, where $OP \subset O$;
2. US is a set of users, where $US \subset U$;
3. IN is a set of interests, expressible as queries (a query is a string that semantically represent an information need);
4. JU is a set of Opinion Relevance Judgments, each $ju \in JU$ is a quadruple $ju = <qu, op, co, re>$, where
   4.1. qu is a query;
   4.2. op is an opinion $(op \subset OP)$;
   4.3. co is a user that is the opinion consumer $(co \subset US)$; and
   4.4. re is the relevance of op to co.
5. TC is a set of Test Cases, where each $tc \in TC$ is a quadruple $ts=<qu, OT, co, JU>$ where
   5.1. qu is a query;
   5.2. $OT \subset OP$ represents a set of opinions that are relevant to qu;
   5.3. $co \in OP$ represents an opinion consumer; and
   5.4. JU is a set of Opinion Relevance Judgments such that for each $ju \in JU$, co is its user and qu is its query.

Intuitively, a Test Case aggregates all the relevance judgments made by a consumer for a given query.

These elements allow us to test models by looking if they are able to predict the relevant opinions given a consumer and a query.

So far, we have addressed what we considered as a social opinion relevance corpus. However, in order to create such a corpus one needs to follow a methodology to avoid well-known problems, such as bias. In this sub-section, we will describe the methodology developed to create social opinion relevance corpora.

Our methodology is based on Information Retrieval strategies to compose relevance corpus, like the TREC Methodology (Aslam, Pavlu, & Yilmaz, 2006; NIST-TREC, 2013) and it is composed of five main steps:

1) **Domain Definition:** based in a feasibility and relevance study. Given a domain, we would like to know if there is data available (in practice: reviews, social networks, etc.); also we would like to know if the there are other projects in the same domain in the opinion mining arena. If the domain was not studied yet by opinion mining researchers, and it has also plenty of data available in the Internet, then this may be an indication a of new area to be explored;

2) **Scope and Sources Selection:** definition of the scope of the data to be collected. For example, given a set of Social Networks available for a given domain at the Internet, which of them should be chosen to collect the data? This decision must be based in a series of factors and constraints. Usually, it is expensive and time consuming to collect data from all the available sources, but it is possible to choose those that cover most of the topics of a domain and, at the same time, respect the resource limitation of the researchers;

3) **Opinion Crawling**: The process of converting non-structured web pages in structured data. This is made by a parser that should be able to read the web pages and extract information like the publishing date and the author of the text;

4) **Opinion Extraction:** The process of selecting, among all the sentences collected in the previous phase, only those sentences that are opinions. This step can be accomplished by a machine learning classifier, as we addressed in chapter 2;

5) **Relevance Judgments:** Judgments are created by polling users from the social networks chosen in step two, playing the role of opinions' sources. In this step, users are asked to judge the relevance of opinions about a given subject.

On section 5.3, we will show how this abstract methodology was applied to create an opinion corpus that meets the necessary requirements to balance and test SORM.

5.2 Evaluation of Social Opinion Relevance Models

The methodology created to evaluate opinion relevance models was developed with the goal of testing SORM in our experiments. The main objective of this methodology is to allow us to compare different opinion relevance models, using the performance of Information Retrieval evaluation metrics as the parameters of the comparison. Our methodology was created with an important thing in mind:

Be coherent from a mathematical and statistical point of view: our evaluation methodology has to be able not only to provide mechanisms to compare distinct opinion relevance models, but also to provide ways to know whether the comparison is significant. Information Retrieval has the mathematical tools (metrics) to compare different models, and Statistics has the mechanisms to allow us to know how significant the comparison was;

We have segmented our evaluation methodology in three important steps:

1) **Corpus selection:** the definition of the corpus(es) to be used on the experiments, if we want to test a social relevance corpus it is natural for us to select a corpus with social features;

2) **Evaluation Metrics:** given the corpus(es) selected, we will explore its size, coverage and test cases to select the proper information retrieval evaluation metrics to perform tests;

3) **Statistical Significance Test: it** acts as a complement to the tests performed on the previous step, giving confidence that the results found were, in fact, coherent.

Now we will present how each of these steps was concretized during our project.

5.2.1 Social Opinion Relevance Corpus Selection

As mentioned in Section 2.1.4, there are many corpora already published and used for testing document ranking models. Cranfield, TREC and Reuters are some samples among the most popular test sets available at the Internet. Unfortunately, as already explained before, we cannot use any of them to evaluate our model for two major reasons (1) relevance judgments in these corpora were made for the Documents and our model was created to rank Opinions, whose *contents* are usually fragments of a Document; (2) there is no User associated information to the Relevance Judgments in the test sets. As a consequence, we could not use either any of these corpora to perform experiments whit SORM.

As discussed before, the related work most similar to our approach had to create its own corpus to be able to perform its experiments (Luo, Osborne, & Wang, 2013). In a similar way, in order to be able to evaluate models for opinion ranking, we also had to create our own corpus, called the Social Opinion Relevance Corpus (SORC) which we will detail in section 5.3.

5.2.2 Evaluation Metrics

Based on the metrics proposed by Information Retrieval researchers, we chose five distinct parameters to evaluate our model: RPrecision, Means Average Precision (MAP), bpref, Normalized Discounted Cumulative Gain (NDCG) and QMeasure. The main metrics for an evaluation are chosen based on the amount of data available (sentences in our case) and the coverage of the relevance judgments in the corpus. Since it is practically impossible to have all of its opinions judged, we choose NDCG and QMeasure as the main evaluation metrics. However, we also suggest the computation of MAP, RPrecision and bpref during the evaluation, in

order to have a more detailed comparison of the models being evaluated. In particular, we propose the computation of the bpref and RPrecision as well because they also work well on corpora where the coverage of the judgments is not complete (Buckley & Voorhees, 2004; Manning, Raghavan, & Schütze, 2008). On the other hand, MAP is a standard IR metric used by many researchers and present many of the TREC task (Macdonald, Ounis, & Soboroff, 2007; Balog, Serdyukov, & Vries, 2010; Ounis, Macdonald, Lin, & Soboroff, 2011). Finally, the computation of these metrics is also important for the next step of our methodology, when we discuss how a significance test complements the results.

5.2.3 Statistical Significance Test

In order to estimate the Statistical Significance of the values obtained by computing the metrics discussed before, we decided to adopt the Fisher's Randomization Test (Fisher, 1935), as suggested by (Smucker, Allan, & Carterette, 2007) and previously presented in Section 2.1.4.3. In our methodology, we propose the application of this test using all the evaluation metrics prosed, hence we will be able to know if the results provided by each of them are significant or not. For example, if the test using the QMeasure shows that it is not significant, it is better to use the other metrics to decide which model among those being evaluated is the best.

In a nutshell, our methodology can be described by four steps:

1) Select a Social Opinion Relevance Model;
2) Compute the RPrecision, MAP, bpref, NDCG and QMeasure metrics for all the models that we want to evaluate;
3) Perform the Fisher Randomization test for the metrics computed in the second step;
4) Use QMeasure and NDCG to decide which is the best model. This can be achieved by looking into the values of both metrics and also the results on the significance test. If the results are not conclusive , we may look into the results of the other (complementary) metrics .

5.3 Creation of a Social Opinion Relevance Corpus

In an abstract level, a Social Opinion Relevance Corpus (SORC) is an effort to model, design and create opinion corpora, providing opinion relevance judgments

coupled with information about their users and authors. SORC was created by aggregating a set of elements designed especially for the evaluation of opinion ranking models. It is composed primary by subjects, opinions about the subjects, user profiles and relevance information associating opinions and users, following **Definition 5.1**. Despite its standardized format of content, the contents that SORC aggregates may vary in size (amount of content).

On the Internet, there are many distinct social networks available, some of them are very focused on specific domains, such as books or movies. On the other hand, there many other social networks where there is no main subject, such as Twitter and Facebook. Therefore, the SORC's corpora may vary on the domains that they represent.

Currently, the only data sources used in SORC are reviews. In addition, from those reviews we want to extract sentences. During our planning phase, we studied many possibilities of extracting opinions from text. We could use the whole document (i.e. review) as an opinion, but we declined this approach given the fact that a review may contain many opinions as well as many facts. We could also choose to use paragraphs as opinions, but once again a paragraph could be relatively big and consequently have more than one opinion and/or fact. Hence we decided that sentences were the best granularity to work with. A sentence can have more than an opinion, but at the same time when compared to the other granularities (document and paragraph) the chance of this happening is relatively small. On the other hand, a sentence may be too small to contain a whole opinion. To overtake this problem, we excluded sentences with less than 6 words from our corpus, and during the polling phase we also decided to show to the user the previous and the next sentences that are going to be evaluated.

5.3.1 SORC-G01: Games 01

SORC-G01 is the first SORC corpus that we have created. We selected the domain of Electronic Games based on the amount of public available reviews about games on the Internet. Its data were collected from a social network called gamrConnect (http://gamrconnect.vgchartz.com). SORC-G01 has 50 Test Cases (i.e., topics), which is typically considered as a "rule of thumb" on experiments on Information Retrieval (Manning, Raghavan, & Schütze, 2008). More information

regarding the composition of SORC-G01 can be found at Table 5.1. Notice that since the relevance judgments are not complete, i.e. they do not cover all the opinions, experiments using this corpus will require some extra actions, as described in section 5.2.

Table 5.1 – SORC-G01 Composition.

Test Cases	50
Relevance Judgments	489
Distinct Users	15
Coverage of Opinions Judged	~5.49%

5.3.1.1 Creation Methodology

The domain of electronic games was chosen as the target of SORC-G01 for a few important reasons. First, there are plenty of data (reviews) publicly available about electronic games on the Internet. Furthermore, we could find many social networks with reviews about electronic games at the Internet, which means that associated with the reviews there is some social information about people who wrote them.

Among the sources available at the Internet, we selected the social network called gamrConnect. It is a relatively small community, with an active forum where users talk about electronic games and their sales. At the data collection date, the network had plenty of reviews about electronic games produced both by the official site crew and by ordinary users. The moderators were also very helpful with all the experiment, providing feedback and helping us to collect the data. Given all this reasons, gamrConnect fitted very well our objectives.

We trained a Naïve Bayesian classifier to extract the opinions. This strategy was applied because it outperformed SVM and logistic classifiers in an experiment conducted previously. This experiment was made using manually labeled samples (opinions) from the reviews that we collected. Details about this experiment are presented in Appendix B.

Once the opinions were filtered, we conducted polls with the users of the network. On the poll, we followed a standard information retrieval methodology

(Manning, Raghavan, & Schütze, 2008) with minor changes to allow us to collect data about the volunteers' profile.

Any user of the social network could play a judge role, and perform as many relevance judgments as he wanted. We adopted initially a predefined number of 10 judgments per poll, in order to avoid the users to spend a lot of time in the experiment, and thus occasionally discourage their participation. However, those users who were more motivated could answer the poll several times. Since our objective was to collect as much data as possible, we decided not to limit the amount of times that a particular user could answer the poll. However, this could introduce bias in the corpus if a user makes many relevance judgments compared to the others. Fortunately, it is also quite easy to avoid such bias while using the corpus to evaluate an opinion relevance model: it suffices to pre-process the corpus and remove some judgments of the most common users, if necessary.

Given our context and restrictions, we have made some adaptations. For example, the TREC Methodology can use many different strategies to generate a list of documents to be polled, but since there are not many opinions relevance models available in the literature, we decided to generate our list of opinions randomly. In our case, random generation also makes sense because for each subject the number of opinions to be evaluated is also relatively small, approximately 60 per subject, when compared to the TREC corpus which can exceed a thousand of documents retrieved per subject.

We could also have tried to generate this list by using a simple Information Retrieval strategy, like the Inverse Document Frequency (IDF). However, this approach could introduce bias in our corpus since the IDF would probably have scored the best results in our test, since all the opinions evaluated as relevant in the corpus would be retrieved. Finally, we also did not try to apply any kind of more elaborated strategies to conduct the polls mainly because they would require a more controlled enrolment for the poll, where we would have to rely on the contact of the judges at any time to perform a more deep poll or even split the process of polling in more than one step.

During the poll, each volunteer played the role of an opinion consumer and judged the relevance of 10 opinions by performing the following steps:

1. Input his/her username, so we could collect information from its profile on the social network;
2. Select an Electronic Game;
3. Judge the relevance of 10 opinions (sentences) about the chosen Electronic Game.

The form the volunteers had to fill before the poll is displayed in Figure 5.1. Furthermore, a hypothetical opinion example to be judged can be found in Figure 5.2.

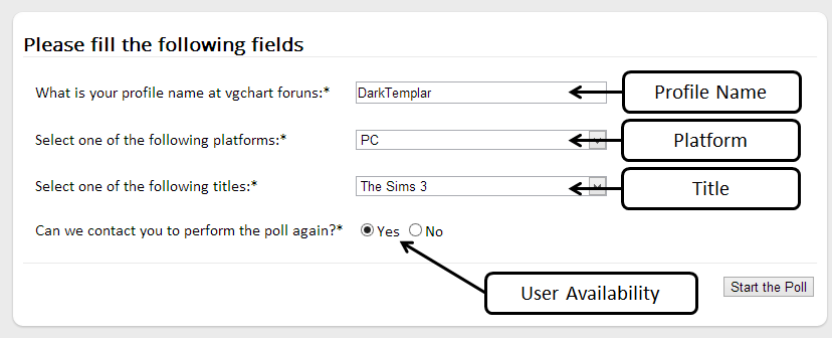


Figure 5.1 – Form that the volunteers had to fill before the poll.

