Tag-based Recommender System

Master's Thesis, 1st February 2011

Jenna Willis

Tag-based Recommender System

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE
TRACK INFORMATION ARCHITECTURE

by

Jenna Willis
born in Pietermaritzburg, South Africa

Web Information Systems Group
Department of Software Technology
Faculty EEMCS, Delft University of Technology
Delft, the Netherlands
http://www.wis.ewi.tudelft.nl

Tag-based Recommender System

Author:	Jenna Willis
Student id:	1134310
Email:	willis.jf@gmail.com

Abstract

Organizers of STRP Art and Technology Festival want to enhance the festival experience of visitors whilst also learning more about these visitors. The proposed solution is a tag-based recommender system, where the feedback received from visitors will allow STRP to learn more about how visitors perceive art pieces and in turn provide visitors with recommendations of other art pieces to view, at the festival. During the course of this thesis, we first explore how we can learn about the preferences of visitors using the tags they contribute, paired with a rating for the art piece. We do this by investigating a semantic mapping tool, Relco from TU Eindhoven, which enables us to map tags to concepts from a vocabulary, with the help of a lexical ontology such as WordNet. We experiment with the stemming of tags before using them in string matching algorithms. Further, we investigate how influential the vocabulary is onto which we map tags, when using Relco. Finally, we evaluate recommendation algorithms. We explore collaborative, content-based and hybrid forms of recommendations. We conclude that, in this context, content-based recommendation algorithms perform the most accurately and consistently. We also conclude that the semantic extension in a tag-based recommendation algorithm enables us to accurately profile a visitor and art piece with a finite number of concepts.

A synonym is a word you use
when you can’t spell the word you first thought of.

Burt F. Bacharach

Acknowledgements

Acknowledgements must be paid to:

Geert-Jan Houben
Kees van der Sluis
STRP Festival
Laura Hollink and Hans Geers

Thank-you:

Dik-Jan Wisse
Michel van Tol
members of the WIS-group, and
fellow occupants of HB 08.160

And to my Mom and Dad: *Thank-you, for everything!*

Jenna Willis
Delft, the Netherlands
1st February 2011

Contents

Chapter 1

Introduction

STRP Art and Technology Festival[1] is an exhibition of new media art that takes place annually in Eindhoven, the Netherlands. New media art can be described as any artwork that utilizes new media technologies such as digital art, computer graphics, computer animation, virtual art, Internet art, interactive art technologies, computer robotics, and art as biotechnology[2]. Tribe and Jana [30] describe new media art as "projects that make use of emerging media technologies and are concerned with the cultural, political, and aesthetic possibilities of these tools".

The organizers of STRP have the goal to "technologize" the festival and increase the level of interaction between visitors, and the pieces of art on exhibition or the festival. Further, STRP organizers are aware of their lack of knowledge about festival attendees and thus they desire to learn more about their target group. Ultimately, whichever solution is implemented to achieve the previously mentioned aspirations of the STRP Festival, this solution must enhance the festival experience of visitors.

One of the proposed solutions was a system that would allow visitors to provide feedback about art pieces they viewed and in turn receive recommendations of other art pieces to view at the festival; whilst at the same time, the organizers of STRP Festival would learn about the preferences of visitors and their perception of the art pieces on exhibition.

As STRP organizers wanted to learn what visitors thought of the art pieces, an option was to allow visitors the opportunity to provide ratings for the art pieces on exhibition. This would however only give an indication as to whether a visitor liked a piece or not. STRP also wanted to ascertain what a visitor perceived, understood or felt emotionally, when viewing an art piece. In order to ascertain this type of information, the idea was that visitors could provide tags along with a rating, for art pieces they viewed. Tagging is, traditionally, a means to organize content for future navigation, filtering or search [12]. When a number of people provide tags for content, even content they do not own, there is talk of collaborative tagging. The value of collaborative tagging is that when we aggregate individual tag contributions, we can create knowledge that a single individual would not be able to provide on their own. Gruber [15] called this "collective intelligence" or "wisdom of the crowd". In

[1]STRP Festival: http://www.strp.nl/strp/content/index

[2]New media art, Wikipedia, http://en.wikipedia.org/wiki/New_media_art, retrieved December 2010

our case, the wisdom of the crowd would be how visitors to STRP Festival categorize different art pieces, what emotions they feel when viewing an artwork etc.

Collaborative tagging encompasses a problem: The vocabulary problem. This is where different users refer to the same object with different terms [11] or users provide different terms that actually have the same meaning. Literature has shown that collaborative tagging produces ambiguity, inconsistency and synonymy [12, 21, 28].

We therefore wanted to add meaning to tags, so that we would be able to not only compare and organize the tags, but also better understand what the visitor means. The meaning of the tags also had to be machine-readable to that we could automatically organize the tags. Passant and Laublet [28] presented the idea of assigning meaning to tags by establishing a link between the tag and its meaning, using the semantic web. Therefore, the meaning of the tag is defined as a URI, and this URI is a concept from a semantic database. Examples of such resources are DBpedia[3], Geonames[4] and WordNet[5].

STRP Festival, therefore needed a recommender system that would establish the meaning of the tags that were provided by users, and use these meanings and the contributed ratings to formulate recommendations of other art pieces.

1.1 Research objectives

Given this background and motivation for a STRP Recommender System that will use the ratings and tags contributed by visitors to formulate recommendations, we formulated the following research question:

Does the incorporation of semantic technology in recommender systems improve the accuracy of recommendations?

Before we could effectively answer the research question, we first needed to carry out a number of tasks. We needed to design the STRP Recommender System, and then establish how we could ascertain the meaning of tags and organize them, accordingly. Finally, we had to incorporate the meaning of the tags into the recommendation formulation process. We formulated these tasks as the following sub-questions:

1. Can we design a recommender system that fits in the context of the STRP Festival?

2. Is it possible to establish relations between tags and ontological concepts?

3. Can we use semantic technology in formulating recommendations?

[3]DBpedia: `http://dbpedia.org`
[4]Geonames: `http://geonames.org`
[5]WordNet: `http://wordnet.princeton.edu/`

1.2 Outline

The rest of this thesis is divided into four parts. The first three parts each answer a sub-question that was introduced above. Part I details the design of the STRP Recommender System: Chapter 2 elaborates on the context of STRP Festival and the proposed solution we have briefly introduced in this chapter, and then in Chapter 3 we present a detailed description of the STRP Recommender System.

In Part II we tackle the second sub-question; the possibility of mapping a tag to an ontological concept. In Chapter 4 we present Relco, a tool for relating tags to concepts, and following that, in Chapter 5 we discuss the semantic databases necessary for achieving this. Finally, we evaluate how well Relco is able to do this, in Chapter 6.

Part III deals with the third sub-question of incorporating the semantic meaning of tags in the formulation of recommendations. We begin by first presenting background information about recommendation algorithms, in Chapter 7. We detail possible recommendation algorithms for STRP, in Chapter 8. In Chapter 9 we present our evaluation of these recommendation algorithms, in the context of STRP Recommender System.

Part IV is our concluding part, where we, in Chapter 10, present our conclusions to this thesis and future work.

Part I

Design of STRP Recommender System

Chapter 2

Proposed Solution

This chapter introduces the solution proposed to STRP Festival, to enable festival organizers the means to achieve their aspirations, that were detailed in Chapter 1. In Section 2.1 we discuss the context and environment in which the proposed solution will be used. Following this, we will expand on the proposed solution in Section 2.2.

2.1 Context of STRP Festival

The general idea was a system whereby visitors could provide feedback about the pieces of art that they viewed and receive recommendations of other pieces to view, at the festival. Figure 2.1, the context diagram of the STRP Recommender System, illustrates that feedback in the form of tags and ratings will act as input in the system and the output will be recommendations.

Tags are keywords or annotations that provide meta-data about the object being tagged [12]. **Ratings**, that are provided by the visitor, measure the quality of the artwork from their perspective and are provided in the form of a nominal scale of natural numbers from 1 to 5.

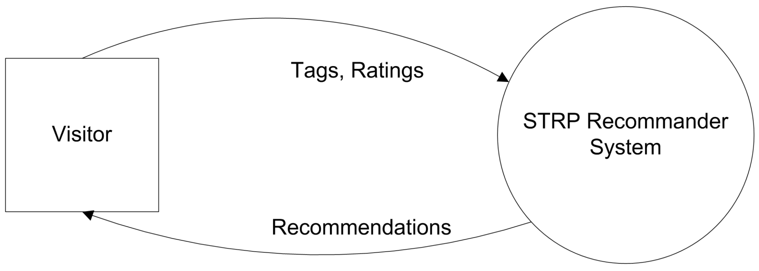

Figure 2.1: Context diagram of STRP Recommender System

The context in which this system will exist is the STRP Art and Technology Festival:

- Visitors purchase tickets for the festival on-line and create a visitor profile, before visiting the festival.

- Once at the festival, visitors can identify themselves to the system using RFID-chip technology.
  - **Radio Frequency Identification** (RFID) is technology that uses radio waves to identify unique objects [8]. This identification process occurs without contact between the transponder and the reader occurring (contactless).
  - As visitors are equipped with RFID-chips (transponders), they are identified when they bring their RFID-chips near the reader.
- Under each art piece there is a touch screen and RFID-chip reader.
  - Using RFID technology visitors are identified at the touch screens.
  - Visitors are then able to contribute tags and a rating for the corresponding piece of art. This contribution is linked not only to this particular artwork but also to the visitor that was identified at the touch screen.
- Finally, visitors can receive recommendations of other art pieces that they have not yet viewed. These recommendations are based on the accumulated contributions of the visitor and other visitors at the festival.

2.2 Proposed Solution: STRP Recommender System

Following the scenario described in Section 2.1, we have identified processes and information models that are present in the STRP Recommender System. These are illustrated in the dataflow diagram of Figure 2.2:

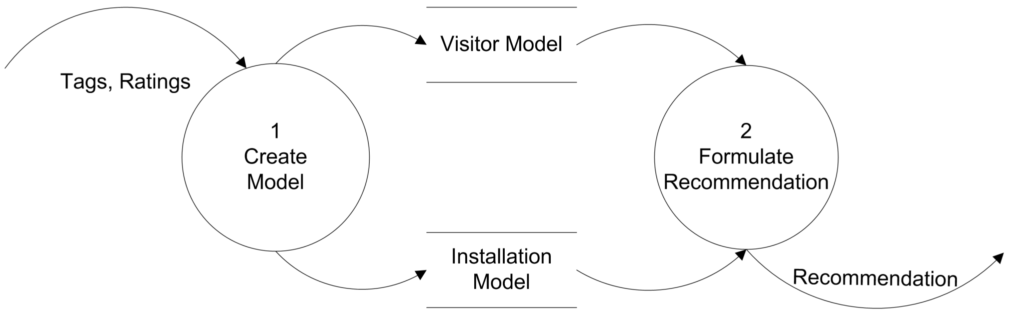

Figure 2.2: Dataflow diagram of the STRP Recommender System

Visitor Models (VM) will catalog the preferences of a visitor. This will be based on the contributions of a visitor to their profile directly, or indirectly through providing tags and ratings of art pieces.

Installations Models (IM) will define information about the art piece.

Model Creator (MC) is an application responsible for creating the visitor models and the installation models. It has the following two processes:

- **Visitor Model Creator** (VMC) will use tags and ratings to create a visitor model.
- **Installation Model Creator** (IMC) will generate tags from text resources that are used to create installation models.

Recommendation Formulation (RF) is the process responsible for formulating recommendations of art pieces for a visitor. The RF process will use IMs and VMs to formulate these recommendations.

This chapter has presented a brief explanation of the context of the STRP Festival and the STRP Recommender System itself. In the following chapter we will discuss each of the processes and information models that we have identified, in more detail.

Chapter 3

STRP Recommender System Design

In Chapter 2 we elaborated on the context of STRP Festival and introduced information models and processes of the STRP Recommender System (RS). In this chapter we will discuss the information models and the dataflow of the processes, in more detail.

In Section 3.1 we present an overview of the dataflow of STRP RS. Following that, in Section 3.2 we introduce the STRPValues and EmotionValues Vocabularies that are present in the information models of the STRP RS. In Section 3.3 we detail the installation model and how it is created, and in Section 3.4 we describe the visitor model and the visitor model creation process. Finally, in Section 3.5, we discuss the Recommendation Formulation Process.

3.1 Overview of the STRP Recommender System

In the previous chapter, we identified that we will need two kinds of information models in the STRP RS: the Visitor Model (VM) and Installation Model (IM). We also established that there were processes for creating VMs and IMs, and for formulating recommendations.

There are a number of elements in this system which are important throughout this thesis, and therefore need to be properly defined:

Tag is a keyword or annotation referring to a particular art piece. Tags can be linked to art pieces, or visitors and art pieces. We use these tags for creating visitor and installation models. We have different means for obtaining tags; either using manual input from visitors or using automatic generation of tags from text.

Concept is an instance from a semantic ontology and is identified using a URI [3, 13]. The URI can be dereferenced and contains information about that particular instance of the ontology. Further, the URI is the global identifier of the instance or, in our case, the concept. Therefore, using this URI reference will avoid confusion as to what we mean when we assign a concept to a visitor or installation model. In STRP RS, tags are used to identify which concepts should be assigned to an installation model or visitor model. Each concept possesses a label in the form of a literal or string term. We often display the literal so that the concept is human readable.