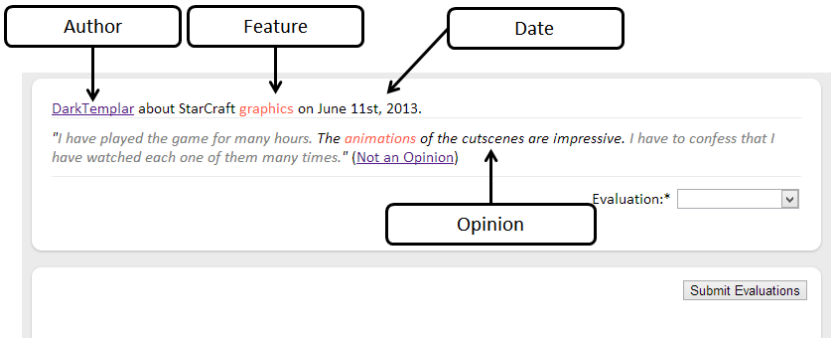


Figure 5.2 – Hypothetical sample of an opinion to be judged on our experiment.

Each opinion on the list could be evaluated in four levels: Irrelevant; Slightly Relevant; Relevant; and Very Relevant. Furthermore, all the sentences that were judged on the poll were previously filtered by an opinion classifier. However, since no classifier is perfect, we also provided the option “mark an opinion as non-opinion (notice this option just after the text of the opinion in Figure 5.2). This option would replace the opinion on the poll page with another opinion from our database. This

measure was introduced mainly to avoid as much as possible non-opinions judged in SORC-01.

5.3.1.2 Original dataset

The initial dataset we crawled and it is composed of 1,121 web pages, each of them representing a non-structured review of a game made by one of the VGChatz official reviewers or ordinary members of the site. The dataset has 243 different authors, and presented reviews about 777 different titles. We were able to extract 69,507 sentences from the reviews, and by using an opinion classifier we managed the find 46,857 opinions.

In order to create the opinion classifier, we manually labeled 1,126 samples of the 69,507 sentences of our dataset, and then we used them to train and test different strategies of text classification. As mentioned before, for our particular problem we found that Naïve Bayes Classifier had the best performance, as shown in Appendix B.

5.3.1.3 Test Cases

On SORC-G01, each Test Case can be expressed as the query “opinions about X” where X is the title of an Electronic Game. X was directly chosen by a consumer in a web page that we created to collect judgments by polling gamrConnect users.

Besides the test cases, SORC-G01 also provides information about both the consumers and the authors of the opinions, such as the Electronic Games they have interest in and their age. These data was collected directly from their public profiles on the social network. It is possible to know more details and download our corpus at https://allanlima.wordpress.com/2014/11/27/sorc-social-opinion-relevance-corpus-project/.

By introducing consumers and authors’ information into a corpus, indirectly we are also creating potential privacy problems. For instance, it is important to ensure the anonymous character of the information present in the corpus. Hence, we submitted our poll project to evaluation for the Ethical Committee at Research of the Hospital of the University of São Paulo (process n° 10141213.9.0000.0076 at Plataforma Brasil) before we start collecting judgments. The poll to create our corpus

was only started once we got the formal permission from the Ethical Committee to perform the experiment.

5.3.1.4 Features Overview

A comparison between SORC-G01 and other corpora in the literature can be found in Table 5.2. The table shows the source of their opinions, their discrimination level (just binary or with more levels), their granularity (what are their entities, i.e. documents, paragraphs, sentences), and if they provide information about the social profile of the author and the consumer of the opinions. We can notice that the information about the consumer and the author profile provided on our corpus consists in its main distinct feature when compared to the others of the corpora available in the literature.

Table 5.2 – Comparison of the corpora available on the Internet.

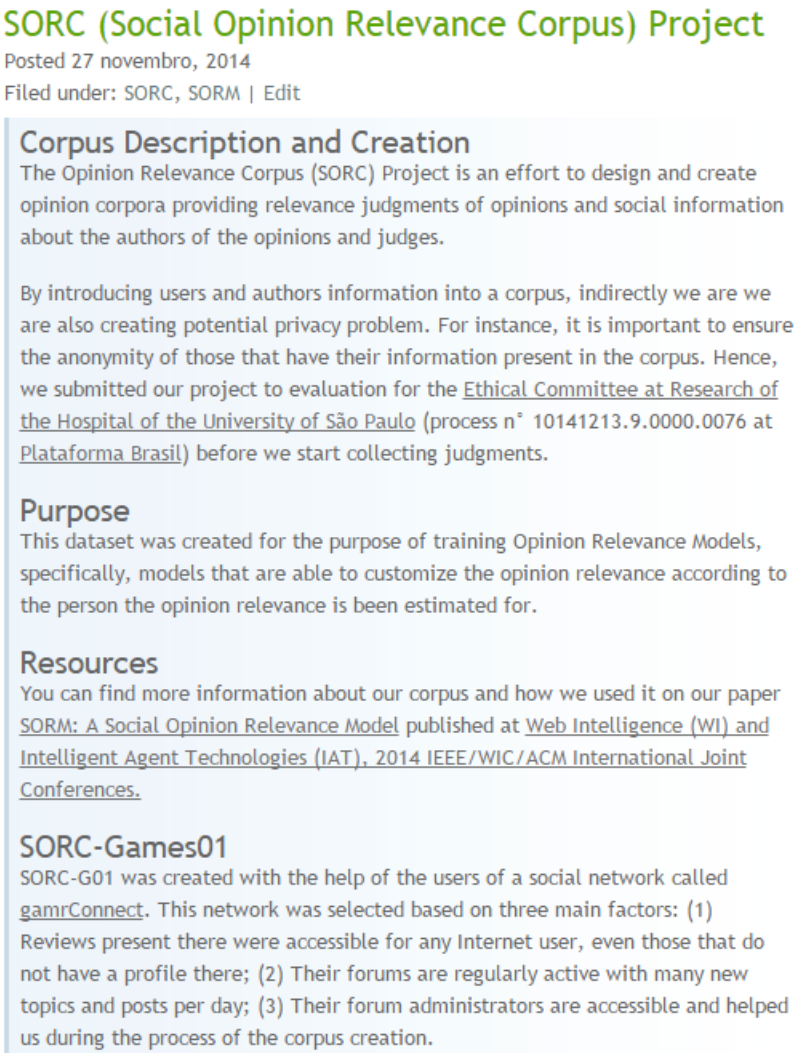
Corpus	Source	Relevance	Granularity	Author Profile	Consumer Profile
Wal-Mart	News on Wal-Mart	Binary	Paragraph	No	No
MPQA	New Blogs	No	No	No	No
NTCIR7	Newspaper	Binary	Sent., Doc.	No	No
TREC-2010	Blogs	Binary	Document	No	No
(Luo, Osborne, & Wang, 2013)	Twitter	Binary	Tweets	Yes	No
SORC-G01	Soc. Net. Reviews	4 levels	Sentence	Yes	Yes

5.4 Conclusions

SORC is a customized corpus created specifically for testing opinion relevance models. In order to achieve this goal, SORC provides all the necessary information to compute social and non-social parameters. As explained in Chapter 4, SORM contains metadata (such as the opinion's publishing date) besides the profiles of both the author and the consumer of the opinions. Given the limitations of the currently available corpora, SORC had to be created. The domain we have chosen to populate SORC was electronic games. This decision was made mainly because there is a reasonable amount of social networks on the Internet on this domain. Hence, this corpus, called SORC-Games01, is the first SORC corpus we have developed.

Despite not being the main contribution of our thesis, we consider that our work in creating SORC is an important result that can be used by other researchers

on their own projects. Therefore, we developed a web site containing a basic SORC documentation (Figure 5.3) and where SORC-Games01 can be downloaded for research purposes at https://allanlima.wordpress.com/2014/11/27/sorc-social-opinion-relevance-corpus-project/.



SORC (Social Opinion Relevance Corpus) Project

Posted 27 novembro, 2014

Filed under: SORC, SORM | Edit

Corpus Description and Creation

The Opinion Relevance Corpus (SORC) Project is an effort to design and create opinion corpora providing relevance judgments of opinions and social information about the authors of the opinions and judges.

By introducing users and authors information into a corpus, indirectly we are we are also creating potential privacy problem. For instance, it is important to ensure the anonymity of those that have their information present in the corpus. Hence, we submitted our project to evaluation for the Ethical Committee at Research of the Hospital of the University of São Paulo (process n° 10141213.9.0000.0076 at Plataforma Brasil) before we start collecting judgments.

Purpose

This dataset was created for the purpose of training Opinion Relevance Models, specifically, models that are able to customize the opinion relevance according to the person the opinion relevance is been estimated for.

Resources

You can find more information about our corpus and how we used it on our paper SORM: A Social Opinion Relevance Model published at Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2014 IEEE/WIC/ACM International Joint Conferences.

SORC-Games01

SORC-G01 was created with the help of the users of a social network called gamrConnect. This network was selected based on three main factors: (1) Reviews present there were accessible for any Internet user, even those that do not have a profile there; (2) Their forums are regularly active with many new topics and posts per day; (3) Their forum administrators are accessible and helped us during the process of the corpus creation.

Figure 5.3 – Site where SORC is documented and SORC-Games01 can be downloaded.

.

After describing SORC-G01, a corpus containing opinions about electronic games, we are now able to show how we used it in our experiments, i.e, how we use it to test an instantiated version of SORM in this domain. This is subject of the next chapter.

6 Experiments and Results

In this chapter, we describe the experiments we have conducted and we discuss the results that these experiments have generated. The design of the experiments is the corner stone of this chapter. We will detail all the decisions we made to perform the experiment, justifying their choice. We present the results we have found and discuss how we complement them by performing a significance test.

We used SORC-G01 as the corpus for the experiments. After a similar study to the one presented on section 4.6, where we examined the data in SORC-G01 we concluded that its data was sufficient for us to compute all SORM parameters. Furthermore, the extracted features on the domain of electronic games from the opinion on SORC-G01 using a dictionary of expressions that is presented on Appendix A.

Genetic Algorithms with cross-validation was the strategy we selected to balance SORM parameters in this domain. GA was selected because it makes no assumption about the problem being solved. Therefore it is a good starting point to solve our problem, since we could not find in the literature about to optimize a model to estimate the relevance of an opinion. Intuitively, we know that GA may not be the best solution for this problem. But at the same time we could not find evidence that it could be possible to others strategies to outperform it. We believe that this is a subject for another experiment, which is out of the scope of this work. Cross-validation is frequently used by researchers to avoid overfitting while learning from samples, this was also the main reason we choose this training police on our experiment.

In other words, we instantiated parameter values of the domain independent model to some concrete values whose combination and weighting were customized to the computer games domain. Using Information Retrieval metrics such as QMeasure and MAP, were could find evidence of the importance of customized results while estimating the relevance of opinions, a result that was later complemented by a significance test.

The remaining of this chapter is organized as follows: first, we describe how we designed an experiment to test SORM using SORC-G01; in section 6.2, we present the results we could find in the tests; in section 6.3, we introduce and comment the significance test results; finally, in the conclusions we make a brief recapitulation of the results obtained in the experiments.

6.1 Experiment Design

The first step of the experiment was a test comparing 9 distinct setups of opinion relevance models, including social and non-social models. The strategies were created using different subsets of SORM parameters in order to mimic social networks and similar projects. Table 6.1 shows the setups selected for comparison during the experiments.

Table 6.1 – Opinion Relevance Models Tested During the Experiments.

ID	Name	Parameters
01	IDF	OIWF
02	Newer	Linear Longevity
03	Non-Social-NO	Feature Score, OIWF and LinearLongeviry
04	Non-Social	Non-Social-NO plus Positive and Negative Orientation
05	Friends	Network Distance
06	Similar	Similarity and Age Similarity
07	SORM-NC	Non-Social plus Author Experience and Feature Experience
08	SORM-NO	all SORM parameters except Positive and Negative Orientation
09	SORM	all SORM parameters

Each setup listed on Table 6.1 has a particular motivation. *IDF* is how a simple information retrieval system would rank opinions. It uses only our adaptation of the Information Retrieval metric Inverse Document Frequency (OIWF, section 4.4.3) to rank the opinions. Intuitively, this model means that opinions containing rare words will have a higher relevance.

Newer is a heuristic used by some social networks, where the newest comments are supposed to be the most relevant. Despite its simplicity, in theory this heuristic can be quite powerful on the domain of electronic games. This happen

because games that were created using cutting edge technology and innovative concepts may generate a lot of positive opinions around their launch window. However, after a few years in the market those opinions may become dated and, consequently, irrelevant given that many games do not age well across the years.

Non-Social-NO uses the parameters that are not classified as social, except the opinion orientations. This model has a very specific importance for us since its results will be relevant to verify the research hypothesis described on section 1.2. Therefore, comparing non-social variations of the parameters with social versions we will be able to find out if our Hypothesis is true or not on the domain of electronic games.

Non-Social represents an Information Retrieval system for ranking opinions which follows a similar paradigm to the related works. This model uses information about the opinions as one of its parameters, but at the same time it does not contains any social parameter. Like *Non-Social-NO,* this model is also important to test our research Hypothesis.

Friends means that opinions from people close to the consumer (on the social network) will be prioritized. This model follows a heuristic that opinions from friends have more relevance for a consumer. As discussed on section 2.2, in social search this is strategy that is used on distinct applications while ranking data from social networks. This is also the first model in our experiment that contains social aspects.

Similar ranks higher the opinions from authors that are similar to the consumer. Once again, this model follows a simple yet, in theory, effective heuristic: opinions from people that are similar to a consumer must be relevant to him. This model is also composed by social parameters, in this case, Similarity and Age Similarity.

SORMC-NC is a model composed by all non-social parameters plus Author Experience and Feature Experience. This model was inspired by the closest work to our project we could find in the literature (Luo, Osborne, & Wang, 2013). Intuitively, this model complements non-social parameters with parameters that are social, but it is not able to provide customization; this is essentially the paradigm followed by Luo on his work.

SORM-NO has all SORM parameters except the information about the polarity of the opinions. This model provides customized results by using a set of social parameters. The exceptions are both opinion orientation parameters. This makes this model a slight variation of the full model, just ignoring how positive and negative the opinions are.

Finally, *SORM* combines all parameters. Hence, our main hypothesis of the customization's importance while estimating the relevance of opinions will be tested by comparing the results obtained by both this model and *SORM-NO* with those obtained by Non-Social and Non-Social-NO models.

6.1.1 Evaluation Metrics

For all those 9 different setups, we test the hypothesis that customization is important in order to achieve better results on information retrieval metrics. Thus, we are looking to compare the results among the strategies that are able to provide custom results with strategies that are not.

For the evaluation, we chose the following metrics: RPrecision, Means Average Precision (MAP), bpref, Normalized Discounted Cumulative Gain (NDCG) and QMeasure (Manning, Raghavan, & Schütze, 2008; Sakai & Kando, 2007). As mentioned in section 5.2.2, RPrecision and MAP are standard metrics used on many information retrieval experiments (Manning, Raghavan, & Schütze, 2008). Bpref was one of the first metrics created to deal with incomplete data, when the judgments in the test cases do cover all the possible results. NDCG and QMeasure can be also useful to perform tests with incomplete data and are considered even better than bpref for this task (Sakai & Kando, A Further Note on Alternatives to Bpref, 2007).

Furthermore, as suggested by (Sakai & Kando, 2007), we also decided to use modified versions of the chosen metrics due to the incompleteness of SORC-G01. For instance, originally QMeasure is computed based on the opinions retrieved by a setup. We also used a modified metric, called QMeasure', that uses the opinions both retrieved by the setup and judged by a consumer. We have repeated this modification for the other metrics. Therefore, when we quote a metric with' (an apostrophe) on its name, we mean that it was computed using only opinions that were both retrieved and judged by a consumer.

Since each test case has a number k of opinions judged as relevant by a consumer, we retrieved the k most relevant opinions on each test case (top k).

6.1.2 Parameter Balance

In order to discover the weight and the combination of each parameter, we decided to use Genetic Algorithms (GA) (Whitley, 1994). As discussed in section 2.5, GA is a strategy to solve optimization problems that makes no assumptions about the solution domain. We followed this generic approach, since there are no in depth studies on how to optimize a Social Opinion Relevance Model for a particular domain.

In our case, each individual on the GA codification represents the weights and combinations of the parameters on the setup. Moreover, each parameter was coded using 19 bits: one bit for the combination type (sum or multiplication); two bits for the weight type (multiplication, division, logarithm, and exponential); and sixteen bits to encode two doubles (eight bits each) allowing 256 possible values with a 0.01 precision. Therefore each double can assume values from 0.00 to 2.55. Table 6.2 illustrates how an element is codded in our implementation of GA. As SORM has 12 parameters, an individual would have a codification with 12 * 19 bits = 228 bits. Models with a small number of parameters would have fewer bits on its individuals. For any given model, the number of bits of an individual is: number of parameters * number of bits per parameter).

Table 6.2 – How a parameter is codded in our GA experiments.

Bits																		
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
C	W		D1								D2							

Legend
C – Combination | W – Weight | D1 – Double #1 | D2 – Double #2

In order to illustrate the translation between an array of bits representing an individual in GA and the meaning of this individual in our context, we introduce here a few samples of how this happens during our experiment. Table 6.3 shows the last 57 bits of an individual found as the best in a test case during one of our experiments.

On the table, it is possible to see that the codification of each section of 19 bits (second row) of it is responsible for a parameter of the model (third row).

Table 6.3 – A sample of the last 57 bits of individual generated during one of our experiments.

Codification Samples			
Codification	0001110101011000010	0001110110011001101	1001001011010000010
Meaning	+ (1.94 * LinearLongevity)	+ (2.05 * NetworkDistance)	* (1.3 * Similarity)

Regarding the training process, the canonical implementation (Whitley, 1994) was applied with a thousand individuals on each generation, where the first generation was randomly initialized, and then evolved for a hundred generations. The classical mutation and crossover operators were applied. Regarding the mutation rate, each bit of an individual has a chance of 1% of being changed. The crossover operator results on the combination of the bits of two distinct individuals to form a new one. The operator was applied to create the new individuals between two generations.

During the experiments NDCG' was used to compute the fitness, therefore individuals with better results on NDCG' had a high chance of generating decedents. The other metrics were computed to provide a complementary point of view of the performance of the models, as well as allow us to know if the optimized model generated was biased towards the main metric (NDCG').

During the balance of the parameters, we applied cross-validation to prevent overfitting. Therefore, for each test case on SORC-G01 we used all other 49 test cases to balance the parameters and then compute the information retrieval metrics. Therefore, the training was similar to use cross-validation, where the performance of the setups was computed in chunks of 10 samples using a model trained with all other samples.

Figure 6.2 illustrates an interaction of our cross-validation process. It shows that based on an initial random generated population, a target test case and the remaining of the test cases GA generates the best individual it could find. As discussed before, the best individual is the one with the best fitness which is computed using the NDCG'.

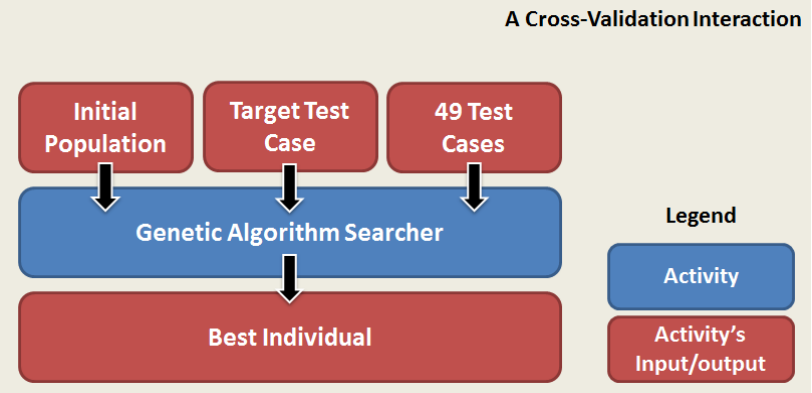


Figure 6.1 – Finding the best individual for test case using cross-validation.

Once GA finds the best individual now we can compute the metrics we proposed for the experiment as displayed in Figure 6.2. This is made by submitting the test case and the best individual to a test framework that is able to compute all the metrics selected for the experiment.

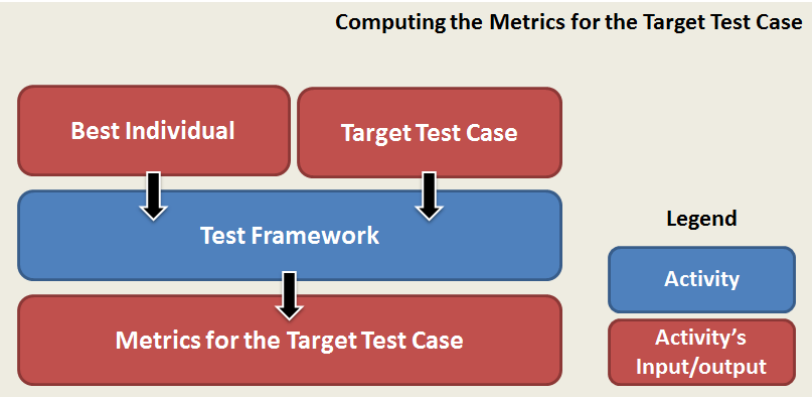


Figure 6.2 – Computing the Metrics for the target test case using the best individual.

The search for the best individual and the computation of its performance based on the metrics is a process that was replicated for each test case on SORC-G01. At end of this task, we have the performance of a model for all test cases of SORC-G01. Now the next step is the computation of the average of each metric, reaching the final value of the metrics for the model. Those values are the ones that we compare with other models and use on the significance tests. Finally, notice that models that contain a single parameter, for instance *IDF,* do not need to be balanced using the process we described. This happens because it would not change the final ranking of the opinions if a model with a single parameter has its parameter's weight or combination balanced by our GA implementation.

6.2 Results

The first results we computed and compared relate to the non-social strategies (Table 6.4). The data displayed on the table is the average of the performance of the model on the 50 test case of SORC-G01. In six out of our ten metrics, represented in bold, *Non-Social-NO* has the better performance. Furthermore, we would like to reinforce that *Non-Social-NO* had the best results on QMeasure', which is one of the most important metrics of our tests, as discussed in Section 6.1.1.

Table 6.4 – Results of the non-social relevance models.

	IDF	Newer	Non-Social-NO	Non-Social
NDCG	0.0297	0.0397	**0.1037**	0.0878
NDCG'	0.2442	0.2342	**0.3390**	0.3157
RPrecision	0.0409	0.0490	0.0690	**0.0747**
RPrecision'	0.2700	0.2167	**0.3733**	0.3367
Bpref	0.4191	0.4269	**0.4504**	0.4502
Bpref'	0.2700	0.2311	**0.3856**	0.3483
QMeasure	0.0093	0.0205	0.0355	**0.0381**
QMeasure'	0.0340	0.0433	0.0600	**0.0665**
MAP	0.0101	0.0214	0.0392	**0.0420**
MAP'	0.2650	0.2133	**0.3711**	0.3367

These first results indicated that the best model included all non-social parameters, *except the opinion orientations*. This means that its parameters are almost all important to improve the results by complementing each other. However, the orientation of the opinions did not help to improve the performance of the *Non-Social* model.

Following a similar pattern, the best social setup (*SORM-NO)*, as shown on Table 6.5, included all parameters, *except opinion orientations*. *SORM-NO* was able to outperform all other models in six out of our ten metrics, represented in bold in the Table. Moreover, we can notice that Author Experience and Feature Experience seem to be the most relevant social parameters, since these are present in SORM-NC, which is the second best social setup model, outperforming the other models in

four metrics. This result is the first evidence we found that parameters idealized during the design of the SORM where mostly useful to compose the model and increase its performance, at least on the domain of Electronic Games. This is also the first step towards the verification of our experiment hypothesis.

Table 6.5 – Results of the social relevance models.

	Friends	Similar	SORM-NC	SORM-NO	SORM
NDCG	0.0397	0.0537	**0.0785**	0.0733	0.0631
NDCG'	0.2342	0.2084	0.3548	**0.3709**	0.3447
R-Precision	0.0490	0.0579	0.0756	0.0828	**0.0865**
RPrecision’	0.2167	0.2233	0.3933	**0.4333**	0.4260
Bpref	0.4269	0.4429	0.4467	**0.4483**	0.4438
Bpref’	0.2311	0.2306	0.3956	**0.4456**	0.4310
QMeasure	0.0205	0.0327	**0.0361**	0.0316	0.0332
QMeasure’	0.0433	0.0509	0.0677	**0.0718**	0.0692
MAP	0.0214	0.0344	**0.0380**	0.0341	0.0358
MAP’	0.2133	0.2211	0.3911	**0.4311**	0.4159

Comparing exclusively the results of the best social model with all other models in this test, we can see that *SORM-NO* outperforms *Non-Social-NO* and *SORM-NC* in most of the metrics, and especially on QMeasure’ (as show on Table 6.6. This illustrates how an opinion relevance model with customized results has advantages with respect to other models.

Interestingly on both tests the fact that opinion orientation parameters were not able to improve the results could mean that either they are not important to the consumers that made the judgments on SORC-G01, or that the SentWorldNet was not able to work as well as we expected on the domain.

Table 6.6 – Results of the best relevance models.

	Non-Social-NO	Non-Social	SORM-NC	SORM-NO	SORM
NDCG	**0.1037**	0.0878	0.0785	0.0733	0.0631
NDCG'	0.3390	0.3157	0.3548	**0.3709**	0.3447
RPrecision	0.0690	0.0747	0.0756	0.0828	**0.0865**
RPrecision'	0.3733	0.3367	0.3933	**0.4333**	0.4260
Bpref	**0.4504**	0.4502	0.4467	0.4483	0.4438
Bpref'	0.3856	0.3483	0.3956	**0.4456**	0.4310
QMeasure	0.0355	**0.0381**	0.0361	0.0316	0.0332
QMeasure'	0.0600	0.0665	0.0677	**0.0718**	0.0692
MAP	0.0392	**0.0420**	0.0380	0.0341	0.0358
MAP'	0.3711	0.3367	0.3911	**0.4311**	0.4159

6.3 Significance Test

As discussed in our methodology, the Fisher's Randomization test was applied to complement our results. This test helps us to know if the difference on the results we showed on the previous section is in fact statically significant. During our experiments, we decided to compute 100 000 permutations.

On this test, two models that are exactly equal will produce a score of 1.0. Therefore, the closest to 1.0 the results of the comparisons are, the less significant the difference between the models is.

Furthermore, given the nature of Genetic Algorithms, every time we balance the parameters of a model the obtained results are slightly different. This is a consequence of the non-determinism of the search process on GA. In practice, if we instantiate a model twice for a domain using GA, despite having the same parameters both models will differ in the way their parameters are weighted and combined.

Fortunately this can also be useful to our experiments. Let us say we want to compare a model A with the model B. We can thus use GA to balance A twice, which generates slightly different results. Probably, the second time when A is trained the best individuals generated during the GA search may result in different weights and

combinations. However, since every time we balance a model using GA, the balance created for each test case tends to converge to a similar point. Hence, the results on the metrics we computed tend to be not much different, especially when compared with results from models containing distinct parameters.