Process [1] is a sequence of events that achieve some kind of goal. In this system we consider goals to be the creation of visitor or installation models, the collection of tags, and the formulation of recommendations.

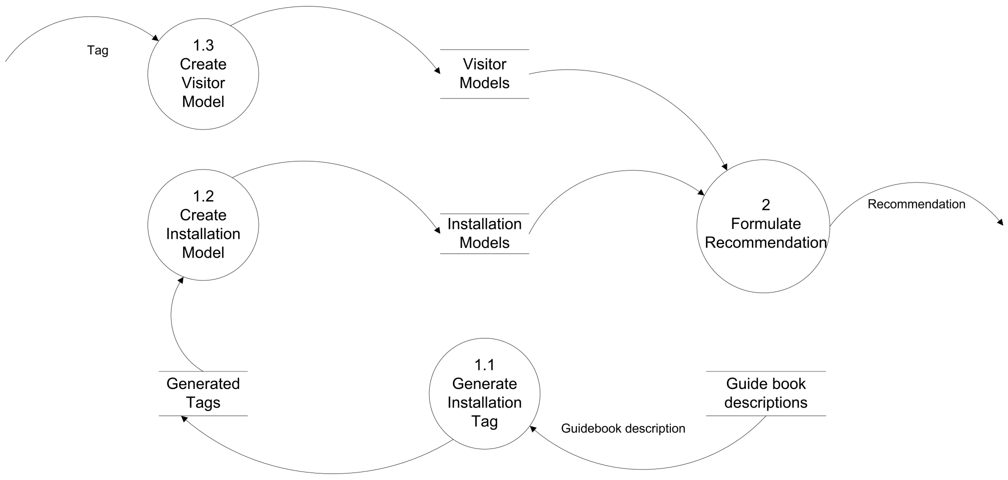

Figure 3.1: Expanded dataflow diagram of STRP Recommender System

Within STRP RS, we identified the following four necessary processes that are illustrated in Figure 3.1, a dataflow diagram of STRP RS:

Process 1.1: Generate Installation Tag is the process whereby tags used in installation model creation are generated from text.

Process 1.2: Create Installation Model is the process whereby tags are related to concepts and a IM incorporating these concepts is created. It can be automated or it can be manually created.

Process 1.3: Create Visitor Model is the process of creating a VM from the input of a user in the form of tags and ratings.

Process 2: Formulate Recommendation is the process that facilitates the formulation of recommendations, using VMs and IMs.

3.2 STRPValues and EmotionValues

In the previous section, we mentioned that IMs and VMs consist, amongst other things, of concepts. In the context of STRP RS, we define these sets of concepts as vocabularies and not taxonomies. A taxonomy can be hierarchical and thus a taxonomy limits the combinations of categories too which an object can belong [16].

[1]Process:: "a series of actions or operations conducing to an end". Merriam-Webster, `http://www.merriam-webster.com/dictionary/process`, retrieved December 2010.

Using a taxonomy in the environment of an art festival would presume we already know all possible combinations of concepts an art piece will possess in the future, and all combinations of visitor preferences possible. We therefore opted for implementing a flat vocabulary for describing an art piece or visitor.

We established a set of concepts, called **STRPValues**, that describe aspects present in art pieces and this vocabulary is shown in Figure 3.2. This vocabulary was created by asking STRP Festival organizers to identify aspects present in the art pieces to be exhibited. These aspects describe the physical appearance of art pieces (i.e. space), the technology used in art pieces (i.e. light, high-voltage, neural), the theme of art pieces (i.e. animals, urban landscape), or the level of interaction required from the visitor (i.e. experience, interact, play). Figure 3.2 is a visualization of STRPValues using Gruff. The three clusters of STRPValues occurs purely for aesthetic reasons. Gruff is a product from AllegroGraph, URL: http://www.franz.com/agraph/gruff/.

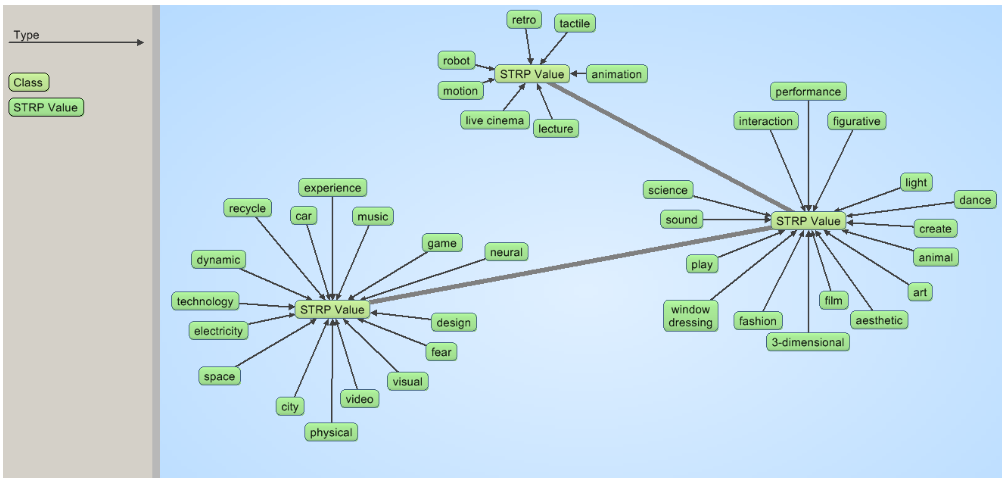

Figure 3.2: STRPValues Vocabulary

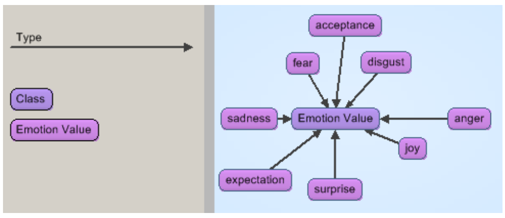

Figure 3.3: EmotionValues Vocabulary

Concepts of STRPValues describe objective aspects of art pieces. The vocabulary misses emotional descriptions. We therefore also created a vocabulary of emotions, the **EmotionValues,** to describe the emotions portrayed by art pieces or the emotions felt by visitors. This set of emotions was derived from Plutchik's "Wheel of Emotions"[2], and can be viewed in Figure 3.3.

[2]The Nature of Emotions, http://www.fractal.org/Bewustzijns-Besturings-Model/Nature-of-emotions.htm, retrieved 09 December 2010.

During the Create Installation Model and Create Visitor Model processes we derive the STRPValues and EmotionValues concepts to assign a IM or VM, from tags. A tool to facilitate this mapping is necessary.

3.3 Installation Model

The IM contains information about the art piece such as artist, date of creation, and most importantly the concepts that the art piece encompasses. We needed to ascertain concepts, in order to be able create the IM. Our first approach for gathering concepts was to simply ask the STRP organizers to assign concepts from the STRPValues Vocabulary. This is of course not a subjective method for gathering concepts as STRP Festival organizers are in this case considered to be experts. We cannot establish firstly, if their opinions are correct and secondly, if their opinions reflect that of the average visitor to the festival.

Figure 3.4 presents the dataflow diagram of IM creation. In this scenario, concepts are derived from tags in **process 1.2.1:Map to Concept**, and these tags are generated in **process 1.1:Generate Installation Tag** from guidebook descriptions. The concepts assigned to the IM are decided by aggregating all concepts that result from the tags that were generated by the guidebook description of that particular artwork. The aggregation of concepts occurs in **process 1.2.2:Aggregate Concepts**.

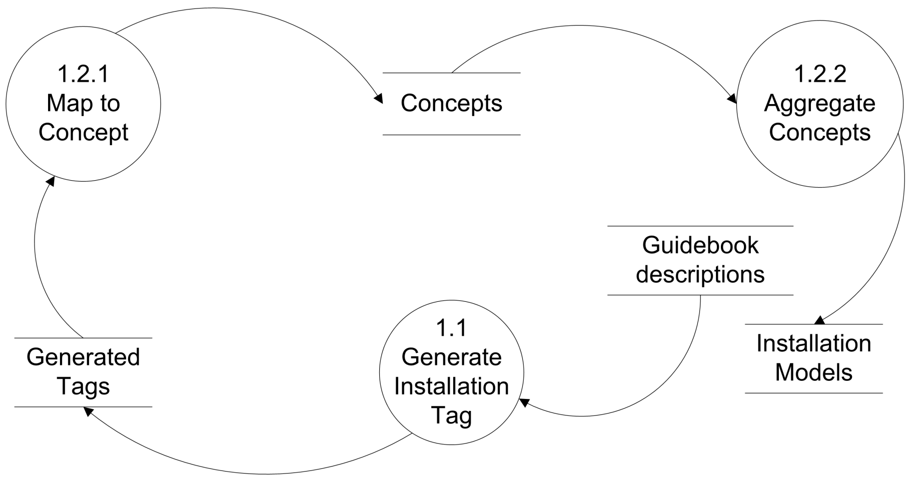

Figure 3.4: Dataflow diagram of the Installation Model Creation

Term Frequency - Inverse Document Frequency

In the scenario of deriving concepts from tags, we have mentioned that there are a number of methods for sourcing tags for the IM. When we have text describing the art piece for which we desire to generate tags, it is convenient to use methods of tag generation that are based on the terms found in the text. There are two main methods of this kind of generation: **Term Frequency (TF)** and **Term Frequency-Inverse Document Frequency** (TF-IDF) [20, 27].

In both methods the frequency of each term (t_i) in each document (d_j) is calculated. We calculate each term frequency (tf_{ij}) as follows (Equation 3.3.1):

$$tf_{ij} = \frac{n_{ij}}{\mid d_j \mid} \tag{3.3.1}$$

where n_{ij} is the number of times that term t_i occurs in document d_j, and $\mid d_j \mid$ is the total number of terms in document d_j.

Inverse Document Frequency (idf_i) for term t_i is calculated as follows (Equation 3.3.2):

$$idf_i = \log\left(\frac{N}{n_i}\right) \tag{3.3.2}$$

where n_i is the total number of documents containing the term t_i and N is the total number of documents in the collection. The product of tf_{ij} and idf_i provides the TF-IDF value of each term in a document.

Using TF we can select the n highest term frequencies (tf_{ij}), whilst in the case of idf_i we can either calculate the idf_i for the n highest tf_{ij} or for all terms.

The terms in the guidebook descriptions that are selected when using TF tag generation are the terms that occur most frequently in that particular document. Whilst the terms that are selected when using TF-IDF are the terms that are the most unique in the guidebook description compared to all terms in all other guidebook descriptions. If we want to select terms that are most frequent in that particular document, we use TF, whereas when we want the most unique term, or the term that distinguishes this document from the rest, we use TF-IDF.

3.4 Visitor Model

Figure 3.5 describes the processes involved in creating a VM where the most important information about visitors can be found. The VM not only includes personal details about the visitor such as name, contact details and age, but also the "likes" of the visitor will be stored in the VM as concepts.

Concepts of STRPValues and EmotionValues will be derived from the tags in **process 1.3.1:Map to Concept**. The resulting concepts will be aggregated and assigned to the VM, in **process 1.3.2:Aggregate Concepts**. In the context of STRP RS, the method we choose for the collection of tags, for VM creation, is **free-tagging**.

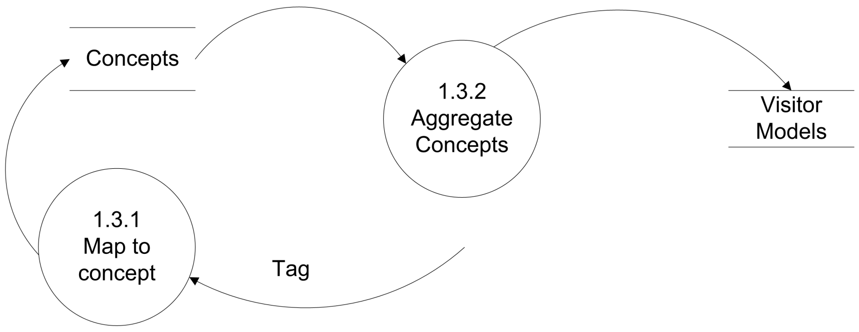

Figure 3.5: Dataflow diagram of the Visitor Model Creation

3.4.1 Free-Tagging

Tagging is defined as the process of annotating content with keywords [12, 19, 21]. Tagging emerged with the advent of Web 2.0, when users acquired the functionalities to assign metadata to content, even if it was not their own [19, 21, 28]. Conventionally, tagging is used to organize content so that it can be found again by the tagger and others [12]. We utilize tagging as the Golder and Huberman, and Kim et al. [12, 19] recognized that tagging achieved the following purposes:

1. identifying who or what the content is about,
2. identifying what it is (e.g. article, blog, book),
3. identifying who owns it,
4. identifying which categories the content fits into,
5. identifying qualities or characteristics of the content (e.g. funny),
6. task organizing (e.g. toread, jobsearch)
7. opinion,
8. self-reference (e.g. mycomment, mystuff), and
9. social views.

We recognize that many of these purposes describe aspects of a visitor profile that we want to describe. We what to identify what type of content a visitor likes (2.), what categories of art the visitor likes (4.), which quality or characteristics of art pieces appeal to the visitor (5.), the opinion of the visitor (7.), and the social views of a visitor (9.).