But how can this feature be useful to our experiments? For instance, let us say we want to study the significance of the difference in the results achieved by *SORM-NO* and *Non-Social-NO*, respectively the best social and non-social models in our experiment. In a given metric, for example MAP. First we can compare two distinct results achieved by slightly different balanced *SORM-NO* models generated by GA. Performing the Fisher's Randomization test using the results of both models in MAP will give us a p-value that we call x (notice that $x < 1.0$, since both models are not exactly equal). Now we perform the significance test for *SORM-NO* and *Non-Social-NO* on MAP too, let us call *now* the p-value of this new test y (notice that $y < 1.0$, as well). Intuitively, the value of y is supposed to be lower than x ($y < x < 1.0$) which means that the difference between *SORM* and *Non-Social-NO* is more significant than two slight different balanced versions of *SORM-NO*. However, if $y > x$ we cannot say that difference between *SORM-NO* and *Non-Social-NO* is significant, therefore any difference on MAP would not be conclusive according to the test.

Since the most important result we want to study is the difference between social and non-social models, the first tests we performed were:

Test #1: Two *SORM-NO* models generated by Genetic Algorithm – this test stablishes a baseline to further comparisons. In theory, the p-value we expect in test are close to 1.0 and at same time higher than other runs of the significance test comparing *SORM-NO* and other models. The test was performed with all the metrics we showed on the previous section;

Test#2: *SORM-NO* and *Non-Social-NO* – A comparison of the best social setup with the best non-social setup. Smaller values than test #1 are expected to confirm the significance of the results previously discussed.

Test#3: *SORM-NO* and *SORM-NC* – A comparison of the best social setup with the setup that is social but not able to provide customization. Smaller values

than test #1 are expected to confirm the significance of the results previously discussed as well.

In addition, we designed a fourth test, this time focused on the phenomenon of the orientation of the opinions not helping to improve the results of the models we have tested. Therefore, we performed the following comparison too:

Test #4: *SORM-NO* and *SORM* – A comparison between SORM and its version without the opinion orientations was made to reinforce that both parameters are not important for SORM on the SORC-G01. Once again, smaller values than test #1 are expected.

The results displayed on Table 6.7 show the two-sided p-value of each comparison. They reinforced our previous tests by usually showing higher values on *SORM-NO/SORM-NO* (#1) significance when compared to the *SORM-NO/Non-Social-NO* (#2) and the *SORM-NO/SORM* (#3) tests.

Table 6.7 – Results of the significance tests.

	#1	#2	#3	#4
NDCG	**0.8373**	0.6790	0.1732	0.7968
NDCG'	**0.8718**	0.3889	0.1752	0.6243
R-Prec.	**0.8878**	0.6168	0.3146	0.8341
R-Prec.'	**0.9406**	0.6117	0.4276	0.8455
Bpref	0.8908	0.1596	0.1127	**0.9097**
Bpref'	**0.9421**	0.6143	0.4498	0.8107
QMeasure	**0.9048**	0.3117	0.2876	0.8915
QMeasure'	**0.8848**	0.6012	0.2081	0.8023
MAP	**0.8794**	0.3781	0.2515	0.8822
MAP'	0.8357	0.6146	0.4321	**0.8570**

For instance, let us analyze the two-sided p-values for QMeasure'. They were 0.8848 (*SORM-NO/SORM-NO*), 0.6012 (*SORM-NO/SORM*), 0.2081 (*SORM-NO/SORM-NC*), and 0.8023 (*SORM-NO/Non-Social-NO*). They represent an evidence that the results found on our first experiments were in fact significant

because, as expected, p-values of first comparison were higher than the others. *Consequently, the score difference on QMeasure' for the models SORM-NO, SORM, SORM-NC and Non-Social-NO presented on* Table 6.6 *(0.0718, 0.0692, 0.0677 and 0.0600, respectively) seems to be significant*. In other words, it means that *SORM-NO* is better than *SORM-NC and Non-Social-NO* on the QMeasure' metric when using SORC-G01 corpus.

At the same time, the results also indicate SORM-NO is also better than SORM an evidence the orientation of the opinion (or at least the strategy we used to compute them) is not necessary to improve the results of an opinion relevance model when using SORC-G01 corpus.

Now let us look to our experiment hypothesis once more. We wanted to verify, by using SORC-G01, whether a model capable of customize the results to the opinion consumer is able to outperform models unable to customize the results on standard IR metrics. At the end of our set of experiments, we found evidences indicating that our hypothesis is true. For instance, by comparing the results on QMeasure' for the models we tested, *SORM-NO* was able to outperform *Non-Social-NO*. Those evidences were later verified by a significance test.

6.4 Conclusions

In this chapter, we discussed how we instantiated and tested SORM in the domain of electronic games using SORC. We also described how we could find evidence that customization, in fact, plays an important role when estimating the relevance of an opinion. Each step of this task was described in detail, since the design of the experiment to the final significance test, in a way that other researchers interested in the subject could replicate our results.

The results we found using SORM and SORC are important baselines that we have established, so that our future work could advance the state of the art. Hence, any change or improvement made either on SORM or on our balancing methodology now will have some baseline results to be compared with.

The legacy, the challenges and the future work of our research are presented in the next, and last, chapter of this thesis.

7 Conclusions and Future Work

We presented a Social Opinion Relevance Model called SORM which, unlike the others found in the literature, is able to generate customized results based on the profile of the user that is searching for opinions. Since the current Opinion Relevance Corpora available did not provide enough information to enable us to test most of the parameters in our model, we also had to create our own Social Opinion Relevance Corpus. For that, we chose the domain of electronic games to populate SORC, creating what we called the SORC-G01 corpus. Using this corpus, we performed some experiments with distinct information retrieval metrics, complemented by a significance test, finding evidence with respect to the importance of the customization in the results when it comes to the problem of opinion relevance estimation. This evidence was accepted by peers working in the area, and has generated a publication in a major international conference (Lima & Sichman, 2014)

During this thesis we documented the methodologies, the decisions and contributions we made so anyone could comprehend them and replicate our results. We also provided as much information as possible about the requirements to comprehend our project, so the reader do not need to look for others resources (papers, books, etc.) in order to understand what we presented here (assuming that he has basic knowledge about computer science).

At the end of this research effort, we were able to achieve the main contribution to Opinion Mining we have aimed for: the creation and formalization of a social and domain independent Opinion Relevance Model. However, this contribution was only possible after we have achieved other important contributions to the opinion mining research field. We want to highlight the three of them we consider more relevant: (1) a methodology to evaluate opinion relevance models; (2) the creation of a Social Opinion Relevance Corpus that, unlike that others we could find on the literature, made the evaluation of our model feasible; (3) formal definitions to classify the social and customizable aspects of an Opinion Relevance Model.

As any other projects, our solution has its possible applications and limitations. Here we will address some examples of how our abstract opinion relevance model

can be used on real-life problems. Furthermore, we will also address its limitations to avoid misunderstandings related to the scope of our work. In order to achieve the contributions we presented here, we had also to face many distinct challenges. Therefore, we have also chosen where to concentrate our effort, which leaded to distinct and important extensions that can be made to our work in the future.

7.1 Applications of our Solution

SORM can be applied to different real life problems. Here we will show three different problems where it could be useful: (1) a Social Opinion Search Engine; (2) Snippets in general purpose search engines; and (3) Social Networks Collaborative pages.

Creating a Social Opinion Search Engine is perhaps the most obvious application of our model. Such engine would use the consumer profile together with the authors profile to generate a customized list of opinions about a subject.

The second application is present on general purpose search engines, like Google. For users that make a query like “just dance review” (Figure 7.1) those engines display a small text fragment below the link to the review, which is called snippet. For this type of query, a Social Opinion Relevance Model could be applied to customize the most relevant opinion for a given user, instead of the same snippet for all users.

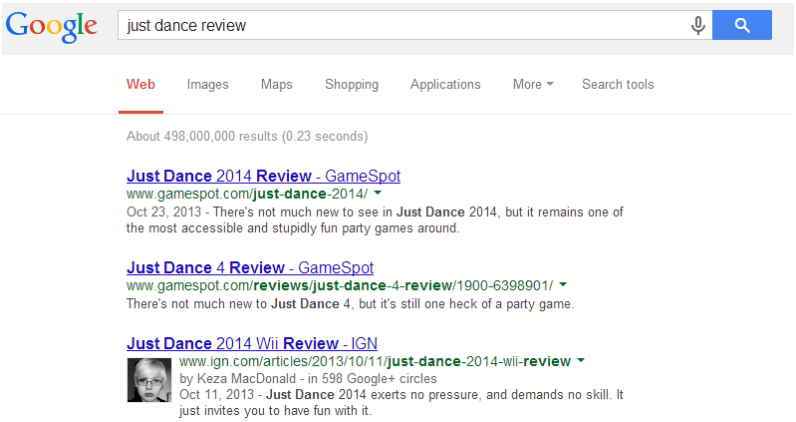


Figure 7.1 – Searching for “just dance review ” on Google.

Social Networks usually have thematic pages about the subjects that their users are talking about. For instance, in the Playfire[9] Social Network about electronic games, each game has its own page with comments from other users (Figure 7.2). Normally, the comments are ranked by date, i. e. the newer has the higher rank and it is presented in the top. However, since many of those comments are opinions, this type of Social Network could use our Social Opinion Relevance Model to filter the opinions that are most relevant to their users.

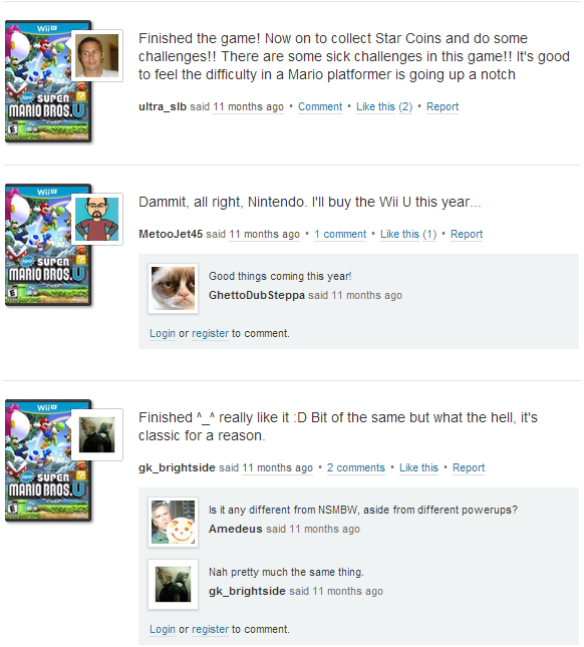


Figure 7.2 – A small fragment of a page of the electronic game New Super Mario Bros U at the social network Playfire.

7.2 Scope Limitation of the Proposed Solution

Our model is conceived as the first step in the development of an Opinion Search Engine. By an Opinion Search Engine, we mean a whole system where the user gives as input a query and the system returns a list of opinions. However, the implementation of a whole opinion retrieval system is out of the scope of a thesis,

[9] www.playfire.com

Otherwise, we would have to dedicate time to the development of a system that would not only require functional tests, but usability tests as well.

SORM is not meant to be applied in any other type of systems such as product recommendation systems and support decision systems. In addition, our project is not a replacement for current Information Retrieval models for document ranking.

No complex syntactical or morphological analysis of the text of an opinion is made in our model. There are many ways to write an opinion, an author could use comparisons, arguments, metaphors and even ironies. Covering all those types of different expression would require a new multidisciplinary research project using concepts from Linguistics and Natural Language Processing, which we see as an effort as big as the one presented here. Thus, during the development of our model we decided that we were not going to cover this issue.

Furthermore, another limitation in the scope of our model is that to compute its parameters it is mandatory that there is available information about the user, the author and the opinion. For example, some of our parameters, related to social search, require access to the friends' social network of a user, so those parameters cannot be computed without this information. Hence, we can say that the easiest environments to apply our model are sites where the users have a social profile and can also freely generate content (opinions). As examples of this kind of sites we have general purpose social networks like Facebook[10] or Google Plus[11], and thematic social networks like Skoob[12] (Books), Raptr[13] (Electronic Games), Filmow[14] (Movies, Series and Cartoons), and Anime-Planet[15] (Animes and Mangas).

---

[10] http://facebook.com

[11] http://plus.google.com

[12] http://skoob.com

[13] http://raptr.com

[14] http://filmow.com

[15] http://anime-planet.net

Topic relevance estimation is also out of the scope of our work. Both Information Retrieval and Opinion Mining have plenty of studies regarding the document and opinion topic relevance respectively. Consequently, our model assumes that the opinions that it has to rank are relevant to one of more subjects.

Finally, it is out of the scope of this project to define how to index and structure web pages in order to have the necessary information available to our model. In other words, our model does not cover or formalizes how to convert html pages into structured data in a database containing the information needed to compute its parameters.

7.3 Main Challenges

During our research project we had to overtake many distinct challenges. Among of them, we can highlight: (1) the lack of related work; (2) the lack of opinion corpora for our experiments; (3) privacy/anonymity issues on the poll; (4) receiving permission to collect data from social networks for our experiments from their representatives.

The lack of related work was an issue that we handled naturally, given the innovative requirement of a PhD thesis; therefore, it was expected not to find many projects close to the one we were developing. However, unfortunately, the absence of similar projects was more intense than we could anticipate. The main consequence of this can be seen on the experiments, where we had to compare our model with strategies inspired by information retrieval approaches, instated of comparing SORM with other opinion relevance models. For instance, we wanted to compare SORM with the model proposed by (Luo, Osborne, & Wang, 2013), but it was not possible since the model is completely coupled with the social network it was crafted for (the Twitter). Even if we wanted to instantiate SORM on twitter using the corpus they used on their experiments, we would not be able to instantiate SORM parameters responsible for providing the customization to the opinion consumer. This happens because of the lack information about the people that judged the tweets in that corpus. For us, it was not clear even if the opinion judges had a social profile on Twitter.

The lack of opinion relevance corpora for our tests was also an important challenge. Given that none of the available corpora that we could find provided the information needed the test our research hypothesis, we had to create one. This process consumed a huge effort during our project, making us to deal with procedures that we had no experience with before, such as submitting a project to the evaluation of an ethical committee in research.

As a consequence of this issue, we had to study how to perform polls with humans while respecting their privacy and anonymity. The borderline between what we could and could not collect and publish about the volunteers on the social networks was also a point of debate and discussion. Consequently, many changes had to be made on our original poll project; for instance, we had to add a restriction to our poll that any volunteer was at least 18 years old, which had an important impact on the chosen domain.

Perhaps the biggest challenge of our work was collecting data to form a Social Opinion Relevance Corpus. Despite the bureaucracy to receive formal permission from an ethical committee to collect data from Social Networks, we have faced many other problems to create our corpus. The main problem was related to receive a permission from the moderators or owners of the social networks to create the opinion corpora. There are some important events related to this challenge and how we have handled them, this issue is addressed in detail on Appendix C.

7.4 Further Work

Intending to improve our work, first we want to test our model in different domains by populating SORC corpus with more opinions. Books and Movies are some of the domains currently on study to perform these new experiments.

We can also improve our results by performing new experiments to compare our model to more non-social opinion relevance strategies. Unfortunately, not all models available in the literature can be used on our tests, as they may be strongly linked to a domain or a social network. However, usually it is possible to identify and follow the paradigms and heuristics behind them.

Now that the baselines are established, it is possible to explore distinct approaches to balance the parameters of the model. Thus, optimizations strategies other than Genetic Algorithms can be applied to find out what is the best way to optimize an opinion relevance function for a domain. Improvements on our methodologies to create opinion relevance corpus and test opinion relevance models are also important to improve the quality and reliability of our research.

Finally, SORM itself can also be refined by adding new parameters. Among the research fields that could not properly address during our project, Marketing and Social Sciences are our priorities. We expect that both of them could be able to improve our model by inspiring new parameters to be taken into account for improving social opinion relevance models.

References

Aslam, J. A., Pavlu, V., & Yilmaz, E. (2005, August). A Sampling Technique for Efficiently Estimating Measures of Query Retrieval Performance Using Incomplete Judgments. *22nd ICML Workshop on Learning with Partially Classified Training Data*, (pp. 57-66).

Aslam, J. A., Pavlu, V., & Yilmaz, E. (2006). A Statistical Method for System Evaluation Using Incomplete Judgments. *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 541-548). ACM.

Attardi, G., & Simi, M. (2006). Blog Mining Through Opinionated Words. *15th Text Retrieval Conference (TREC)*.

Baccianella, S., Esuli, A., & Sebastiani, F. (2010). SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining. *The Seventh International Conference on Language Resources and Evaluation, 10*, pp. 2200-2204.

Bäck, T., & Schwefel, H.-P. (1993). An Overview of Evolutionary Algorithms for Parameter Optimization. *Evolutionary computation, 1*, pp. 1-23.

Balog, K., Serdyukov, P., & Vries, A. P. (2010). *Overview of the TREC 2010 entity track.* Norwegian University of Science and Technology Trondheim.

Brin, S., & Page, L. (1998). The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks and ISDN Systems*, pp. 107-117.

Buckley, C., & Voorhees, E. M. (2004). Retrieval evaluation with incomplete information. *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 25-32.

Carterette, B., Allan, J., & Sitaraman, R. (2006). Minimal test collections for retrieval evaluation. *29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 268-275.

Chris , B., Dimmick, D., Soboroff, I., & Voorhees, E. (2006, August). Bias and the Limits of Pooling. *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 619-620). ACM.

Cormack, G. V., Palmer, C. R., & Clarke, C. L. (1998). Efficient Construction of Large Test Collections. *Iternational ACM SIGIR Conference on Research and Development in Information Retrieval*.

Dawkins, R. (2006). *The selfish gene.* Oxford university press.

Dupont, C. (2008, November 20). *Google Official Blog*. Retrieved August 3, 2014, from SearchWiki: make search your own: http://googleblog.blogspot.com.br/2008/11/searchwiki-make-search-your-own.html

Esuli, A., & Sebastiani, F. (2006). Sentiwordnet: A publicly available lexical resource for opinion mining. *Language Resources and Evaluation Conference*, *6*, pp. 417-422.

Evans, B. M., & Chi, E. H. (2010). An Elaborated Model of Social Search. *Information Processing & Management*, 656-678.

Facebook. (2014). *Introducing Graph Search*. Retrieved August 3, 2014, from Facebook: https://www.facebook.com/about/graphsearch

Fisher, R. A. (1935). *The Design of Experiments* (1 ed.). Oliver and Boyd.

Ghose, A., Ipeirotis, P. G., & Sundararajan, A. (2007). Opinion mining using econometrics: A case study on reputation systems. *Annual Meeting-Association for Computational Linguistics*, *45*, p. 416.

Google. (2013, September 2013). *We Hear You: Better Commenting Coming to Youtube*. Retrieved August 3, 2014, from Youtube Official Blog: http://youtube-global.blogspot.com.br/2013/09/youtube-new-comments.html

Google. (2014, 06 15). *Author information in search results*. Retrieved August 3, 2014, from Webmaster Tools: https://support.google.com/webmasters/answer/1408986?hl=en

Hearst, M. A., Dumais, S. T., Osman, E., Platt, J., & Scholkopf, B. (1998). Support vector machines. *Intelligent Systems and their Applications, 13*(4), 18-28.

Horowitz, D., & Kamvar, S. D. (2010). The Anatomy of a Large-Scale Social Search Engine. *19th International Conference on World Wide Web* (pp. 431-440). ACM.

Huang, X., & Croft, W. B. (2009). A Unified Relevance Model for Opinion Retrieval. *18th ACM Conference on Information and Knowledge Management*, pp. 947-956.

Järvelin, K., & Kekäläinen, J. (2002). Cumulated Gain-based Evaluation of IR Techniques. *ACM Transactions on Information Systems (TOIS)*, 422-446.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. *Springer Berlin Heidelberg*, pp. 137-142.

Jones, K. S. (1972). A Statistical Interpretation of Term Specificity and its Application in Retrieval. *Journal of Documentation*, 11-21.

Kleinberg, J. M. (1999). Authoritative Sources in a Hyperlinked Environment. *Journal of the ACM*, pp. 604-632.

Lee, D., Jeong, O.-R., & Lee, S.-g. (2008). Opinion mining of customer feedback data on the web. *International Conference on Ubiquitous Information Management and Communication*, (pp. 230-235).

Lewis, D. D., Yang, Y., Rose, T., & Li, F. (2004). A New Benchmark Collection for Text Categorization Research. *The Journal of Machine Learning Research, 5*, pp. 361-397.

Li, Y. H., & Jain, A. K. (1998). Classification of Text Documents. *The Computer Journal, 41*(8), pp. 537-546.

Lima, A. D., & Sichman, J. S. (2014). S.O.R.M.: A Social Opinion Relevance Model. *Web Intelligence Conference.* Warsaw.

Liu, B. (2009). Opinion Mining. In T. M. Özsu, & L. Liu, *Encyclopedia of Database Systems* (pp. 1986-1990). Springer.

Liu, B. (2010). Sentiment analysis and subjectivity. In *Handbook of Natural Language Processing* (Vol. 2, pp. 627-666).

Liu, B. (2012). Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies, 5*, pp. 1-167.

Luo, Z., Osborne, M., & Wang, T. (2013). An Effective Approach to Tweets Opinion Retrieval. *World Wide Web*, pp. 1-22.

Macdonald, C., & Ounis, I. (2006). *The TREC Blogs06 collection: Creating and analysing a blog test collection.* Glasgow: Department of Computer Science, University of Glasgow Tech Report TR-2006-224.

Macdonald, C., Ounis, I., & Soboroff, I. (2007). Overview of the TREC 2007 Blog Track. *Text REtrieval Conference, 7*, pp. 31-43.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval.* Cambridge University Press.

Microsoft. (2014, 06 15). *Connecting Facebook to Bing*. Retrieved August 3, 2014, from Connecting Facebook to Bing: http://www.bing.com/explore/social

Miller, G. A. (1995). Wordnet: a Lexical Database for English. *Communications of the ACM, 38*(11), pp. 39-41.

Mishne, G. (2006). Multiple Ranking Strategies for Opinion Retrieval in Blogs. *15th Text Retrieval Conference*.

Morris, M. R., Teevan, J., & Panovich, K. (2010b). What Do People Ask Their Social Networks, And Why?: A Survey Study Of Status Message Q&A Behavior. *SIGCHI Conference on Human Factors in Computing Systems* (pp. 1739-1748). ACM.

Morris, M., Teevan, J., & Panovich, K. (2010a). A Comparison of Information Seeking Using Search Engines and Social Networks. *4th Int'l AAAI Conference on Weblogs and Social Media*, (pp. 23-26). Washington.

Muller, G., & Vercouter, L. (2010, September). L.I.A.R.: Achieving Social Control in Open and Decentralised Multi-Agent Systems. *Applied Artificial Intelligence, 24*(8), pp. 723-768.

Nigamy, K., Lafferty, J., & McCallum, A. (1999). Using Maximum Entropy for Text Classification. *Workshop on Machine Learning for Information Filtering*, (pp. 61-67).

NIST-Reuters, N. I. (2013, 05 03). *http://trec.nist.gov/data/reuters/reuters.html*. Retrieved 06 12, 2013, from Reuters Corpora (RCV1, RCV2, TRC2).

NIST-TREC, N. I. (2013, 06 04). *Text Retrieval Conference (TREC)*. Retrieved 06 12, 2013, from Text Retrieval Conference (TREC): http://trec.nist.gov/

Orimaye, S. O., Alhashmi, S. M., & Siew, E.-G. (2012a). Natural language opinion search on blogs. *Trends in Artificial Intelligence*, pp. 372-385.

Orimaye, S. O., Alhashmi, S. M., & Siew, E.-G. (2012b). Can Predicate-argument Structures be Used for Contextual Opinion Retrieval from Blogs? World Wide Web. *World Wide Web*, pp. 1-29.

Ounis, I., Macdonald, C., Lin, J., & Soboroff, I. (2011). Overview of the trec-2011 microblog track. *20th Text REtrieval Conference .*

Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M. M., & Gatford, M. (1995). Okapi at TREC-3. *NIST SPECIAL PUBLICATION*, pp. 109-109.

Russell, S., & Norvig, P. (2009). *Artificial Intelligence: A Modern Approach* (3 ed.). Prentice Hall.

Sabater, J., & Paolucci, M. (2007, December). On representation and aggregation of social evaluations in computational trust and reputation models. *International Journal of Approximate Reasoning, 46*(3), pp. 458-483.

Sabater, J., & Sierra, C. (2005). Review on Computational Trust and Reputation Models. *Artificial Intelligence Review, 24*, pp. 33-60.

Sabater, J., Paolucci, M., & Conte, R. (1996). Repage: REPutation and ImAGE Among Limited Autonomous Partners. *Journal of Artificial Societies and Social Simulation, 9*(2).

Sakai, T. (2004). New performance metrics based on multigrade relevance: Their application to question answering. *Proceedings of the Fourth NTCIR Workshop on Research in Information Access Technologies Information Retrieval, Question Answering and Summarization*.

Sakai, T., & Kando, N. (2007). A Further Note on Alternatives to Bpref. *ディジタル図書館*, *33* , pp. 52-59.

Sanderson, M., & Zobel, J. (2005). Information Retrieval System Evaluation: Effort, Sensitivity, and Reliability. *International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 162-169.

Seize, T. K. (1977). Student's t-test. *Southern Medical Journal, 11*(70), 1299.

Seki, Y., Evans, D. K., Ku, L.-W., Sun, L., Chen, H.-H., Kando, N., & Lin, C.-Y. (2008). Overview of multilingual opinion analysis task at NTCIR-7. *Seventh NTCIR Workshop. .*

Smucker, M. D., Allan, J., & Carterette, B. (2007, November 6). A Comparison of Statistical Significance Tests for Information Retrieval Evaluation. *Sixteenth ACM Conference on Conference on Information and Knowledge Management* (pp. 623-632). ACM.

Srinivasan, A., Teitelbaum, J., & Wu, J. (2006). DRBTS: Distributed Reputation-based Beacon Trust System. *In the 2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing (DASC'06)* (pp. 277-283). Indianapolis: IEEE.

Stone, P. J., Dunphy, D. C., & Smith, M. S. (1966, June). The General Inquirer: A Computer Approach to Content Analysis. *Journal of Regional Science, 8*(1), pp. 113-116.

Stoyanov, V., Cardie, C., & Wiebe, J. (2005). Multi-perspective question answering using the OpQA corpus. *Conference on Human Language Technology and Empirical Methods in Natural Language Processing* (pp. 923-930). Association for Computational Linguistics.

Whitley, D. (1994). A Genetic Algorithm Tutorial. *Statistics and Computing, 4*(2), pp. 65-85.

Whitley, D. (2001). An Overview of Evolutionary Algorithms: Practical Issues and Common Pitfalls. *Information and Software Technology, 14*, pp. 817-831.

Wooldridge, M. (2009). *An Introduction to Multiagent Systems.* John Wiley & Sons.

Xu, X., Tan, S., Liu, Y., Cheng, X., Lin, Z., & Guo, J. (2012, February). Find me Opinion Sources in Blogosphere: a Unified Framework for Opinionated Blog Feed Retrieval. *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining* (pp. 583-592). ACM.

Yu, B., Diermeier, D., & Kaufmann, S. (2009). *The Wal-Mart Corpus: A multi-granularity corporate opinion corpus for opinion retrieval, classification and aggregation.*

Zacharia, G., Moukas, A., & Maes, P. (2000). Collaborative reputation mechanisms for electronic marketplaces. *Decision Support Systems*, pp. 371-388.

Zacharia, O. (2000, October). Trust Management Through Reputation Mechanisms. *Applied Artificial Intelligence, 14*, pp. 881-907.