When a number of people provide tags for content that is not their own, collaborative tagging occurs [12]. In the STRP RS visitors, who do not own the artworks, will be providing tags for them. Li and Lu [21] identified a number of different approaches to collaborative tagging. Amongst them is the ontology approach, where taggers can only provide tags that adhere to a formal taxonomy. The advantage to this approach is that ambiguity and inconsistency amongst tags can be mitigated. The disadvantage to ontology approaches is that the tags are limited by the ontology. In the scenario of an art festival where we are wanting to ascertain how the visitor really perceives the art an ontology approach would influence this perception. We therefore choose to implement a system of free-tagging, where the visitor has no restriction or coaching in the keywords they provide.

3.5 Recommendation Formulation

Using the VMs and IMs created in the VM and IM creation processes, we are able to formulate recommendations. The dataflow of the recommendation formulation process is depicted in Figure 3.6. The essence of this process is to compare the visitor model, for which a recommendation is being formulated, with the installation models of the art pieces. An example of a method of recommendation formulation is as follows: The art pieces that have the most concepts in common with the visitor model will be included in the recommendation, provided the visitor has not already viewed these art pieces and contributed tags for them.

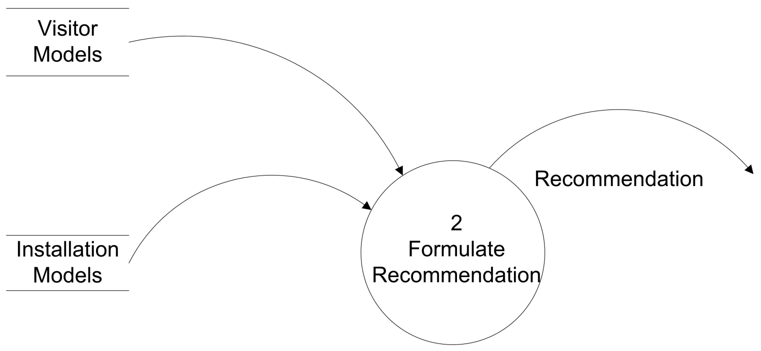

Figure 3.6: Dataflow diagram of Recommendation Formulation

In this part we have described the processes present in STRP RS. A common re-occurring process is the Map to Concept process, used for relating tags to concepts from the STRPValues and EmotionValues Vocabularies. In the following part we explore this process in detail. Following that, in Part III, we investigate recommendation formulation in more depth.

Part II

Relating Tags to Ontological Concepts

Chapter 4

Tag-To-Concept:Relco

In Part I we presented the design of the STRP Recommender System where we also discussed that it was necessary to implement a technology that would enable us to map tags to concepts defined in a vocabulary, in this case STRPValues and EmotionValues. In this part, we explore the process of mapping tags to concepts of a vocabulary using a mapping tool, Relco. **Relco** [4, 18] is a tool developed by TU Eindhoven[1] that allows us to map tags to a concept in an ontology, and will form part of visitor model creation and installation model creation processes.

In the following section we will give a brief introduction to Relco. In Section 4.2 we discuss the process flow of Relco, and in Section 4.3 we explain the different settings of Relco.

4.1 Introduction to Relco

Figure 4.1 presents the context diagram of Relco. Relco is a tool that allows you to map a tag (input) onto a concept (output) from an ontology. This mapping occurs through a combination of string matching and semantic concept matching algorithms. SeRQL[2] queries, which we define, are incorporated in the string matching and semantic concept matching algorithms. The correct specification of these queries is critical for achieving a mapping of the tag onto a concept. In order to create a mapping we need another ontology, in this case a lexical ontology, that enables us to establish the relation between the tags and the concepts from our STRPValues and EmotionValues Vocabulary.

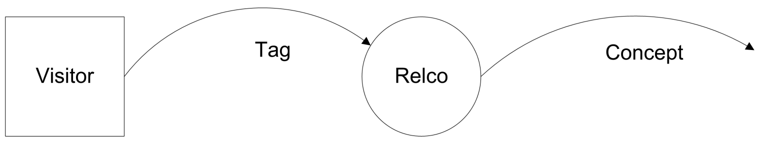

Figure 4.1: Context diagram of Relco

[1]Eindhoven University of Technology, http://w3.tue.nl/nl/

[2]OpenRDF.org Sesame, Chapter 6 SeRQL query language, http://www.openrdf.org/doc/sesame/users/ch06.html, retrieved December 2010

4.2 Process Flow of Relco

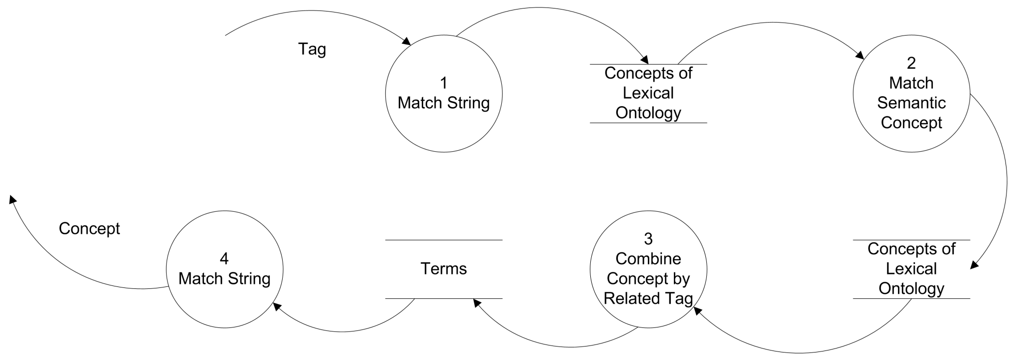
Tag
1 Match String
Concepts of Lexical Ontology
2 Match Semantic Concept
Concept
4 Match String
Terms
3 Combine Concept by Related Tag
Concepts of Lexical Ontology

Figure 4.2: Dataflow diagram of Relco

The process of mapping a tag to a concept in the STRPValues or EmotionValues Vocabulary, using a lexical ontology is depicted in Figure 4.2 and is carried out as follows:

1. The input of Relco is a string literal. Some string manipulation occurs before Relco processes it. Firstly, we remove empty spaces, special characters and numbers, and convert all letters to lowercase.

2. Using the input terms, the String Matching algorithm (**process 1:Match String**) first creates variations of the input terms and then matches these terms to concepts from the lexical ontology. The effect of this approach is that we can deal with instances of spelling mistakes in the input terms and different forms (i.e. plural, singular, past and present tenses) of the same word. Section 4.3.1 details the string matching algorithms available in Relco and their differences. The string distance measurement decides the certainty value for that concept. The smaller the string distance between the input term and the variation, the closer the certainty value is to 1.0.

3. The output of process 1:Match String consists of concepts (URIs), from the lexical ontology, where the string distance measurement between the new term and the original term falls above the threshold we have set.

4. **Process 2:Match Semantic Concept** performs an algorithm that finds other concepts from the lexical ontology that are related to the concepts that were produced in process 1: Match String. Match Semantic Concept uses object properties from the lexical ontology to find related concepts. The certainty value of the found concepts decreases the further the distance from the concepts produced in process 1.

5. The output of process 2:Match Semantic Concept consists of concepts (URIs) from the lexical ontology.

6. The objective of **process 3: Combine Concept By Related Tags** is to combine the certainty values of concepts that share the same string literals. This is

achieved through examining all literals of related concepts that resulted from process 2, removing duplicate entries whilst adjusting their certainty values.

7. The output of process 3 consists of the literals of related concepts from the lexical ontology with their combined certainty values.

8. The final process, **process 4:String Match** of the Relco process is to use a String Matching Algorithm to match the literals of process 3 to literals from concepts in the STRPValues Vocabulary. As in process 1, variations of the input terms are created whose string distance measurement falls above the established threshold, and concepts with these literals are identified in the STRPValues Vocabulary.

9. The output of Relco are the string literals from the STRPValues Vocabulary.

Table 4.1: Example of intermediate results of Relco after each process for the input term “scare”

Input Tags	Output of process 1:Match String using lexical ontology	Output of process 2:Match Semantic Concept using lexical ontology	Output of process 3:Combine Concept By Related Tags using lexical ontology	Output of process 4: Match String using STRPValues Vocabulary
scare	scare scare scare scare scare	scare panic attack fear	scare panic attack fear fearfulness fright	fear

Table 4.1 is an example of the labels of the intermediate results of Relco (1., 3., 5., 7. and 9.) where the input term (first column) is “scare”. The second column displays all labels of the concepts from the lexical ontology that match variations of the input terms. The third column contains the intermediate results after process 2:Match Semantic Concept has been carried out. In this example, three labels of concepts in the lexical ontology have been identified : “scare”, “panic attack” and “fear”. The fourth column displays the results of process 4:Combine Concepts By Related Tags; there are 5 identified concept literals of the lexical ontology identified: “scare”, “panic attack”, “fear”, “fearfulness”, and “fright”. Finally, the fifth column displays the concept literal of the STRPValues Vocabulary that was identified in process 4:String Match: “fear”.

4.3 Settings of Relco

Relco was designed to be flexible and provide operators the opportunity to define the workflow, and other specifics about the workflow to be used to map tags onto concepts of an ontology. Along with defining the pattern of string matching and semantic concept matching algorithms, we can specify the algorithm used in the string matching algorithm, and the threshold of an acceptable string distance measurement between the input tags and the intermediate term literals (Section 4.3.1). Operators can also initiate extra pre-processing of the input terms: the stemming of the input terms to their root form (Section 4.3.2). Further, operators are able to specify whether the semantic matching algorithm searches recursively through the ontology looking for related concepts, to what depth the recursion can search in the ontology (Section 4.3.3).

4.3.1 String Matching Algorithms

Relco implements an open source Java library of similarity or distance metric algorithms: SimMetrics[3]. In particular, Relco has implemented the Levenshtein Distance, Jaro Winkler and SoundEx algorithms from this library.

Levenshtein Distance

Levenshtein Distance, also known as the Edit Distance, is an algorithm that calculates the “minimum edit distance” to transform $string1$ into $string2$ [6]. This algorithm calculates the minimum number of removals, insertions, and substitutions, of characters that are necessary to convert a string into another string. The cost of each of these actions is 1. In programming, this is implemented using a matrix of $(m+1)$x$(n+1)$: m being the length of $string1$ and n the length of $string2$. Table 4.3 illustrates the example[4] where $string1$ = “$sunday$” and $string2$ = ”$saturday$”.

Table 4.3: Levenshtein Distance Matrix

		S	**A**	**T**	**U**	**R**	**D**	**A**	**Y**
	0	1	2	3	4	5	6	7	8
S	1	0	1	2	3	4	5	6	8
U	2	1	1	2	2	3	4	5	6
N	3	2	2	2	3	3	4	5	6
D	4	3	3	3	3	4	3	4	5
A	5	4	3	4	4	4	5	3	4
Y	6	5	4	4	5	5	5	4	**3**

[3]SimMetrics, Natural Language Processing Group, Department of Computer Science, University of Sheffield, http://staffwww.dcs.shef.ac.uk/people/S.Chapman/simmetrics.html, retrieved December 2010.

[4]Levenshtein Distance, Wikipedia, http://en.wikipedia.org/wiki/Levenshtein_distance retrieved December 2010.

The first step of the Levenshtein distance matrix is to fill the first row and the first column (Listing 4.1):

Listing 4.1: Listing of Levenshtein initial values in the matrix

```
for i from 0 to m
        d[i, 0] := i
for j from 0 to n
        d[0, j] := j
```

Following this, starting with the position $d[1,1]$, fill each position ($d[i,j]$) in the matrix as follows:

If $string1[i] = string2[j]$ then

$$d[i,j] = d[i-1,j-1] \tag{4.3.1}$$

Otherwise:

$$d[i,j] = min \begin{cases} d[i-1,j]+1, & //deletion \\ d[i,j-1]+1, & //insertion \\ d[i-1,j-1]+1. & //substitution \end{cases} \tag{4.3.2}$$

For each position of the matrix where $string1[i] = string2[j]$ (the same letters in these positions of the strings), the position $d[i,j]$ is filled with the minimum edit cost of position $d[i-1,j-1]$ (the previous value on the diagonal), using Equation 4.3.1. As the letters are the same no extra edit cost needs to be made, therefore the minimum edit cost is not incremented.

When $string1[i] \neq string2[j]$ (letters in that position are not identical), we calculate the cost of either inserting, deleting or substituting the letter of $string2[j]$ using Equation 4.3.2, and thus, using the method that will entail the least edit cost.

Once the entire matrix is filled the value of position $d[m,n]$ is the minimum edit distance of transforming $string1$ into $string2$. In this example (Table 4.3), the Levenshtein Distance between the strings "*sunday*" and "*saturday*" is 3.

Jaro Winkler

Jaro Winkler is an extension of Jaro Similarity; both were originally developed for record linkage purposes [6, 7]. Whereas Levenshtein Distance takes into account the number of mutations to transform one string into another, Jaro and Jaro Winkler take into account the number and order of common characters in the two strings [6, 7].

The Jaro similarity measurement for strings s and t, $Jaro(s,t)$, is calculated in Equation 4.3.3 as follows:

$$\frac{1}{3}\left(\frac{|s'|}{|s|} + \frac{|t'|}{|t|} + \frac{|s'| - T_{s',t'}}{|s'|}\right) \tag{4.3.3}$$

- s and t are given.
- $s'$is the characters in s that are common with t and appear in roughly the same place. There are number of equations for measuring whether two characters appear in about the same place, differing in accuracy.

- $t'$is the same principle as s'.
- $T_{s',t'}$ is the number of transpositions of characters in s' relative to t'. A transposition of s',t' is defined as a position where $s'_i \neq t'_i$.