Zhang, M., & Ye, X. (2008). A Generation Model to Unify Topic Relevance. *31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 411-418). Singapore: ACM.

Zobel, J. (1998). How Reliable are The Results of Large-Scale Information Retrieval Experiments? *International ACM SIGIR Conference on Research and Development in Information Retrieval*, (pp. 307-314). ACM.

Appendix A – Dictionary of Expressions Used to Extract Features from Opinions about Electronic Games

This appendix contains the feature selected to be extracted from opinions about electronic games. Table A.1 lists all of them and their meaning:

Table A.1 – A set of feature for the domain of electronic games.

Feature	Meaning
AI	Implementation Artificial Intelligence on the game.
World	Virtual environment where the game takes place.
Controls	How the game is controlled by its users.
Graphics	Style and visual identity of the game presentation (user interface).
Characters	Playable and non-playable characters on the game.
Content	Contents of the game.
Overall game	Perception related to the game as a whole (game value, bugs, and load screens)
Combat	How combat is performed on the game (when the game features it).
Story	Story of the game and its characters.
Multiplayer	Online and local interactions between players.
Sound	Soundtrack, sound effects, and voice acting of the game.
Gameplay	How the game is played
Wireless	Wireless features of the game, mainly related to the Nintendo 3DS wireless features (street pass and spot pass).
Value	Price of game.

For each feature on this table we have created a list of expression that refers to them. This list presented on table A.2 was composed using our own experience working on the field in the past and complemented by reading reviews about electronic games and extracting expression from them.

Table A.2 – Expression used to identify each feature.

Feature	Expressions
AI	computer AI, artificial intelligence, artificial intelligences, artificially
World	world, worlds, dungeon, dungeons, overworld, overworlds, map, maps, room, rooms, scenery, sceneries, landscape, landscapes, exploration, explorations
Controls	controls, control, motion plus, motion control, motion controls, wiimote, wiimotes, wii mote, wii motes, wii remote, wii remotes, controller, controllers, nunchuck, nunchucks, control scheme, control schemes, wii zapper, wii zappers, timed button press, timed button presses, quick time event, quick time events, touch screen, touch screens, stylus, voice command, voice commands, button assignment, button assignments, touchscreen, touchscreens, sixaxis, touchpad, touchpads, mouse, keyboard
Graphics	graphics, graphic, art style, art styles, looking, presentation, visual, visuals, framerate, design style, design styles, resolution, resolutions, sprite, sprites, background, backgrounds, environment, environments, cinematic, cinematics, drawn, animation, animations, graphical, looks, models, 3D, 2D, graphically, 3D effect, 3D effects, colour palette, colour palettes, scene
Characters	characters, character, companion, companions, partner, partners, party, member, members
Content	content, quest, quests, level, levels, mission, missions, puzzle, puzzles, playthrough, playthroughs, hours, scenarios, side mission, side missions, classes, enemy, enemies, stage, stages, collectable, collectables, collectible, collectibles, community sharing, level editors, level editor, equipment, equipments, replay value, replayability, mode, modes, courses, item, items, campaign, campaigns, route, routes, boss, bosses, boss fights, boss battle, boss battles, unlockable, unlockables, static, statics, easter eggs, easter egg
Overall game	overall game, game value, the title, load time, load times, load screen, load screens, the game, glitches, glitch (es), innovation, innovations
Combat	combat, combats, battle system, combat system, attack, attacks, weapon, weapons
Story	story, stories, cut scene, cut scenes, facial expression, facial expressions, storyline, ending, storytelling, plot
Multiplayer	multiplayer, multiplayers, local multiplayer, on line multiplayer, online multiplayer, co op, another player, splitscreen, split screen
Sound	sound, soundtrack, soundtracks, sound design, voice, voice acting, audio, audios, rhythm, music, musical, sound effect, sound effects
Gameplay	gameplay, travel system, travel systems, linearity, linear, camera, cameras, game mechanic, game mechanics, learning curve, learning curves, difficulty, difficulties, mechanic, mechanics, design choice, design choices, camera work, camera works, mini game, mini games, augmented reality, physics
Wireless	wireless, streetpass, street pass, spotpass, spot pass
Value	value,price

Appendix B – Comparing Opinion Classification Strategies for Building an Opinion Corpus

Allan Diego Silva Lima, and Jaime Simão Sichman

Laboratório de Técnicas Inteligentes, Universidade de São Paulo, São Paulo, Brazil
adsl@usp.br, jaime.sichman@poli.usp.br

Abstract. During the development an opinion corpus we faced a challenge in which we needed to find a way to discover if text sentences were opinions or not. To overtake this challenge we decided to compare several text classification strategies to decide which one would be the best in order to build our opinion classifier. However, in order to create such classifier we needed to deal with some important peculiarities of this problem like, for example, to avoid as much as possible non-opinions classified as opinions. In paper we discuss our tests regarding choices and tests of the classification algorithms, the training strategies and the features definition. To evaluate the results we conducted two experiments in a domain that is not commonly addressed by opinion mining projects: Electronic Games. Our experiments showed that the Naïve Bayes classifier, trained with undersampling and n-gram = 2 (biword) had the best results for our dataset.

Keywords. Opinion Mining; Sentiment Analysis; Text Classification; Machine Learning

1. Introduction

Opinion classification is one of the main tasks of the Opinion Mining research field [8]. Intuitively this problem can be seen as the following: given a text fragment how to determinate if it is an opinion or not? As an opinion, we mean that the text fragment expresses the subjective vision of its author about a theme. For example, given the text fragment "The battery of this notebook has 6 cells", we can say that it is a fact since it is something that does not change independently of its author. However, the text fragment "The weather today is very cold" is considered as an opinion because it represents the point of view of its author. In this case, the expression "very cold" is something subject to change from people to people.

Typical opinion classification strategies try to categorize each element of a sentences' set as an opinion or not. Sometimes, algorithms for opinion classification also try to find out if the opinion is positive, neutral or negative which in the opinion mining literature is called as opinion orientation [8]. Furthermore, while developing general purpose strategies to solve the opinion classification problem, researchers usually try to evaluate their solutions by the overall accuracy of the system, i.e. the number of documents correctly classified divided by the total number of documents. Hence the classifier that is able to be more accurate in the classification of all the documents in the test set for all the classes of the problem is considered the best.

On the other hand, our problem can be intuitively described as the following: given a set of sentences, we need to discover which of them are opinions, so that we can use them to populate a corpus of opinions of a given subject. In the evaluation of the strategies to solve this problem, we have two especially relevant types of results: (1) opinions correctly classified as opinions; (2) non-opinions classified as opinions. The first type of result is the number of opinions that where correctly categorized as opinions (true positives) and the second type is basically the amount of noise (false positives) generated during the categorization. Moreover the two other types of results: non-opinions classified as non-opinions and opinions classified as non-opinions are less relevant when compared to the first two have listed.

With this paper we make a contribution to answer the following question: How to use an opinion classifier as a filter to select opinions among a set of sentences in order to create an opinion corpus? Hence we want to give more importance (higher weight) to opinions classified as opinions and, at the same time, punish classification strategies that result in many non-opinions classified as opinions; this a discussion that is not common in the literature as we will see in our related work review.

In the next section, we will formalize our problem and discuss the context of our classifier. In section 3, we will describe related work to our categorization problem. In sections 4 and 5, we will show respectively the classification algorithms and training techniques used in our experiments. Section 6 describes how we defined an evaluation methodology for the different classifiers setups that we tested. Section 7 presents the data corpus we

gathered to develop our tests while in sections 8 and 9 we show the main results we obtained with the different classifiers we developed. In section 9 we discuss the results and, finally, in section 10 we present our conclusions and future work.

2. Problem Formalization and Context

The first step of our formalization is the definition of the function **IsOpinon**, shown in Equation (1), as a function that represents the abstraction of an opinion classifier.

IsOpinon: *S → {true, false}, where S is the set of all possible sentences (i.e. text fragments).*

$$IsOpinion(s) = \begin{cases} true, if\ s\ is\ an\ opinion \\ false, otherwise \end{cases} \quad (1)$$

It is possible then to use this function to filter opinions as the following:

Opinion's filter: *Given a set of text sentences TS, where TS = { ts_1, ts_2, … , ts_n } we want to obtain a subset OP of TS, where OP = { op_1, op_2, … , op_m } such as for each op_i in OP IsOpinion(op_i) = true. Formally, OP = { op | op ∈ TS ∧ IsOpinion(op) = true }.*

In this context, this paper focuses is on how we developed and tested the function **IsOpinon** so we could use it to build our set OP. In order to do that we had to face a problem with three important particular features:

1. **Avoid non-opinions classified as opinions:** There is no way to develop a function **IsOpinion** that is 100% accurate. So during the evaluation of the many approaches to develop such a function, the amount of noise (intuitively, the percentage of non-opinions classified as opinions) generated by **IsOpinion** is an important metric to evaluate the results obtained;
2. **Offline classification:** fast computational performance is not a priority. We will use the classifier to find opinions in a set of sentences. This type of problem does not need a fast classifier unlike, for example, an opinion classifier used to track the latest opinions about a given subject at the Internet;
3. **Document granularity:** many classification strategies were developed and tested for documents with many features (one feature is usually represented by one or more words). So with small documents (in our case, text sentences are composed by a few words) they may not perform as well as they are supposed to do.

3. Related Work

With the definition and the context of our problem in mind, we decided to analyze previous work available in the literature from four different points of view: (1) the domain where the classifier was used (type and size of the documents classified); (2) the classification algorithm used; (3) the training and pre-processing policies applied; (4) the metrics used to evaluate the results.

The literature review that we have made showed the application of opinion classifiers for documents in domains such as economy news [1]; celebrity news [12]; news articles [2]; Japanese blogs [3]; books' reviews, DVDs and electronic appliances [9]; and Twitter (www.twitter.com) messages [5] which are text sentences similar in size (number of characters and words) to the sentences we want to classify. Among those projects it is important to highlight also a difference about how they work with documents, some projects developed classifiers to analyze the overall sentiment orientation (as positive, negative, neutral, etc.) of the whole document. However others projects dealt with the sentence level of granularity trying to define if each sentence on a document is an opinion and eventually trying to find its orientation.

Regarding the learning strategy used to the opinion classification we can highlight the use of the Bayesian Naïve Classifier [2], Support Vector Machines (SVM) [3, 9, 12], the Logistic Classifier [5] and Linear-chain Conditional Random Field Model together with dictionary based annotations [1]. Among the training and preprocessing policies developed to handle the data, it is possible to see that the cross-validation technique is frequently used [2, 1, 9], we could also find the application of Random Subspaces of features, where multiples classifiers are generated each of them considering only a random subset of features allied with undersampling to balance the number of documents of each class in the training dataset [9]. Furthermore, there is also a discussion concerning the improvements of the use n-grams (the number of words per feature) larger than one [5].

Our analysis of the metrics used to test and validate those projects revealed that precision, coverage and accuracy are the typically applied measures to the evaluation of the results [2, 3, 5, 9, 12]. In addition, the F-Measure, which is the harmonic average between coverage and precision, is also present in the evaluation of the results [1]. Finally, at the text classification research field there is also a discussion concerning how to select the

best features while training a classifier. In practice it means that if some of the features may not be relevant for the classification then, even without them a classifier could achieve a good accuracy. However, a detailed study in the area [4] showed that it is hard to increase the performance of the classifier by applying this strategy.

After our analyses we could see that besides classification algorithm is it also important to test variations of the training and feature generation polices. Hence we decided to limit our tests in three distinct classification algorithms (Naïve Bayes, Support Vector Machines and the Logistics Classifier). But at the same time, we could dedicate our resources to explore setups of the algorithms with cross-validation, undersampling, random subspace generation and n-gram. Notice that the Linear-chain Conditional Random Field Model was not included in this list because it was used in [1] together dictionary based annotations, which is out of the scope of our research. Finally, we defined the F-Measure as the main metric to evaluate our results it because it not only includes precision and coverage in its equation, but it also allows us to determine the weight of each of them as well. Furthermore, given our problem we are working with sentences that already have a few words, we decided to not use a feature selection policy.

4. Overview of the Classification Algorithms

In the section, we briefly present the main characteristics of the classification algorithms that we have used in our experiments.

Naïve Bayes Classification [10]**:** this technique works by computing the conditional probabilities of a word being present in a document of a given class. So each word has an estimated probability for each possible class, higher probably for a class c means a high indication that documents containing that word will be samples of the class c. In this algorithm there is the assumption that for each class, the probability a feature be part of a document is independent of the others [7].

Support Vector Machines (SMV) [6, 14]: this technique is based on the principle of structural risk minimization. The algorithm seeks for a hypothesis h such that it is possible to ensure the best true error. This can be achieved by the probability that h will make a mistake in one instance not previously seen and randomly selected. For example, d present in the training set is considered as a point in a space of dimension D, where D is the number of different features present in the training set. This technique assumes that the points (documents) present in each of the classes are linearly separable, and then it tries to find a hyperplane that can separate them.

Logistic Classification [11, 17]: this classifier tries to satisfy constraints by modeling them by uniform models. For example, in a two-class problem where 75% of the documents containing the word “feeling” are opinions, intuitively when given a sentence containing “feeling” it is possible to say that it has a 75% chance of being an opinion. However, if the sentence does not have the word “feeling” on it has a 25% chance of being an opinion. Hence, this algorithm sees each feature in the training set as a possible constraint, like “feeling” in our example, and tries to build a model that satisfies all of them.