Jaro Winkler adjusts the weight of poorly matching strings that share a common prefix [6]. This method influence the similarity measurement positively for strings that have matching prefixes of a particular length. Jaro Winkler, $JaroWinkler(s,t)$, is calculated in Equation 4.3.4 as follows:

$$Jaro(s,t) + (prefixLength * PREFIXSCALE * (1.0 - Jaro(s,t))) \qquad (4.3.4)$$

- $prefixLength$ is the length of the common prefix at the start of the string.
- $PREFIXSCALE$ is a constant scaling factor for how much the score is adjusted upwards for having common prefixes.

SoundEx Distance Matrix

The SoundEx distance matrix measures the similarity between two strings taking into account phonetic spelling mistakes [6]. The idea of SoundEx distance is to analyze a string forming a code representation of it. The code consists of the first letter of the string and then the code representation of the the next three consonant letters of the string. Consonants which are often substituted for each other are assigned the same numerical representation. Listing 4.2 is the coding according to the SoundEx algorithm of Chapman's open source Java library, SimMetrics.

Listing 4.2: Listing of SoundEx encoding

```
The digits are based on the consonants as in the following list:
1) B,P,F,V
2) C,S,K,G,J,Q,X,Z
3) D,T
4) L
5) M,N
6) R
```

4.3.2 Stemming of Tags

Additional to basic manipulation of the input strings (lowercase, removal of special characters etc.), operators can also specify that the input terms are reduced to their stem[5] before Relco relates them to concepts in an ontology. We implemented the Porter Stemming Algorithm to realize this [29]. Porter published an algorithm that would automatically remove suffixes from English words for information retrieval purposes. The Porter Stemming Algorithm was based on Lovins' [22] approach to the stemming of English words [31]. Lovins proposed using a list to identify the longest possible

[5]Stem: the part of an inflected word that remains after the inflected part is removed <strength is the stem of strengths>; also "root". Merriam-Webster `http://www.merriam-webster.com/dictionary/stem`, retrieved December 2010.

suffix to remove from the word being stemmed if it was “context appropriate”, and then examine what acceptable ending could be given to the words (recoding), so that words with the same semantics can be matched syntactically [22].

Porter is a more simplified algorithm compared to Lovins; using less rules for the suffix removal and recoding, and handling context (semantics) matching in a single general approach [31].

We chose to implement the Porter Stemming Algorithm as it proves to work well in practice, and also for many languages other than English [31]. Further, this algorithm is effective for purposes other than information retrieval [31].

4.3.3 Recursion of Semantic Matching

The operator of Relco can also specify whether process 4:Match Semantic Concept, in Figure 4.2, where concepts resulting from process 1:Match String are related to other concepts of the ontology, will be carried out recursively or not. We wish to implement recursion in the Match Semantic Concept process in order to broaden the area of the concepts in the lexical ontology reached in this search.

Listing 4.3 is an exert from Relco code, in particular the recursive search in the Match Semantic Concept process, and it works as follows:

- For each concept, that is a result of the initial Match String process (inputCT list), we search for concepts relating to this concept.
    - We use object properties, of the lexical ontology, in SeRQL queries to find related concepts.

If the operator has stipulated that the search for related concepts must be carried out recursively, then the following occurs to the concepts that result from the SeRQL queries:

- The certainty of the concept must be strong enough so that the certainty of a concept related to this concept will still be above the threshold (line 4). A certainty value assigned to a concept which results from Match String process is a value between 0 and 1; the normalized string distance between the input term and the label of the concept [18].
- If the certainty is strong enough we compare the URI of the concept with the URIs of each concept in inputCT (line 10).

- If this concept does not already exist in the inputCT list, add this related concept to the end of the inputCT list (line 17). This creates the recursion.
    - Through Recursion we weaken the relation between the input term and the concept found. The certainty value of this concept is adjusted to its initial certainty minus the deterioration value specified by the operator: $(concept.getCertainty() - deterioration)$. This is important to do as this concept is not directly related to the input term, but is separated by a number of nodes from it. In this way we can account for the strength of relations.
    - As the concept is added to the end of inputCT list, related concepts for this concept will be found through SeRQL queries, and should they not already exist in the inputCT list, and their certainty value is strong enough, these too will be added to the end of the inputCT list.

The Match Semantic Concept function recursively searches the ontology until either the inputCT list is empty or the pre-defined maximum depth has been reached.

Listing 4.3: Listing from Match Semantic Concept process, Relco: recursive semantic search for related concepts

```
1  if(MCProperties.getRecursion()== true)
2  {
3    System.out.println("in recursion");
4    if ((concept.getCertainty()-deterioration)> threshold)
5    {
6      boolean alreadyInInput = false;
7      for (int j=0 ; j<inputCT.size(); j++)
8      {
9        // check if URI already in input, to prevent cycles
10       if(inputCT.getConcept(j).getURI()==returnConceptValue)
11       {
12         alreadyInInput = true;
13       }
14     }
15     if (alreadyInInput == false)
16     {
17       inputCT.addConcept(new Concept(returnConceptValue,(
             concept.getCertainty()-deterioration), returnVarValue
             , concept.getRelatedTags(), inputCT.getConcept(i).
             getSourceTag()));
18     }
19     else {System.out.println("alreadyInInput == TRUE");}
20   }
21 }
```

In this chapter we have discussed the workflow of Relco, and the different settings available to the operator of Relco. We have eluded to the use of a lexical ontology to relate terms and to concepts in the vocabularies. We will elaborate on our chosen lexical ontology, WordNet, in the following chapter. Following this, in Chapter 6, we will evaluate the performance of settings that we discussed in Sections 4.3.1, 4.3.2 and 4.3.3.

Chapter 5

Choice of Lexical Ontologies and Vocabularies

In Chapter 4, we mentioned that in order to relate a string term to a concept from an ontology, we need to enlist a lexical ontology and vocabularies. We chose to use WordNet[1], and the STRPValues and EmotionValues Vocabularies we defined together with STRP Festival Organizers. In Section 5.1 we discuss the data model of WordNet and the object properties we used with Relco to relate a term to a concept in WordNet, and in Section 5.2 we discuss the data model of STRPValues and Emotion Values.

5.1 WordNet

WordNet is an electronic database for the English language, that was developed and is maintained by Princeton University. The aspirations of the creators of WordNet was to create a database that could represent a language by representing lexical concepts with hierarchical relations in a network structure [10]. They based this on research of human semantic memory for learning a language. Nowadays, WordNet is a valuable tool for Information Retrieval (IR), Machine Translation, and Natural Language Processing (NLP) [2, 10, 26]. Miller describes WordNet as follows [25]: *This database links English nouns, verbs, adjectives, and adverbs to sets of synonyms that are in turn linked through semantic relations that determine word definitions*. Similarly, Fellbaum defines WordNet as *a large, manually constructed semantic network where words that are similar in meaning are interrelated* [10].

WordNet was structured as follows [25]:

- The vocabulary of a language is a set W of pairs (f, s), where **form** f is a string over a finite alphabet and **sense** s is an element from a given set of meanings.
  - Each pair of a form and a sense is a **word** in the language.
  - A **dictionary** is an alphabetical list of words.

[1]WordNet: A lexical database for English, Princeton University, http://wordnet.princeton.edu/.

- Two words that share at least one sense are **synonymous**. A core concept of WordNet is the **synset**; this is a group of words that are synonymous [2].
  - A synset can contain one or more wordsenses, however a wordsense can only belong to one synset [2].
- A word with more than one sense is **polysemous**.
- Further, a word's usage is a set C of **linguistic contexts**, in which a word can be used.
  - The syntax of the language partitions C into **syntactic context** categories. For example, subset N are nouns.
  - Within each category of syntactic context are categories of **semantic contexts**. Semantic contexts is a set of situational contexts in which a particular f can be used to express a particular s.
- The **morphology** of a language is defined by set M of relations between word forms. For example, English has the following morphology relations: inflectional, derivations and compounds.
- The **lexical semantics** of a language is defined by set S of relations between wordsenses. Examples of lexical semantic relations between wordsenses are relations such as synonymy, antonymy, hyponymy[2], hypernymy[3]. In the WordNet network, two nodes (wordsenses) are related through an arc, depicting a relation such as "IS-A-KIND-OF", "IS-A-PART-OF", "IS-AN-ANTONYM-OF", and "ENTAILS"[4] [26].

RDF/OWL representation of WordNet

Originally, the WordNet database was represented in an ASCII database [9]. W3C[5] has since created an RDF/OWL[6] representation of WordNet[7] which is available on-line. W3C describes RDF as a *general-purpose language for representing information on the Web* [13]. Information is represented in graph form, consisting of nodes and directed arcs. Two nodes that are linked by a directed arc form a **triple** [13]. A triple consists of a subject node, predicate and object node. Nodes can be URI references, blank nodes or literals, whereas predicates are URI references and

[2]Hyponym: The specific term used to designate a member of a class. X is a hyponym of Y if X is a (kind of) Y. WordNet Glossary of Terms, `http://wordnet.princeton.edu/man/wngloss.7WN.html`, retrieved January 2011.

[3]Hypernym: The generic term used to designate a whole class of specific instances. Y is a hypernym of X if X is a (kind of) Y. WordNet Glossary of Terms, `http://wordnet.princeton.edu/man/wngloss.7WN.html`, retrieved January 2011.

[4]Entailment: A verb X entails Y if X cannot be done unless Y is, or has been, done. WordNet Glossary of Terms, `http://wordnet.princeton.edu/man/wngloss.7WN.html`, retrieved January 2011.

[5]The World Wide Web Consortium: `http://www.w3.org/`

[6]Web Ontology Language (OWL), Word Wide Web Consortium, `http://www.w3.org/TR/owl-ref/http://www.w3.org/TR/owl-ref/`

[7]WordNet RFD/OWL Files: `http://www.w3.org/2006/03/wn/wn20/`

represent the relationships between the two nodes or define an attribute value (object node) of the subject node.

McGuinness and van Harmelen [23] describe **OWL Web Ontology Language** as *designed for use by applications that need to process the content of information instead of just presenting information to humans*. OWL enables increased machine-readability and increased processing of web content, than is possible with RDF, as it provides the means to define relationships between instances in an ontology [23].

The data model of WordNet was modeled in RDFS[8] with some additional OWL statements providing more semantics [2]. The RDF/OWL data model of WordNet, in Figure 5.1, has three main classes conform the structure introduced earlier: Word, WordSense and Synset [2]. The WordSense class and Synset class each have subclasses for the lexical groups of WordNet: nouns, verbs, adjectives and adverbs. The Word class has a subclass for words that are collocations[9]. The AdjectiveWordSense and AdjectiveSynset have subclasses for satellite wordsenses and synsets, respectively.

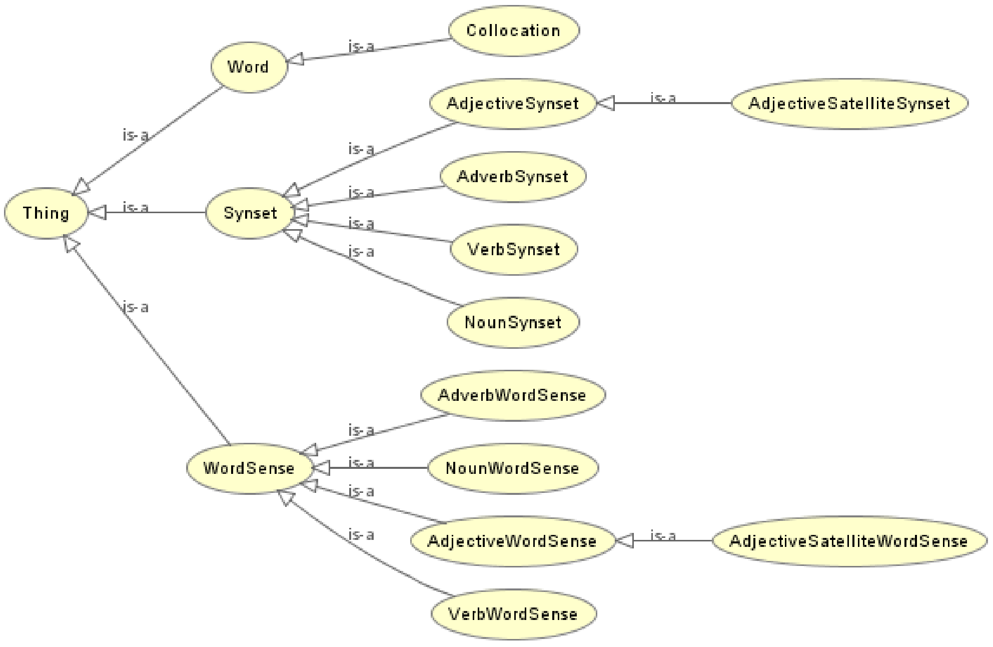

Figure 5.1: Schema of WordNet in Protege Browser

[8]RDF Schema, Word Wide Web Consortium, http://www.w3.org/TR/rdf-schema/

[9]Collocation: A collocation in WordNet is a string of two or more words, connected by spaces or hyphens. WordNet Glossary of Terms, http://wordnet.princeton.edu/man/wngloss.7WN.html, retrieved January 2011.

Each instance of a Word, WordSense and Synset has its own URI. Likewise each relation in WordNet is represented by an object property that has its own URI, as inherent to linked-data [3]. There are three types of object properties in the WordNet RDF/OWL data model [2]:

- Properties that connect instances of the main classes together.
- Properties that relate two WordSense instances with each other.
- Properties that relate two Synset instances with each other.