5. Overview of the Training and Pre-processing Polices

In the section we discuss the training and pre-processing polices we chose to apply on our experiments. Given our problem, intuitively each of them has its own importance: undersamping allows us to deal with unbalanced data; n-grams allows us to compensate the small number of word in our sentences; cross-validation makes the results more close to final version of the classifier; and random subspace generation may increase the precision and the coverage of a classifier.

n-grams: N-grams are number of words that are considered as a feature. In text classification a feature is usually a word, however we can also use combinations of words. When we say that a classifier was trained using n-gram = 1 this means that every word was considered a feature, while for n-gram = 2 each pair of words was considered a feature as well. For example, given the sentence “I like this type of experience.” if n-Gram = 2 “I like”, “like this”, “this type”, “type of”, “of experience” are all considered as different features. However “I”, “like”, “this”, “type”, “of”, “experience” are considered as features as well when n-gram = 2.

Undersampling: this technique is essentially removes the documents of the most common classes in the training set. Usually, training sets have many documents of one or more classes while just a few of other classes, and this fact can introduce bias in a classifier; so before training a classifier researchers may remove documents of the most common classes.

Cross-validation: instead of splitting the labeled documents in a training set and a testing set, it is common to split all the labeled documents available in k folds of x documents each. So for each fold all other documents in the k-1 folds available are used to train a classifier and then classify the documents in the selected fold. While evaluating a classifier, this is particularly important to avoid overfitting, where the classifier only learns how to classify samples previously seen samples.

Random Subspaces: Given a set of features S = {s_0, … , s_n}, it is possible to generate a random subset of S, S' such that if s' ∈ S', s' ∈ S too. Then, we train a classifier using only the features present in S'. This approach is potentially good when for a probabilistic classifier C' generated using S' only the document classified with the higher confidence (i. e. probability) by C' are considered as an correct classification. Hence, for this approach to be applied we need to generate multiples classifiers, for example, one for each document in the testing set.

Given those pre-processing and training policies, it is possible to combine them generating many classifiers with different setups. For example, we could train a Bayesian classifier balancing the documents with undersampling, pre-processing their content generating features with one and two words (n-gram = 2), and finally train the classifier using cross-validation.

6. Overview of the Validation Metrics

The results in a two-class problem opinion classifier can have four different types as discussed before (true positives, false positives, true negatives, false negatives). Furthermore, accuracy, coverage and precision, Equations (2), (3) and (4), respectively, are basic metrics to test a classifier. In addition, F Measure, Equation (5), is a more elaborated metric evolving weights for precision and coverage.

$$accuracy = \frac{\#\ true\ positives + \#\ true\ negatives}{\#\ documents} \tag{2}$$

$$coverage = \frac{\#\ true\ positives}{\#\ true\ positives + \#\ false\ negatives} \tag{3}$$

$$precision = \frac{\#\ true\ positives}{\#\ true\ positives + \#\ false\ positives} \tag{3}$$

$$F\ Measure = \frac{1}{\alpha \frac{1}{precision} + (1-\alpha)\frac{1}{coverage}} \tag{5}$$

$$where\ 0 < \alpha < 1$$

In this particular problem there is a sensible distinction from ordinary classification problems: we want to avoid false positives, in other words, non-opinions classified as opinions. For this reason instead of using accuracy as the main evaluation parameter of our classifier we have choose the F-Measure. This is a metric where we can weigh the penalty of the false positives so it becomes more important than other type of results. For the experiments, we would like to have precision slightly more relevant than two times the coverage then, we decided to use α = 0.7.

7. Domain Experiment and Datasets

The domain we choose to develop our opinion corpus was composed of reviews about electronic games. We made this decision mainly because of two factors: (i) the large number of public reviews about electronic games available at the Internet; (ii) our literature review showed that this domain was not explored as others like reviews about electronic appliances. The first reason makes our work easy since there is plenty of data available, while the second increases the innovation and the relevance of our work since we could not find projects focusing on same domain.

After the domain has been decided, we crawled 1431 reviews about Electronic Games written in the English language over the Internet, and then we extracted and normalized (by removing capital letters) the sentences of the documents, resulting in 29165 sentences. For our first experiment we randomly selected 1000 sentences to compose the dataset used to the development of our experiments. Those sentences were manually labeled as opinions (821 samples) or non-opinions (179 samples) and used as input to all the different types of classifiers generated by the combination of algorithms, pre-processing and training techniques selected by us as we described in sections 4 and 5.

In order to have a second measure of the performance of the different setups of our research we decided to conduct a second experiment. In this experiment, we took random samples of the remaining 28165 sentences and compared the F-Measure of the best classification setups on first experiment on more time. To create the new dataset we defined that we wanted it to have enough sentences to give us approximately 95% of confidence with 5% of error rate with the new results. Then we used a standard sampling size estimation [13] to define that we need 226 samples to our second dataset. After manually labeling the sentences of the new dataset we reached a configuration were 191 of them were opinions and 35 were non opinions. It is possible to download and find more about both at https://allanlima.wordpress.com/2014/11/27/opinion-setence-dataset/.

8. Fist Experiment

In this section, we describe the performance of the most important tests we made during the development of our classifier. First we describe the performance of the best combinations between cross-validation (with folds of 10 sentences), n-gram (varying from 1 to 5), deterministic undersampling (where the first sentences of the dataset where always choose to balance the training set) and random undersampling (where the sentences to balance training data were randomly chosen in each classifier generated) with subspaces (with half of the size of the original set of features). The Naïve Bayes classifier we used was based on the description of the algorithm available in [10]. Regarding the SVM classifier we used the implementation that is available at the library LIBSVM [18] with the RBF kernel as suggested in its documentation [19], finally the Logistics classifier implementation we used is also part of a public available at http://nlp.stanford.edu/software/classifier.shtml.

The results presented in Table 1 show that a Naïve Bayes classifier trained only with cross-validation leaded to a classifier that labeled all of our examples as opinions. This happened because there were more samples of opinions in our training set. However, by preprocessing our data to balance the number of our samples with deterministic undersampling, the classifier trained with n-gram = 2 managed to reach the best results in the F-Measure. Moreover when we introduced the random subspaces to train the classifier, the results suffered degradation mainly in the coverage.

The SMV classifier had similar results for precision in the three configurations showed in Table 2, but the coverage results responsible for the fact that the setup with deterministic undersampling, cross-validation and n-gram = 1 had the best results of F-Measure. The results of the classifier with only cross-validation were close to the best F-Measure in this classifier, while the results setup with subspaces presenting once again the worst F-Measure numbers. By training a SVM classifier for our data we managed to achieve the best results with the combination of deterministic undersampling together with cross-validation as well. All the n-gram variations to this setup leaded to a scenario that covered basically all the opinions (high coverage) but with a lot of false positives compared to Naïve Bayes. Also we saw small variations in the F-Measure across all the n-gram values with n-gram = 1 performing slightly better.

Table 1. Results of the Naïve Bayes Classifier.

Naïve Bayes	n-gram	Precision	Coverage	F-Measure
Cross-validation	n-gram = 1	0.8210	1.0000	0.8676
	n-gram = 2	0.8210	1.0000	0.8676
	n-gram = 3	0.8210	1.0000	0.8676
	n-gram = 4	0.8210	1.0000	0.8676
	n-gram = 5	0.8210	1.0000	0.8676
Cross-validation + Deterministic Undersampling	n-gram = 1	0.8791	0.8770	0.8785
	n-gram = 2	0.8888	0.8758	0.8848
	n-gram = 3	0.8874	0.8733	0.8831
	n-gram = 4	0.8883	0.8721	0.8834
	n-gram = 5	0.8883	0.8721	0.8834
Random Undersampling + Random Subspace	n-gram = 1	5.7297	0.8514	0.4048
	n-gram = 2	5.7045	0.8508	0.3981
	n-gram = 3	5.7281	0.8514	0.3982
	n-gram = 4	5.7236	0.8513	0.3963
	n-gram = 5	5.7369	0.8516	0.3976

Table 2. Results of the Support Vector Machines Classifier.

SVM	n-gram	Precision	Coverage	F-Measure
Cross-validation	n-gram = 1	0.8206	0.9976	0.8668
	n-gram = 2	0.8206	0.9976	0.8668
	n-gram = 3	0.8206	0.9976	0.8668
	n-gram = 4	0.8206	0.9976	0.8668
	n-gram = 5	0.8206	0.9976	0.8668
Cross-validation + Deterministic Undersampling	n-gram = 1	0.8210	1.0000	0.8676
	n-gram = 2	0.8194	0.9890	0.8638
	n-gram = 3	0.8235	0.9890	0.8671
	n-gram = 4	0.8208	0.9988	0.8672
	n-gram = 5	0.8201	0.9939	0.8655
Random Undersampling + Random Subspace	n-gram = 1	0.8298	0.6977	0.7852
	n-gram = 2	0.8140	0.3604	0.5909
	n-gram = 3	0.8151	0.3015	0.5394
	n-gram = 4	0.8144	0.5803	0.7265
	n-gram = 5	0.8110	0.6258	0.7449

The Logistic classifier (Table 3) had the best coverage in our tests with cross-validation and n-gram = 1, but the coverage at the setup was not as high as others setups which made its performance in the F-Measure lower when compared the best results we could find. The other two setups for this classifier had lower values of F-Measure also featuring high precision; however once again the coverage prevented best results for F-Measure. Finally, compared to the previous algorithms, the Logistic Classifier showed in its best setup (with cross-validation) a good coverage but its values of precision prevented it from reaching better results in F-Measure. Moreover for n-gram > 2, this algorithm showed a decrease in coverage without increasing its precision.

Table 3. Results of the Logistics Classifier.

Logistic	n-gram	Precision	Coverage	F-Measure
Cross-validation	n-gram = 1	0.8483	0.9537	0.8774
	n-gram = 2	0.8381	0.9647	0.8724
	n-gram = 3	0.8429	0.8758	0.8525
	n-gram = 4	0.8438	0.8161	0.8353
	n-gram = 5	0.8320	0.7844	0.8172
Cross-validation + Deterministic Undersampling	n-gram = 1	0.8964	0.7065	0.8295
	n-gram = 2	0.5000	0.6711	0.5414
	n-gram = 3	0.8935	0.3374	0.5979
	n-gram = 4	0.9155	0.0792	0.2196
	n-gram = 5	0.8571	0.0073	0.0239
Random Undersampling + Random Subspace	n-gram = 1	0.8604	0.3864	0.6289
	n-gram = 2	0.8555	0.2270	0.4673
	n-gram = 3	0.7577	0.0608	0.1707
	n-gram = 4	0.7525	0.0112	0.0361
	n-gram = 5	0.8916	0.0112	0.0363

In the Fig. 1 is possible to see a comparison between the best setups for three classification algorithms we choose (Naïve Byes with Cross-validation, Deterministic Undersampling and n-gram = 2; SMV with Cross-validation, Deterministic Undersampling and n-gram = 1; Logistic Classifier with Cross-validation and n-gram = 2). Despite the similar results on our main metric (F-Measure) the difference among them on precision and coverage are easy to notice.

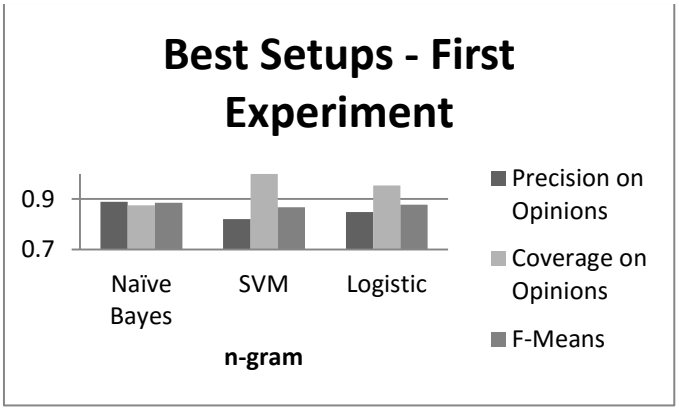


Fig 1. Results of the first experiment for the best setups of the three classifiers we tested.

In terms coverage the previous work we find in the literature indicated that Logics and SMV classifiers were better for opinion classification. However, the Bayesian classifier was still able to perform better in our metric due its high precision.

9. Second Experiment

In this experiment, we tested the best setups of each classification algorithm, as shown in Table 4. This time the Logistics classifier outperformed the Naïve Bayes Classifier by a relatively small difference compared to first experiment. One more time the Naïve Classifier showed a relatively good performance compared to others especially because of its good balance between precision and coverage. Also following the same path of the previous experiment the best SVM setup was the one that classified all the samples as opinions. In other words, it did not perform as well as we previously expected. In our second experiment all three classifiers showed similar results to the first experiment especially when it comes to balance between precision and coverage. Hence the results of our second experiment reassure what we initially found on first experiment with few variations as we address in the next section.

Table 4. Results of the Second Experiment.

Setup	n-gram	Precision	Coverage	F-Measure
F-Measure Naïve Bayes with Deterministic Undersampling	n-gram = 1	0.9061	0.8586	0.8913
	n-gram = 2	0.9101	0.8482	0.8906
	n-gram = 3	0.9091	0.8377	0.8864
	n-gram = 4	0.9143	0.8377	0.8899
	n-gram = 5	0.9143	0.8377	0.8899
SVM with Deterministic Undersampling	n-gram = 1	0.8454	0.9162	0.8655
	n-gram = 2	0.5000	0.4136	0.4705
	n-gram = 3	0.8451	1.0000	0.8863
	n-gram = 4	0.8451	1.0000	0.8863
	n-gram = 5	0.0000	0.0000	0.0000
Logistics with no Undersampling	n-gram = 1	0.8706	0.9162	0.8838
	n-gram = 2	0.8433	0.9581	0.8748
	n-gram = 3	0.8524	0.9372	0.8762
	n-gram = 4	0.8618	0.9791	0.8939
	n-gram = 5	0.8514	0.9895	0.8886

10. Discussion

In the first experiment the Naïve Byes classifier produced the best results. However in the second experiment the Logistics classifier had the best performance although by a relatively small difference compared to the first experiment. This difference can be explained by the small difference in the opinions/non opinions proportion between among both data sets. Since the second dataset had more opinions compared to first, the Logistics classifier, which had better coverage, manager outperform the Naïve Bayes by a small margin. Given the performance of both experiments, the fact that the Naïve Byes Classifier generated less false positive and

especially the small F-Means difference in last experiment we choose the Naïve Bayes classifier as the best suited algorithm to our problem.

The Naïve Bayes classifier results in the second experiment also showed a difference related to the best n-gram value. Given how close the results for all n-gram were for this classifier we believe that the small difference are acceptable and since the setup with g-gram = 2 was the best in the first experiment and the second best in the last we choose it as the value to train Naïve Bayesian Classifier.

We could not find a setup for the SMV classifier that performed as well as the others on our main metric, so in overall it was the worst of the three algorithms we have tested. Furthermore, its F-Measure results varied more than the other classifiers. We believe that these results happened because the lack features and the relatively small number of features in a sentence made it hard for this algorithm to find a good hyperplane that cloud separate both classes. Perhaps by increasing the number of sentences in the training set could make this classifier perform better in our problem.

Moreover, we were able to improve our results with the Naïve Bayes and the SMV classifier by balancing the number of samples in the training set (undersampling). However, unlike our expectations based on a related work [9], by introducing random subspaces to train the classifiers we could not improve the values of F-Measure. We believe that this happened because the subspaces generated limited even more the already small number of features available for the classifier. Intuitively, we expected this limitation to be compensated by the introduction n-grams lager than one which however was not the case. The use of undersampling was also a key pre-processing strategy specially to improve the performance of the Naïve Bayes classifier which because of its relatively simple classification algorithm tends to be very sensitive to unbalanced data. On the other hand, the Logistic classifier was less sensitive to unbalanced data.

11. Conclusions and Future Work

In this work we compared several classification strategies to decide which would be the best in order to build an opinion classifier that could generate an opinions corpus. In this problem, we faced one important and uncommon feature in the previous work: we wanted to avoid false positives as much as possible. Given that, we compared a list of algorithms and preprocessing techniques for this task. We had also to develop a proper evaluation methodology for our problem since previous work had different objectives and consequently different evaluation metrics.

After two experiments, we found that the Naïve Bayes classifier trained with undersampling and n-gram = 2 had the best performance for our opinion dataset. However, others classification strategies that we tested achieved results that were close to the best Naïve Bayes classification setting for our dataset.

As future work, we intend to develop a classifier based on a dictionary of opinion words like the SentiWordNet [15] and compare its performance with the results we presented in this paper. An opinion words dictionary is a collection of words that are commonly used in sentences that are opinions. Hence a classifier based on this strategy is essentially an algorithm that searches for occurrences of the words in the dictionary on the sentences to be classified.

Finally, given that our classifier is finished, the next step in the development of our opinion relevance database is the data pooling, associating relevance information to the opinions set generated by the classifier. This was the main motivation behind of the development of experiments presented in this paper. Once our corpus is ready we will use it to test and refine an opinion relevance model currently in development by us.

References

1. Breck, E., Choi, Y., Cardie, C.: Identifying expressions of opinion in context. In: 20th International Joint Conference on Artificial Intelligence, pp. 2683-2688, Morgan Kaufmann Publishers Inc. (2007)
2. Brew, A., Greene, D., Cunningham, P.: Using crowdsourcing and active learning to track sentiment in online media. In: European Conference on Artificial Intelligence, pp.145-150 (2010)
3. Furuse, O., Hiroshima, N., Yamad, S., Kataoka, R. Opinion sentence search engine on open-domain blog. In: 20th International Joint Conference on Artificial Intelligence (IJCAI2007), pp.2760-2765 (2007).
4. Forman, G. An extensive empirical study of feature selection metrics for text classification In: The Journal of Machine Learning Research 3, pp.1289-1305 (2003).
5. Go, A., Bhayani, R. Huang, L. Twitter Sentiment Classification using Distant Supervision In: CS224N Project Report, Stanford, pp.1-12 (2009).
6. Joachims, T. Text categorization with support vector machines: Learning with many relevant features. In: Springer Berlin Heidelberg, pp. 137-142 (1998).
7. Li, Y. H., Jain, A. K. Classification of text documents. In: The Computer Journal 41, no. 8, pp.537-546 (1998).
8. Liu, B. Opinion Mining. In: Encyclopedia of Database Systems. Springer. (2008).

9. Li, S., Wang, Z., Zhou, G., Lee, S. Y. M. Semi-supervised learning for imbalanced sentiment classification. In: Proceedings of the 22nd Second International Joint Conference on Artificial Intelligence-Volume Volume Three, AAAI Press, pp. 1826-1831 (2011).
10. Manning, C., Raghavan, P., Schütze, H. Introduction to information retrieval. Vol. 1. Cambridge: Cambridge University Press (2008).
11. Nigamy, K., Lafferty, J., McCallum, A. Using maximum entropy for text classification. In: Workshop on Machine Learning for Information Filtering, vol. 1, pp. 61-67 (1999).
12. Tan, C., Lee, L., Tang, J., Jiang, L., Zhou, M., Li, P. User-level sentiment analysis incorporating social networks. In: 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '11). ACM, New York, NY, USA, pp.1397-1405 (2011).
13. M. Natrella. NIST/SEMATECH e-Handbook of Statistical Methods. (2010).
14. Hearst, M. A., Dumais, S. T., Osman, E., Platt, J., Scholkopf, B. Support vector machines. Intelligent Systems and their Applications, IEEE 13, no. 4, pp 18-28 (1998).
15. Baccianella, S., Esuli, A., Sebastiani, F. SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining. In LREC, Vol. 10, pp. 2200-2204 (2010).
16. Forman, G.. An extensive empirical study of feature selection metrics for text classification. In: The Journal of Machine Learning Research 3, pp. 1289-1305 (2003).
17. Chang, C.-C., Lin, C.-J. LIBSVM: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology (TIST) 2, no. 3, pp.1-27 (2011).
18. Hsu, C.-W., Chang, C.-C., Lin, C.-J. A practical guide to support vector classification. (2003).

Appendix C – Collecting Data from Social Networks for our Experiments

The first step involved in creating an opinion corpus is the decision of the source from where we would like to collect the data. Initially, popular social networks such as Twitter and Facebook popped out as the obvious candidates. However, they have much more information then we could handle with our limited resources. For instance, creating a graph of all Twitter users to compute the Network Distance among two of them would cost a lot of time and require a volume of data to be indexed that extended the one available during this research. Furthermore, those sites have usually very restrictive terms on how the data available on the site (even public data) can be used and redistributed. Therefore, it would be difficult to release a corpus based on data collected from them.

On popular social networks, even for specific domains, we would have to collect a lot of opinions; hence, we decided to focus on relatively small networks. Then we started to look into small or specific domain networks on the Internet.

We call small networks sites that still have plenty of opinions but in a volume that was feasible to collect. Those networks are usually focused on a few domains and have less than a million users. We had as samples the network from SORC-G01 gamrConnect (on the domain of electronic games); the networks about Animes and Mangas My Anime List and Anime Planet; the Brazilian networks Skoob and Filmow about books and movies, respectively.

These networks have also a very important advantage: their moderators and, in some cases, even their owners are accessible. So it is possible to discuss the experiment with them and eventually get formal permission to recruit volunteers from their forums. Once we decided to perform the first experiments on the domain of electronic games for the reasons mentioned in section 5.3.1, we created a profile in the social gamrConnect and started to interact and observe the users to better understand their behavior and also gain their trust. A few months later, we approached their moderators and discussed all the important aspects of our

experiment with them. After a few messages, we managed to receive permission from the leader of the moderation team to recruit users on their forum.

Given the relative small size of the network, it was easy to collect opinions from all the reviews available on the site at that time. We could also index the profiles of the users in order to create a graph with connections representing the friendship relation between them. The creation of our first corpus (SORC-G01) can be considered successful, since we managed to collect enough opinion judgments to fulfill 50 test cases in around a month. Overall, our call for volunteers was very well perceived and did not disrupt the everyday routine of the site by any means.

After the success of our first experiment results, showing evidence that customization can be important to rank opinions, we then decided to try a second experiment, a task that was not originally scheduled for this work. For similar reasons of the first choice, this time we selected the domain of Animes. Initially, we expected this experiment to be easier to perform, as we had already the gained experience of the first experiment to help us; we could use the success of our first experiment to get around any distrust that the moderators could eventually have with our work; and we could also use our previous success to persuade more volunteers to judge opinions.

First, we elected the social network My Anime List and shortly we started to discuss our experiment with the moderators of their forums. As expected, they were very kind and helpful, which gave us confidence to advance and contact one of the highest members on the site management hierarchy. Since the negotiations were going well at the beginning, we even felt confident to create a functional prototype of the site that we would use to collect the relevance judgments as shown in Figure C.1. This prototype we filled with 303,337 publicly available opinions about the top 350 rated Animes by My Anime List users. But unfortunately, our final negotiations with the high members on the site management hierarchy did not go well, resulting in a denial on the permission to recruit volunteers on their forums to judge public opinions available on the site.

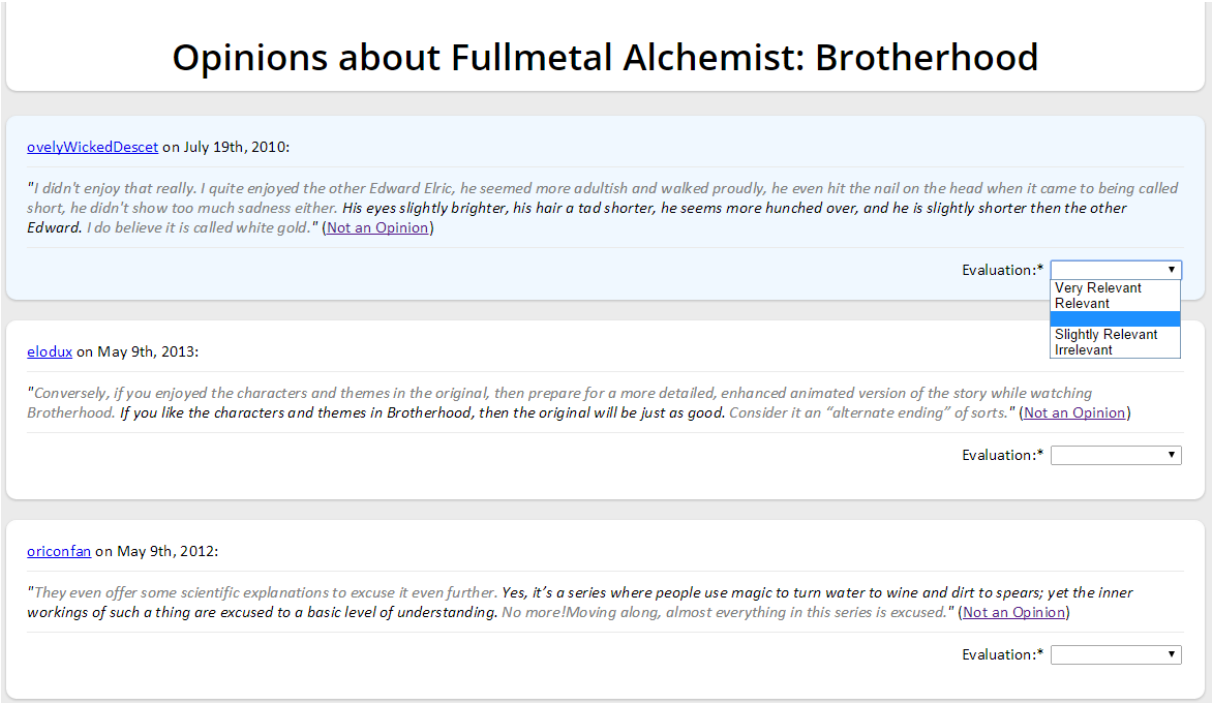


Figure C.1 – Picture showing our functional prototype for collecting relevance judgments from My Anime List Users.

After this unsuccessful negotiation with the members of My Anime List, we decided to try choosing another network for our second experiment. This time, we selected the Anime Planet social network. Initially, we collected all the public opinions available on reviews about the top 300 rated Animes on the site, resulting in a collection of 67,855 opinions. After a small study about the size and content of the data, we decided to start to contact the moderators of the site. Once again they were kind and helpful. We then started to contact people from the high management hierarchy of the site until a point in which we only needed permission from the owner of the site to start the experiment. After many flawed attempts to contact the owner, we finally got to a response from her that, unfortunately, resulted in a new denial to proceed with our experiment on her network.

These reports illustrate the effort we made to perform a second experiment for our study. After that, we researched and tried to contact representatives of other social networks, even in different domains than Anime, but so far we were not able to get permission to start a second experiment from none of them.

After the problems we have faced to create a second Social Opinion Relevance Corpus, we decided to look into new possibilities to perform a new experiment. Recently, Facebook introduced a new feature on its site where people are now able to express their opinions on some of the pages on the sites. Notably, the pages where options can be expressed on reviews are mostly about touristic places. Figure 2 shows a sample of a popular opinion (liked by many people) on the page about the famous touristic attraction “Niagara Falls”.

Figure 2 – A popular opinion about the Niagara Falls.

We believe that this new Facebook feature can be used by us in the future to create a new corpus and then test our model by performing a similar experiment as we did with SORC-G01. By the time this thesis was written, we were collecting public opinions available on a selected list of pages from Facebook and starting to study them. But so far we have identified two important problems:

1) We still have restrictions to redistribute and even to use the data on the network for scientific experiments. We are searching for ways allowed by their terms of service to at least use the opinions on the site on an experiment;

2) There are not so many opinions, compared to the ones from previous networks, on the Facebook review pages. In addition, many of the opinions are simple expressions such as “Very nice” and “Beautiful”. Perhaps this happened because people are still getting used to this new feature.