Given the scale of WordNet, the creators of the OWL/RDF representation of WordNet have published WordNet as a collection of datasets. In this manner, one does not need to load and query WordNet in its entirety, and therefore the memory and computation needed for querying WordNet is reduced [2]. There is a dataset for all words, wordsenses and properties that link them together. Further, there is a dataset representing all synsets. Finally, for each object property, be it one that links synset instances together or wordsense instances together, there is a dataset.

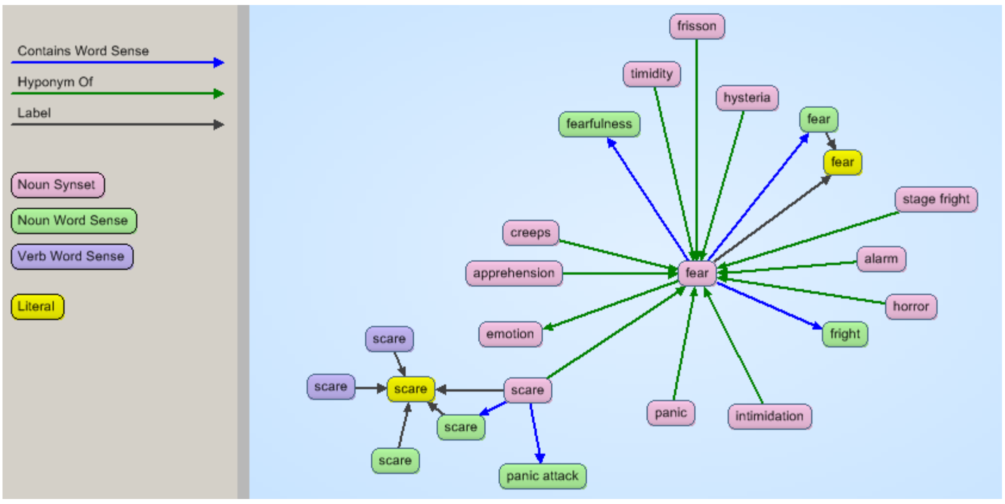

Figure 5.2: WordNet object and data properties (relations) between the literals "scare" and "fear"

Figure 5.2 depicts the relations between the literals "scare" and "fear", in WordNet. In Section 4.2 we discussed how Relco can relate the term "scare" to the concept label "fear" in STRPValues Vocabulary, using a lexical ontology. Here we can see the structure of WordNet and how it would enable Relco to find this relation. The literal "scare" is the literal for the NounSynset-scare and the NounWordSense-scare. The NounSynset-scare *containsWordSense* WordSense-scare and WordSense-panic_attack. There is a *hyponym* relation between the NounSynset-scare and NounSynset-fear. The literal for NounSynset-fear is "fear". We see that using the relations between synsets, we can traverse across different synsets with different wordsenses, and using the relation containsWordSense we can obtain the wordsense. Using data property rdfs:label we can obtain the literal for that wordsense.

WordNet utilized with Relco

When operating Relco we need to decide which object properties to use in the creation of SeRQL queries, so that we can effectively map tags to concepts from STRPValues and EmotionValues.

We could query the entire data set of WordNet and include all object properties in the SeRQL queries of Relco but that approach would be futile. The size of WordNet would increase the query time of Relco, perhaps without improving the final results. Further, not all object properties are applicable to the relationship between a tag and a concept from the STRPValues or EmotionValues vocabularies, so we do not need include all WordNet datasets in Relco's RDF store.

We need to make a trade off as to which object properties to include, and thereby what relationships we can identify between the tags and concepts in WordNet. As Figure 5.2 illustrated we can effectively traverse the RDF graph of WordNet through the object properties between Synsets. We therefore decided to narrow our selection of object properties to object properties that exist between Synsets. From the technical report of van Assem et al., we identified the following object properties that illustrate relations between synsets [2][10]:

containsWordSense: This property informs which synset is related to which wordsense. The domain and range is therefore synset and wordsense respectively.

hyponymOf: A synset A is a hyponym of B if it is a more specific type of B. For example, a dog is a hyponym of an animal, and British is a hyponym of European. This relationship occurs between verb-synsets and noun-synsets.

entails: The relations implies that synset A is an entailment of synset B. For example, inhale is an entailment of breathe. The object property entails relates verb-synsets to each other.

similarTo: This object property implies that synset A is similar in meaning to synset B. This relationship holds for adjective-synsets.

memberMeronymOf: This object property illustrates the member-of relationship between noun-synsets. An example of this relationship is that synset person is a member of synset people.

substanceMeronymOf: This relation implies that synset A is a substance that makes up synset B. For example oxtail is a substanceMeronymOf oxtail soup. This relationship occurs between noun-synsets.

partMeronymOf: This object property illustrates the part-of relationship between noun-synsets. For example, cell is a partMeronymOf organism.

classifiedByTopic: This specifies that a synset is a member of the class represented by another synset. For example synset cell is classifiedByTopic synset Biology. The domain and range of this property is a noun-synset.

[10]For further information about these object properties and other properties of the RDF/OWL representation of WordNet, we refer you to this document.

classifiedByUsage: This object properties relates two noun-synsets together, according to the relationship that one noun-synset is used in the activity represented by the second synsets. For example noun-synset blind_alley is used in noun-synset figure_of_speech.

classifiedByRegion: This object property illustrates that a noun-synset is related to another noun-synset according to physical regions. For example, outcaste is related to region India.

causes: This relates a verb-synset to be the cause of another verb-synset. For example, verb-synset anesthetize causes verb-synset to_sleep.

sameVerbGroupAs: This implies that two verb-synsets are similar in meaning.

attribute: This relation illustrates an attribute relation between noun and adjective-synset pairs. For example: adjective-synset commercial is an attribute of noun-synset mercantilism.

adjectivePertainsTo: This illustrates that an adjective-synset pertains to either another adjective-synset or a noun-synset.

adverbPertainsTo: This illustrates that an adverb-synset pertains to an adjective-synset.

Our second criteria was that we wanted to find synsets that shared a similar meaning, so that we could eventually map these to concepts of STRPValues and EmotionValues. The nature of these vocabularies, was that their instances were nouns. Further, tags were generally in the noun or verb form once they had been stemmed, using Porter Stemming. Therefore were were interested in relations between Noun-Synsets and Verb-Synsets. Therefore, the following list of object properties remained in our selection:

containsWordSense: We use this property to find the related wordsenses and synsets.

hyponymy: The object property hyponymOf related noun-synsets with other noun-synsets in a manner that followed our intuition for relating a tag to a concept from the vocabularies. This property is useful when provided with a very specific term, to find a more general term.

entails: When we incorporated this object property in Relco, we were disappointed that it did not seem to improve the mapping of tags onto concepts of STRPValues.

meronymy object properties: On closer inspection of the meronymy relations we found them too specific for our needs. Intuitively, they did not seem to hold the relations between noun-synsets that we were looking for.

classification object properties: The object property classifiedByTopic is an appropriate relation for relating a tag to concept of STRPValues and EmotionValues. We use this property as a SeRQL query with this property will allow us to broaden the semantic meaning of the term provided. The other classification object properties, classifiedByRegion and classifiedByUsage did not relate synsets to other synsets that were applicable in STRPValues and EmotionValues.

causes: As we had identified above, the need to map verbs to concepts, we incorporated this object property in our initial experimentation with Relco, however it did not appear to improve the mappings.

sameVerbGroupAs: Like the object property causes, we found this property did not improve the mappings. Our intuition is that verbs from the Porter Stemmer are still challenging to map onto concepts of STRPValues as they are nouns.

attribute: These relations, when we incorporated them in Relco, did not provide the results we had anticipated.

As a result of examining the contents of the datasets and the specifications of the object properties, and finally substituting combinations of these object properties in Relco, we selected WN20schema:containsWordSense, WN20schema:HyponymOf, WN20schema:classifiedByTopic to be used in the SeRQL queries of Relco in order to find relations between the input terms and concepts of WordNet. The literals, of which, we finally mapped onto concepts of STRPValues and EmotionValues. As a result we included the following WordNet datasets in the RDF Store used by Relco:

- WordNet-WordSensesAndWords.rdf,
- WordNet-Synset.rdf,
- WordNet-HyponymOf.rdf, and
- WordNet-ClassifiedBy.rdf.

With reference to the example of mapping the literal “scare” onto the concept literal “fear” of STRPValues, Table 5.1 records the intermediate results of Relco when using the above mentioned object properties and datasets. From these results we can also follow the certainty value of Relco for each concept after each process. The second column displays the results of the Match String process, the only concepts from WordNet to fall above the string distance threshold share the literal “scare”. After the Match Semantic Concept process, which searches for related concepts using the containsWordSense, hyponymOf and classifiedByTopic object-properties, the concepts wordsense-scare, NounSynset-fear and wordsense-panicattack are identified. The process Combine Concept By Related Tags identifies the literals “scare”, “panic attack”, “fear”, “fearfulness”, and “fright”. Their combined certainty values can be viewed in the fourth column. The fifth column displays the only literal to match the literal of concepts from STRPValues: “fear”.

Table 5.1: Intermediate results of Relco after each algorithm for the input term “scare”

Input Tags	Output of Match String onto WordNet	Output of Match Semantic Concept onto WordNet	Output of Combine Concept By Related Tags	Output of Match String onto STRPValues
scare	http://www.w3.org/2006/03/WN/wn20/instances/wordsense-scarenoun- 2 - Label: scare - c:1.0 http://www.w3.org/2006/03/wn/wn20/instances/synset-scare-noun-2 - Label: scare - c:1.0 http://www.w3.org/2006/03/wn/wn20/instances/wordsense-scarenoun- 1 - Label: scare - c:1.0 http://www.w3.org/2006/03/wn/wn20/instances/wordsense-scareverb- 1 - Label: scare - c:1.0 http://www.w3.org/2006/03/wn/wn20/instances/wordsense-scareverb- 2 - Label: scare - c:1.0	http://www.w3.org/2006/03/wn/wn20/instances/wordsense-scarenoun- 2 - Label: scare - c:1.0 http://www.w3.org/2006/03/wn/wn20/instances/wordsensepanicattack-noun-1 - Label: panic attack - c:1.0 http://www.w3.org/2006/03/wn/wn20/instances/synset-fear-noun-1 - Label: fear - c:1.0	scare - w:1.0 c:1.0 panic attack - w:1.0 c:1.0 fear - w:1.0 c:0.8999999985098839 fearfulness - w:1.0 c:0.8999999985098839 fright - w:1.0 c:0.8999999985098839	fear

5.2 Vocabularies

The final step in the Relco workflow is to identify a concept in either the STRPValues or EmotionValues Vocabularies, as was discussed in Section 4.2. We created a STRPValues Vocabulary using the expertise of STRP Festival organizers to identify aspects present in the art pieces to be exhibited, and we created an EmotionValues Vocabulary using Plutchik's Wheel of Emotions, discussed in Section 3.2.

STRPValues Vocabulary

In order for Relco to be able to query these datasets we had to create RDF representations of these vocabulary lists. We created an ontology in RDF/OWL using Protege[11]. We used the established SKOS[12] ontology to create our own ontology. SKOS is a data model for linking and sharing knowledge organization systems, and captures the common structure present in thesauri, classification schemes, and taxonomies [14]. For this reason, we adopted the data model of SKOS, creating a STRPValue as a sub-class of skos:Concept, as shown in Listing 5.1. Instances of STRPValues, such as Animation, have data properties rdfs:label and skos:prefLabel and this can be seen in Listing 5.2.

Listing 5.1: Exert of the RDF/OWL schema of STRPValues.owl: Creation of the STRPValue class

```
<owl:Ontology rdf:about="&ontologies;STRPValue.owl">
  <owl:versionIRI rdf:resource="&ontologies;STRPValue.owl"/>
  <owl:imports rdf:resource="http://www.w3.org/2004/02/skos/
      core"/>
</owl:Ontology>

...

<!-- http://www.semanticweb.org/ontologies/2010/9/STRPValue.
    owl#STRPValue -->
  <owl:Class rdf:about="&ontologies;STRPValue.owl#STRPValue">
    <rdfs:subClassOf rdf:resource="&skos;Concept"/>
  </owl:Class>

<!-- http://www.w3.org/2002/07/owl#Thing -->
    <owl:Class rdf:about="&owl;Thing"/>

<!-- http://www.w3.org/2004/02/skos/core#Concept -->
    <owl:Class rdf:about="&skos;Concept"/>
```

[11]The Protégé Ontology Editor and Knowledge Acquisition System, Standford University, http://protege.stanford.edu/

[12]Simple Knowledge Organization System (SKOS), World Wide Web Consortium, http://www.w3.org/2004/02/skos/core

Listing 5.2: Exert of the RDF/OWL schema of STRPValues.owl: Creation of the individual STRPValue:Animation

```
<!-- http://www.semanticweb.org/ontologies/2010/9/STRPValue.
    owl#Animation -->
  <owl:Thing rdf:about="&ontologies;STRPValue.owl#Animation">
    <rdf:type rdf:resource="&ontologies;STRPValue.owl#
        STRPValue"/>
    <rdf:type rdf:resource="&owl;NamedIndividual"/>
    <rdfs:label>animation</rdfs:label>
    <skos:prefLabel>animation</skos:prefLabel>
  </owl:Thing>
```

EmotionValues Vocabulary

In the case of creating a RDF representation of the EmotionValues vocabulary, we took a different approach and used an existing ontology for Emoticons: The SmileyOntology[13]. The SmileyOntology has evolved since emoticons have become a prevalent form of communicating emotions over the Internet and other media. Over time different sets of emoticons have come into existence. SmileyOntology is a data model for aligning different sets of emoticons with each other [24].

Importing the SmileyOntology allowed us to create an RDF representation of Emotions with the class SmileyOntology:Emotion. We also imported the SKOS ontology so that we could provide emotions labels and preferred labels. These imports are visible in Listing 5.3. Individuals of SmileyOntology:Emotion were created that had data properties rdfs:label and skos:prefLabel, for example the individual Fear, in Listing 5.4.

Listing 5.3: Exert from EmotionValues.owl: Creation of the EmotionValues Ontology

```
<owl:Ontology rdf:about="http://www.semanticweb.org/ontologies
    /2011/0/EmotionValues.owl">
  <owl:versionIRI rdf:resource="http://www.semanticweb.org/
      ontologies/2011/0/EmotionValues.owl"/>
  <owl:imports rdf:resource="http://www.smileyontology.com/ns"
      />
  <owl:imports rdf:resource="http://www.w3.org/2004/02/skos/
      core"/>
</owl:Ontology>
```

[13]SmileyOntology, http://www.smileyontology.com

Listing 5.4: Exert from EmotionValues.owl: Creation of the individual EmotionValues:Fear

```
<!-- http://www.semanticweb.org/ontologies/2011/0/
     EmotionValues.owl#Fear -->
<owl:NamedIndividual rdf:about="http://www.semanticweb.org/
     ontologies/2011/0/EmotionValues.owl#Fear">
  <rdf:type rdf:resource="&ns;Emotion"/>
  <rdf:type rdf:resource="&ontologies;EmotionValues.owl#
       EmotionValue"/>
  <rdfs:label>fear</rdfs:label>
  <skos:prefLabel>fear</skos:prefLabel>
</owl:NamedIndividual>
```

In this chapter we examined the lexical ontology WordNet and we also discussed how Relco would use these to map terms to concepts from STRPValues and EmotionValues vocabularies. In the next chapter, we will evaluate the performance of Relco in mapping tags to concepts of STRPValues and EmotionValues.

Chapter 6

Evaluation of Mapping Tags to Concepts

In Chapter 4 we introduced Relco, a tool for relating a string term to a concept in an ontology. Further, we also discussed the different string matching algorithms that can be initialized, the functionality of stemming a term before running Relco, and the possibility of carrying out a recursive semantic search for concepts in an ontology. The ontology and vocabularies we chose to use with Relco, in the context of the STRP Festival (Section 2.1), were introduced in Chapter 5: WordNet, STRPValues and EmotionValues.

In this chapter we evaluate the performance of Relco in relating tags to concepts in an ontology. We have established that, for the STRP Recommender System, it is imperative that we are able to relate visitor tags to concepts in STRPValues, we also need to be able to identify STRPValues for art pieces using tag generation algorithms such as TF and TF-IDF (Section 3.3). We evaluate the performance of relating user tags to concepts in Section 6.1, where we also evaluate the performance of a range of different Relco settings. Following this, we evaluate and compare the performance of Relco using generated tags from guidebook descriptions (Section 6.2). Finally, in Section 6.3 we evaluate the influence other vocabularies would have on the performance of Relco.

6.1 Evaluation of Relco Settings

Visitor models encompass the concepts that are derived from tags contributed by the visitor. It is therefore important to understand the performance of Relco in mapping tags, from free-tagging contributions, onto concepts of STRPValues vocabulary.

Relco is an elaborate tool with many settings available to the operator. We chose to focus on the performance of relating tags to concepts whilst experimenting with the following three settings:

- matching algorithms (Match String process): Levenshtein, Jaro Winkler and SoundEx (Section 4.3.1)
- stemming terms before Relco performs matching algorithms (Section 4.3.2)
- semantic search for concepts (Match Semantic Concept Process) in WordNet (Section 4.3.3)

All possible combinations of these settings are depicted in Table 6.1. There are 12 different combinations of settings and for each we have assigned a code. For example Levenshtein with Recursion and Stemming is encoded as *L_R_S*.

Table 6.1: Settings of Relco

	Levenshtein	Jaro Winkler	SoundEx
No Stemming or Recursion	L	J	X
Recursion	L_R	J_R	X_R
Stemming	L_S	J_S	X_S
Recursion and Stemming	L_R_S	J_R_S	X_R_S

Experiment

The aim of our experiment was to establish which combination of settings would perform the best.

In order to evaluate this we needed to have some way of measuring the correctness of Relco relating tags to concepts. Thus, we needed a dataset of user tags for a variety of art pieces (**tag_dataset**), and a dataset of the STRPValues for these art pieces (**concept_dataset**). We created the tag_dataset by collecting tags, from the public, on YouTube[1] and delicious[2] for resources of art pieces. In the case of the concept_database, we asked the organizers of STRP Festival to assign STRPValues to these art pieces. In total, we collected a dataset of 21 different pieces of art.

We evaluated this by relating the tags of the public (tag_dataset) to concepts from STRPValues and compared these results with the STRPValues assigned to the same art pieces (concept_dataset).

[1]YouTube http://www.youtube.com/

[2]delicious, Yahoo, http://www.delicious.com/

Results

We measured the performance of Relco in two ways: We measured the number of correct and incorrect matches to concepts of STRPValues. These results can be viewed in Table 6.2. Secondly, we measured the number of instances (21 sets of tags) that a combination of settings was the best and worst performing, and these results can be viewed in Table 6.3.

In order to avoid ambiguity in our results, it is important to define what characterizes an instance of best or worst cases:

Best Case: This is an instance where the number of matches for this combination is equal to the most number of matches for this art piece amongst all combinations.

Worst Case: This is an instance where the number of matches for this combination is equal to the lowest number of matches for this art piece amongst all combinations.

Table 6.2: Results of Matches and Mismatches

	L	L_R	L_S	L_R_S	J	J_R	J_S	J_R_S	X	X_R	X_S	X_R_S
Incorrect	26	50	16	55	32	33	29	19	56	49	57	46
Correct	29	36	17	31	29	29	29	27	36	28	31	32
Correct/Incorrect	1.12	0.72	**1.69**	0.56	0.91	0.88	1	**1.42**	0.64	0.57	0.54	0.7

Table 6.3: Results of Best and Worst Cases

	L	L_R	L_S	L_R_S	J	J_R	J_S	J_R_S	X	X_R	X_S	X_R_S
Worst Case	9	10	3	6	18	18	18	18	18	14	11	13
Best Case	14	8	12	12	21	21	16	17	16	14	16	18
Best/Worst	1.56	0.8	**4**	**2**	1.17	1.17	0.89	0.94	0.89	1.07	1.45	1.38

From the results we can see that the combinations that produce the most correct matches also produce the most incorrect matches. For example L_R, L_R_S and X. Similarly, we saw that combinations that had the most number of best cases also had the most number of worst cases. We therefore looked at the ratio of correct_matches:incorrect_matches and best_cases:worst_cases. From these results we saw that L_S, L_R_S and J_R_S scored the best, and L_S scored best, overall.

Conclusion

From these results we can conclude that stemming the input terms improves the performance of Relco, and that a recursive semantic search for concepts does not improve the performance of Relco. When we stem the input terms, the cost of creating new terms is less, therefore the string matching algorithm can create more words that are above the string distance threshold. For this reason, we believe the combinations using stemming may perform better than combinations without stemming.

The recursive semantic search increases the number of concepts that are found, and therefore this improves the chance of a correct concept being found but also increases the chances of incorrect matches occurring. For this reason we believe that combinations without recursion in the semantic concept search may perform better than combinations with recursion in the semantic concept search.

From these results we choose to use L_S, L_R_S and J_R_S combinations in our next experiment where we evaluate Relco's performance using generated tags.

6.2 Evaluation of Generated Tags

Tags provided by humans have a level of intuition and intelligence that could perhaps influence Relco to perform better or worse. We wanted to establish the possibility of automatically generating STRPValues for art pieces using the guidebook descriptions. This is important as these STRPValues are used in the creation of the installation model, and will be used when formulating recommendations.

Experiment

For this experiment we needed to generate a dataset of generated tags (**genTags_dataset**) from guidebook descriptions of art pieces. We did this using two different algorithms: TF and TF-IDF. We then compared the STRPValues generated from these tags with the STRPValues assigned to these art pieces by STRP organizers (concept_dataset). We evaluated the performance of Relco using a set of 22 guidebook descriptions.

Result

The results of this experiment can be viewed in Table 6.4 and Table 6.5, where genTags_dataset was generated using TF-IDF algorithm and TF algorithm, respectively.

We measured the performance of Relco by calculating the average number or correct matches per art piece (average intersection). We also calculated the average percent of correct matches per art piece, and the average percent of mismatches per art piece. We calculated the total number of correct matches and incorrect matches for the entire dataset (genTags_dataset). Finally we calculated the ratio correct_matches:incorrect_matches for the dataset.

We noted that TF-IDF tag generation results in more correct matches but also more incorrect matches to concepts of STRPValues, across the entire dataset. When we examine the average intersection for each art piece, we see that TF produces on average less number of correct matches. With regards to L_S, TF tag generation vastly reduces the number of correct matches and incorrect matches, compared to L_S with TF-IDF tag generation. The ratio correct_matches:incorrect_matches shows that L_S with TF tag generation performs the best.

Table 6.4: Results of TF-IDF generated tags

	L_S	L_R_S	J_R_S
Average Intersection	1.33	0.89	1.38
Average Matches (%) per art piece	25.32	36.43	33.43
Average Mismatches (%) per art piece	74.68	73.57	66.57
Correct	28	17	29
Incorrect	79	38	76
Correct/Incorrect	0.35	0.45	0.38

Table 6.5: Results of TF generated tags

	L_S	L_R_S	J_R_S
Average Intersection	0.85	0.9	1.19
Average Matches (%) per art piece	32.86	26.11	24.99
Average Mismatches (%) per art piece	67.14	72.89	75.01
Correct	18	19	25
Incorrect	20	44	72
Correct/Incorrect	**0,9**	0.43	0.35

Conclusion

TF-IDF algorithm is designed to generate tags that are unique to a particular document in a repository, and the TF algorithm will generate tags most frequent to a particular document [20]. Given this, we expected the TF algorithm to result in more correct matches than the TF-IDF algorithm, as it would generate terms that are most common and thus express the essence of the document. This was however the opposite, the TF algorithm resulted in less matches but also in less mismatches. TF may be more accurate in producing tags that correctly map to concepts of STRPValues, as it selects the most frequent words found in the guidebook description. This implies that very common, simple words, provided they are not common stopwords, would be selected. Whilst, in case of TF-IDF tag generation, the terms selected may be very obscure, as they would score high in the TF-IDF calculation, being unique to this document.

We can only assume that TF terms, being common in a document, are very clear in meaning to Relco, their variations are mapped to a fewer number of concepts in STRPValues. The variations of the TF-IDF terms however are ambiguous to Relco as they map to a larger number of concepts in STRPValues. An example of such a TF-IDF term is the tag “acts” from the guidebook description of Underworld[3]. Using L_S, Relco produces the concepts: play and performance. The STRPValues assigned to Underworld in concept_dataset include neither of these two concepts. The TF generated terms do not include the tag “acts”, and the concepts play and performance are not amongst the concepts mapped by Relco.

As we concluded in the previous section, Relco performs better when terms are stemmed before used in a string matching algorithm. Also, the use of recursion in the semantic concept search increases the likelihood of correct matches but also incorrect matches. It is therefore, feasible that L_S with TF tag generation performs the best in the context of mapping tags to concepts of an ontology.

6.3 Evaluation of STRPValues Vocabulary

STRPValues vocabulary was created by people who have knowledge about art but not necessarily knowledge or experience in creating vocabularies or classification taxonomies. We therefore needed to explore the influence the choice of vocabulary has on the performance of Relco. We ultimately wanted to explore whether we could improve the performance of Relco by tweaking the STRPValues vocabulary.

Through literature research we learned of the phenomenon: lexical gaps or semantic distance unevenness in WordNet [10]. This simply means that the semantic difference between a parent and a child node varies amongst nodes in WordNet. Through earlier experimentation with Relco and WordNet we encountered a few instances of pairs of WordNet words always being identified simultaneously by Relco. A very prominent example was word game and word play often mapped to from a single term, and thus the STRPValues STRPValue:game and STRPValue:Play were identified. When we examined WordNet, we see that the relation between these two terms is semantically closer together in WordNet than we would expect it to be.

Experiment

In this experiment we continued to use the genTags_dataset of TF algorithm generated tags, but created a second STRPValues vocabulary (**STRPValues2**). STRPValues2 is a vocabulary based on STRPValues. We looked at the rdfs:label of each concept in STRPValues2 and examined these terms in the Merriam-Webster[4] on-line dictionary and WordNet. In some instances we added extra rdfs:label tuples to the concept[5]. We added, for example, synonyms to the concept that had more semantic relations in WordNet than the original rdfs:label assigned to the concept in STRPValue. For example, for the STRPValue:Aesthetic we added a rdfs:label “appearance” to the instance. Another example is STRPValue:HighVoltage: there was no actual concept in WordNet with a similar label. In STRPValue we had already preempted this by adding rdfs:label “voltage”. In STRPValue2 we chose a less specific term that was also more common; rdfs:label “electricity”.

[3]Underworld: http://en.wikipedia.org/wiki/Underworld_(band)

[4]Merriam-Webster: Dictionary and Thesaurus. http://www.merriam-webster.com/

[5]The data model of STRPValues is defined such that STRPValue:STRPValue has at least one rdfs:label and only one skos:preflabel

Result

When we compare the performance of Relco using STRPValues2, in Table6.6, with the performance of Relco using STRPValues, in Table6.5, we notice that the number of matches only slightly improves but the number of mismatches also increases.

Table 6.6: Results of TF generated tags using STRPValues2 Vocabulary

	L_S	L_R_S	J_R_S
Average Intersection	0.86	0.95	1.33
Average Matches (%) per art piece	32.06	26.51	25.32
Average Mismatches (%) per art piece	67.94	73.49	74.68
Correct	18	20	28
Incorrect	24	48	79
Correct/Incorrect	**0.75**	0.42	0.35

Conclusion

The improvement of a vocabulary marginally improves the chance of a match. The strategy to add more labels to a concept in an ontology increases the chance of mismatches, as there are more labels that a term can be matched to semantically or syntactically. Therefore, we can conclude that, overall, adding more rdfs:label entities to STRPValues vocabulary reduces the performance of Relco. In this situation, a vocabulary such as STRPValues should be assigned only one term (rdfs:label) per concept.

Interestingly, the altered vocabulary negatively effects the best performing combination (L_S) the most, and barely effects the combinations that perform dramatically worse. We can conclude that this occurs because L_S does not use recursive semantic search, it collects less concepts using STRPValues, however the combinations that include recursive semantic search cover so many terms, that STRPValues2 vocabulary does not influence the number of concepts collected to such a degree.

Part III

Formulating Recommendations Using Concepts

Chapter 7

Recommendation Systems

In Part II we explored relating a tag to a concept in an ontology. In this part, we explore how we can use concepts to formulate recommendations. In this chapter, we present an introduction to recommender systems. In Section 7.2 we sketch different approaches to formulating recommendations. Following that, in Section 7.3, we discuss the different problems that occur with different recommendation approaches. Finally, in Section 7.4 we will present different hybrid approaches that aim to combat the problems that occur in traditional recommendation approaches.

7.1 Introduction to Recommender Systems

Recommender systems are defined as systems that formulate individualized recommendations as output [5]. A recommendation system is also defined as any system that guides the user to interesting or useful resources amongst a large number of resources. The difference between recommender systems and information retrieval systems or search engines is that recommender systems use an individualized search for resources that are interesting and useful to a particular user [5].

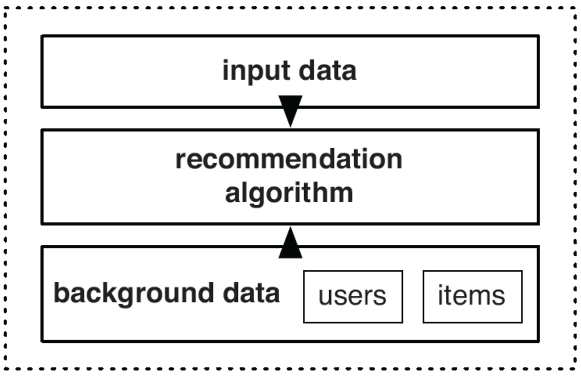

Figure 7.1: Generic Recommendation System [17]

Burke [5] describes a recommender system as consisting of the following three elements, as depicted in Figure 7.1: input data, recommendation algorithm and background data. Input data is the information the user must provide in order for the recommender to provide a recommendation, whilst background data is information the recommender has before the recommendation process commences. In this figure, the background data includes information about the users and items that can be recommended. Finally, the task of the recommendation algorithm is two combine these to sources; the input and background data; so as to formulate suggestions.

7.2 Recommendation Approaches

There are a range of different approaches to recommendation algorithms available. Adomavicius [1] and Burke [5] classified recommendation algorithms into three main categories: **Collaborative** recommendations, **Content-based** recommendations and **Knowledge-based** recommendations.

Collaborative Recommendation

Collaborative recommendations (CF) are formulated by recommending items that people with similar preferences have liked in the past [1]. The most common approach to this algorithm, is to aggregate the ratings of objects so as to recognize commonalities between users, based on their rating habits. Typically, a user profile is stored as a vector, with each object rating combination that was provided by the user [5]. A rating can be either binary (for example, like or dislike) or real values (such as a nominal scale of natural numbers from 1 to 5). The input data is thus a set of ratings for objects by a user, and the background data is a history of all ratings from all users, for all objects.

This approach is the most frequently implemented and developed, of all recommendation algorithm approaches [5].

Content-Based Recommendation

Content-based recommendations (CN) are based on the features of objects, which are available for recommendation, that meet the needs of the user [5]. A slightly different definition of content-based recommendations is provided by [1]: The user is recommended objects similar to objects the user has liked in the past. In their definitions of this approach, Burke [5] puts the emphasis of matching objects with user needs, whilst Adomavicius [1] puts the emphasis on measuring the similarity between objects. They, however, share the view that a recommendation is formulated by comparing the user profiles with object profiles.

Features of the objects is background knowledge, whilst the input data describes the needs of the user [5]. The features of objects and users, needs to be described using the same set of features, otherwise formulating a recommendation will be unsuccessful [1, 5].

In the past most content-based recommender systems have been built for textual resources, such as documents and websites. Their profiles consist of keywords from the text and can be ascertained using methods such as TF-IDF. User profiles are formed either by explicit means such as a questionnaire, or implicit means such as the history of the user's interaction with the system.

Knowledge-based (KB)

Knowledge-based recommendation (KB) uses a set of inference rules to match the user's needs to the features of an object. Although all recommendation systems use some kind of inference rule to formulate a recommendation [5], KB possess **functional knowledge** as background data; this is knowledge about how a particular item meets a particular need. The relationship between the need of a user and a possible relation, with an object, can be reasoned.

We do not discuss knowledge-based recommendation further as we are investigating the formulation of recommendations using concepts and ratings. We will, therefore, focus on issues concerning CF and CN recommendations.

7.3 Common Recommendation Approach Problems

Different approaches to recommendations suffer a variety of problems. We focus on the problems inherent to CF and CN approaches and we recognize the following problems [1, 5]:

New user Problem is the lack of information, about the user, available to the recommender when a new user is added to the system. This is especially prominent in a system where the rating history of a user (CF) or an implicitly-gathered user profile (CN), is used to formulate a recommendation. The system simply has no material to compare a user with other users or to compare a user with objects, in the system.

New Item Problem is similar to the new user problem. A new item has not yet been rated and so, in the case of CF, it will not be recommended to similar users.

Sparsity Problem occurs when there are still few ratings in the system. This will mean that few users would have rated the same items and therefore, in the case of CF, the similarity between users will be quite low. There is a critical mass of users and ratings necessary before the recommender can perform relatively accurately.

Overspecialization is a phenomenon prevalent to CN where recommendations of objects tend to be very similar to objects the user has been recommended in the past. Users are seldom provided with recommendations of objects that are not highly similar to previously recommended objects, but are still of interest to them. For example, a restaurant recommendation system that uses CN would not recommend a Greek restaurant to a user who has never been recommended a Greek restaurant before, even if this restaurant was the best restaurant in the city.

7.4 Hybrid Approaches

As each recommendation approach presents potential problems when formulating recommendations, a combination of CF and CN recommender algorithms, a **hybrid** is prescribed [1, 5]. The most common type of hybrid is CF combined with another kind of recommendation approach, to combat the new item and new user problems [5].

There are a number of ways of combining two recommendation approaches [1, 5]:

Weighted Hybrid is where recommendations are formulated separately and then combined. The advantage of this approach is that the capabilities of each recommendation approach is fully exploited and the recommendations are adjusted in the final stage in the hybrid process.

Switching Hybrid makes a choice as to which recommendation approach to use, depending on the current situation when the recommendation is needed. In this way, switching between a CN system and a CF would reduce the chance of a user being recommended objects that are too similar (overspecialization).

Mixed Hybrid presents results from both recommendation approaches. This is an appropriate approach when a number of recommendations needs to be provided in the same instance.

Cascade Hybrid is a staged process, where recommendations that were formulated by the first recommendation approach are fed into the second recommendation approach to be refined. The advantage of this approach is that a rough selection can be formulated using a less costly approach, and a more costly approach can be used on a selected set.

Feature Combination Hybrid uses features of two different approaches in a single algorithm. For example, features of the CN approach are incorporated into the CF recommendation algorithm.

Feature Combination of Collaborative and Content-based Recommendation Algorithms

Feature combination of CF and CN algorithms is very popular. When we incorporate aspects of CN into a CF algorithm, we can avoid the new item and sparsity problems that are present in CF approaches [1, 5]. This is because the profile of the user is compared with profiles of objects, so a new object could be matched even if it has not yet been rated by a user.

The disadvantage of this approach is that a system of rating still remains necessary, not only for the recommender itself to become more accurate, but also so that the user profiles and the object profiles can evolve with the preferences of users [5].

The feature combination of CF and CN approaches is advantageous to the requirements of the STRP Recommendation System. In this instance, we have STRPValues profiling both the visitor and art pieces, and visitors provide ratings for the art pieces that they view. In the following chapter we will discuss the design of the STRP Recommendation Algorithm.

Chapter 8

STRP Recommendation Approach

The objective of Chapter 7 was to introduce recommendation algorithms and discuss different approaches available. We learned that collaborative recommendations (CF) and content-based recommendations (CN) each have their own strength and weaknesses. A hybrid algorithm of the two is effective in combatting some of these problems. In this chapter we will present a recommender algorithm for STRP Recommender System. In Section 8.1 we present the data model of STRP, which stores the visitor profiles and artwork profiles. The possible algorithms of the STRP Recommendation System are detailed in Section 8.2, amongst which we present a possible hybrid algorithm for formulating recommendations of artwork at the STRP Festival.

8.1 STRP Ontology

In the data model of the STRP Recommendation System, information is stored about the installations (artworks), visitors, artists and festivals. In Figure 8.1 we only depict the most important information, especially that relevant to the formulation of a recommendation. The following information is stored in this data model:

- A **Visitor** visits different festivals and views different **Installations**.
- Likewise, a **Festival** exhibits installations and is visited by visitors.
- An installation is created by an **Artist** and is exhibited at a festival.

In previous discussions about the STRP Recommender System, we established that visitors would provide tags and ratings for artworks that they viewed. We have also discussed that these tags would be related to concepts from the STRPValues or EmotionValues Vocabulary. A record of these concepts and their corresponding ratings must also be stored in the data model, as it is needed for recommendation formulation. Thus an important relation in the STRP Recommendation data model is the **ConceptValue**. Every instance of a visitor providing tags and a rating, for an installation results in ConceptValues being stored for each related concept: The concept, either from the STRPValues or the EmotionValues Vocabulary, and the rating are stored as a ConceptValue record, as depicted in Figure 8.1.

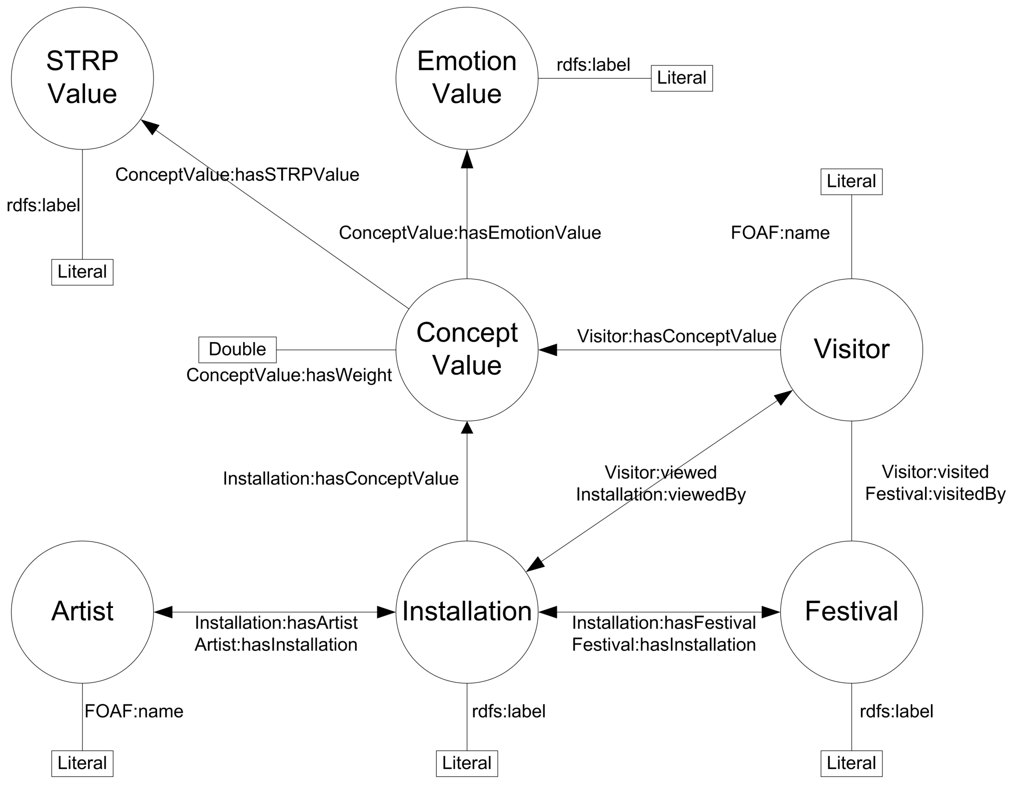

Figure 8.1: STRP Recommender System data model

8.2 Possible STRP Recommendation Algorithms

Each recommendation system shares the following characteristics: input data, background data and a recommendation algorithm, as depicted in Figure 7.1 [5]. Likewise, there is a common notation that can be used for describing a recommendation algorithm [1].

Let

- C be a set of all visitors,
- S be a set of all installations that can be recommended, and
- u be the utility function that measures the usefulness of installation s to visitor c.
  - Therefore $u : C \times S \rightarrow R$, where R is an ordered set of recommendations.

In other words: $\forall c \in C, s'_c \arg\max u(c,s)$

What we compare to decide which installation s is an appropriate recommendation for visitor c, defines what type of recommendation algorithm is utilized. Since we have access to the tags and ratings provided by a user for installations, we have a number of algorithms which we can implement.

Content-Based Recommendation

An option is to compare the visitor profile with the profiles of installations, and suggest installations that have similar profiles to that of the visitor. To do so, we can implement the CN algorithm. When we implement the CN algorithm, the recommender still suffers from the new user problem and overspecialization. The implementation of CN is achieved as follows [1]:

- The $u(c, s_j)$ of the installation s for visitor c is estimated on the utilities of $u(c, s_i)$ assigned by visitor c to installations $s_i \in S$ that are similar to installation s. We do this by comparing the profiles of visitors with the profiles of installations.
- The profile of visitor c, $ContentBasedProfile(c)$, is defined as the vector of weights $(w_{c1}, \ldots w_{ck})$, where each weight w_{ck} represents the aggregate of the sum of ratings where concept k was assigned to installations by visitor c.
- The installation profile of s, $Content(s) = (w_{1j}, \ldots w_{sk})$, has weight w_{sk} of concept k for installation s. The weight w_{sk} is the aggregate of the sum of the ratings assigned when this concept k was assigned to installation s.
- We calculate the similarity between $ContentBasedProfile(c)$ and each installation profile s_j; $u(c, s_j)$, by calculating the cosine of the angle between the two vectors using the cosine rule found in Equation 8.2.1. Therefore, $u(c, s_j) = \cos(\overrightarrow{w_c}, \overrightarrow{w_s})$.

$$\cos(\overrightarrow{w_c}, \overrightarrow{w_s}) = \frac{\overrightarrow{w_c} \cdot \overrightarrow{w_s}}{\|w_c\|_2 \times \|w_s\|_2} = \frac{\sum_{i=1}^{K} w_{c,i} w_{s,i}}{\sqrt{\sum_{i=1}^{K} w_{c,i}^2}\sqrt{\sum_{i=1}^{K} w_{s,i}^2}} \tag{8.2.1}$$

- Once we have an ordered list of $u(c, s_j)$ for all installations s, we can select the n most similar installations $(Top - n)$.

We recommend installations that appear in the $Top - n$ but have not yet been rated by visitor c.

Collaborative Recommendation

Since we have ratings for installations, we can implement a CF algorithm, where we recommend installations that similar visitors have also liked. A CF algorithm as the STRP Recommendation Algorithm would mean that we would only take ratings into account, and the tags a users provides would have no influence on the formulation of the recommendation. The recommender would suffer the new item, new user and sparsity problem.

In CF, the utility of installation s for visitor c is estimated based on the utilities of $u(c\prime, s)$ assigned to installation s by those visitors $c\prime \in \hat{C}$ who are *similar* to visitor c: [1].

- A visitor profile is a vector of the ratings assigned by the visitor c for each installation s, $RatingBasedProfile(c) = (w_{c1}, \ldots w_{cs})$. If a visitor has yet to rate a particular installation the weight is 0. We find the n most similar visitors $c\prime$

by using the cosine rule in Equation 8.2.1, where we calculate the similarity measurements between the visitor c and all the other visitors.

- Using this smaller set of similar visitors $c\prime$, we predict the ratings visitor c will give to all installations s that visitor c has yet to rate. We calculate the predicted rating, $r_{c,s}$, of installation s for visitor c using Equation 8.2.2.

$$r_{c,s} = k \sum_{c\prime \in \hat{C}} sim(c, c\prime) \times r_{c\prime,s}, \tag{8.2.2}$$

where normalizing factor $k = 1/\sum_{c\prime \in \hat{C}} |sim(c, c\prime)|$ and $sim(c, c\prime) = sim(x, y)$ (Equation 8.2.3).

- In Equation 8.2.2, $r_{c\prime,s}$ is the actual rating provided by visitors $c\prime$ for installation s. If this is not yet rated the rating would be 0.

$$sim(x,y) = \cos(\overrightarrow{x}, \overrightarrow{y}) = \frac{\overrightarrow{x} \cdot \overrightarrow{y}}{\|x\|_2 \times \|y\|_2} = \frac{\sum_{s \in S_{xy}} r_{x,s}, r_{y,s}}{\sqrt{\sum_{i=1}^{K} r_{x,s}^2} \sqrt{\sum_{i=1}^{K} r_{y,s}^2}} \tag{8.2.3}$$

Once we have predicted ratings $r_{c,s}$ for all installations s that visitor c has yet to rate, we recommend the n installations s that have the highest predicted ratings.

Hybrid Collaborative Content-based Recommendations

A third possibility is to implement a hybrid algorithm using both CF and CN algorithms. We could implement CN, using not the rating history of visitor c to find similar visitors, but using the weight values for concepts assigned by visitor c for installations, which are stored in vector $ContentBasedProfile(c)$.

- First we calculate the n most similar visitors $c\prime$, using the cosine rule in Equation 8.2.1.
- We then want to predict the rating of installations s visitor c has yet to rate using the ratings visitors $c\prime$ have assigned these installations. The predicted rating $r_{c,s}$ is the aggregate of the ratings all visitors c assigned to that particular installation (Equation 8.2.4).

$$r_{c,s} = aggr_{c\prime \in \hat{C}} r_{c\prime,s} \tag{8.2.4}$$

Installations s with the n highest predicted ratings $r_{c,s}$ are recommended for visitor c.

In summary, an implementation of CF will have new user, new item and overspecialization hazards, whereas a CN implementation will possess the new item and sparsity problems. Should we combine the two algorithms into a Feature Combination Hybrid, then the new item, sparsity and overspecialization problems will be reduced. We evaluate the actual performance of these three algorithm approaches in Chapter 9.

Chapter 9

Evaluation of STRP Recommendation

We presented an outline of the recommendation algorithms available, in Chapter 7. Using this theory, in Chapter 8, we explored three different recommendation algorithms that can be implemented in the STRP Recommender System. These were the Collaborative Recommendation (CF) using ratings of users, Content-based Recommendations (CN) using visitor and installation (art piece) profiles, and the Hybrid Feature Combination of Collaborative and Content-based Recommendation (CF-CN).

In order to evaluate a STRP Recommender System we set up an experiment; the details of this experiment are found in Section 9.1. We present our evaluation of the results of the three STRP Recommendation Algorithms in Section 9.2.

STRP algorithms, CN and CF-CN, will use not only the ratings of installations but also the concepts provided by visitors. These concepts are a result of mapping the tags, provided by visitors when they rate installations, to concepts of the STRPValues and EmotionValues vocabularies. This mapping is achieved by exploring semantic relations in WordNet between the tags provided and the concepts in WordNet, and eventually mapping the literals of the WordNet concepts onto the literals of concepts in STRPValues and EmotionValues. We investigated the value of using this semantic extension instead of tags in CN and CF-CN, and we present our findings in Section 9.3.

9.1 The STRP Recommendation Experiment

We aspired to simulating a festival situation where visitors would provide ratings and tags for installations which they viewed. From this simulation we wanted to, using different recommendation algorithms, formulate recommendations. We planned to evaluate the performance of each recommendation approach by comparing the predicted rating with that of the actual rating.

The dataset

We collected a dataset (**STRP_dataset**) of ratings and concepts through means of an on-line questionnaire. The questionnaire had YouTube[1] and Vimeo[2] footage of installations, embedded on its pages. We required participants to view these videos and provide a rating and between 3 and 5 tags, for each installation. The rating was a nominal scale of natural numbers from 1 to 5, and participants were requested to provide English tags.

In total, 27 participants provided tags and ratings for 7 different installations. Using Relco, we mapped the tags onto concepts of STRPValues and EmotionValues vocabularies. This resulted in a dataset (STRP_dataset) of 189 ratings, 648 tags and 237 concepts.

Simulations

We wanted to simulate formulating recommendations using STRP_dataset. We randomly removed ratings, tags and concepts from visitors (participants of the on-line questionnaire). For each visitor in the set, we proceeded to formulate recommendations using different recommendation algorithms: CF, CN and CF-CN.

As all of these algorithms, select either the $Top-n$ number of similar visitors (CF and CF-CN) or installations (CN) during the process of recommendation formulation, we wanted to investigate the effect the size of n would have on the accuracy of the recommender. We therefore ran all 3 simulations using both $Top-3$ and $Top-5$. We were limited to such a small variation in $Top-n$ as there are only 7 installations in our dataset. It would, therefore, not be feasible to take, for example, the $Top-13$ installations, in the case of CN.

Finally, as we wanted to investigate the value of semantic extension in recommender systems, we ran each algorithm using the concepts we had mapped from the tags, and then again using the tags themselves. As CF does not take into account the concepts or tags contributed by visitors, we only carried out CF for $Top-3$ and $Top-5$, once.

In total, we ran 10 different simulations.

Evaluation Measurements

During each simulation we formulated a list of recommendations for each visitor (participant of the on-line questionnaire). A recommendation is, in essence, a predicted rating for a number of installations. The higher the predicted rating for an installation, the more likely the visitor will appreciate this installation.

From each simulation, we calculated the following for each visitor using their list of recommendations:

- We calculated the difference ($diff$) between the predicted rating and the actual rating for each recommendation.

[1] YouTube: http://www.YouTube.com/
[2] Vimeo: http://vimeo.com/

- We calculated the *mean* of the values of diff ($mean_c$) for each visitor c. The $mean_c$ will indicate how accurate the recommendation is. Therefore, the smaller the value for $mean_c$, the more accurate the recommendation algorithm.
- We calculated the sample standard deviation std_c of all $diff$ measurements per visitor c, using Equation 9.1.1. We calculated the sample standard deviation so that we can gain insight into the variance (spread) of the $diff$ measurements for each visitor c. We want to know the variance as it will indicate whether the recommendation algorithm is consistent in its level of accuracy.

  $$std_c = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{9.1.1}$$

  where $\bar{x}$ is the sample mean ($mean_c$)

We then proceeded to calculate the following values for each simulation:

- The average of all $mean_c$ ($mean_{avg}$) in the simulation
- The average of all std_c (std_{avg}) in the simulation
- The total number recommendation formulated per simulation (tot_r).
- The average number of recommendations formulated per visitor, for that specific simulation (avg_r).
- The total number of visitors for whom at least one recommendation was formulated (*coverage*).

9.2 Results of STRP Recommendation Algorithms

We wanted to investigate the performance of the three algorithms (CF, CN and CF-CN) which we detailed in the previous chapter. We ran each simulation using the ratings and concepts from the STRP_dataset for both the Top-3 and Top-5. From this dataset, ratings and their corresponding concepts had been removed from visitors, at random. From the 189 rating and concepts pairs, we removed 74. In total, we removed ratings and corresponding concepts from 23 of the 27 visitors (on-line participants). Each simulation formulated predicted ratings for the "missing" rating and concept pairs.

Table 9.1: Results of the Top-3 CF, CN and CF-CN Simulation using concepts

	CF	CN	CF-CN
Average mean of the difference between predicted ratings and actual ratings of visitor *c* ($mean_{avg}$)	1.55	1.13	1.62
The average of the standard deviation amongst visitors *C* (std_{avg})	0.86	0.67	0.90
The total number of recommendations formulated (tot_r)	74	18	74
The average number of recommendations formulated per visitor (avg_r)	3.22	1.64	3.22
The number of visitors for whom at least one recommendation was formulated out of 23 visitors (*coverage*)	23	11	23

Table 9.2: Results of the Top-5 CF, CN and CF-CN Simulation using concepts

	CF	CN	CF-CN
Average mean of the difference between predicted ratings and actual ratings of visitor *c* ($mean_{avg}$)	1.50	1.00	2.02
The average of the standard deviation amongst visitors *C* (std_{avg})	0.82	0.71	0.96
The total number of recommendations formulated (tot_r)	74	47	74
The average number of recommendations formulated per visitor (avg_r)	3.22	2.32	3.22
The number of visitors for whom at least one recommendation was formulated out of 23 visitors (*coverage*)	23	22	23

Tables 9.1 and 9.2 display the results of these 6 simulations. From these results we see that CN performs the most accurately in both Top-3 and Top-5, and it is also the most consistent in its level of accuracy.

From the results of the CF and CN simulations we conclude that we can better predict ratings with CN than with CF, as the $mean_{avg}$ and std_{avg} is smaller in the case of CN. For this, we have the following explanation: In Section 8.2 we introduced the vectors $ContentBasedProfile(c)$ and $Content(s)$ for visitors and installations in CN, and the $RatingBasedProfile(c)$ for visitors in CF. In the case of CF we profile visitors using only the ratings they provided for other installations. Whilst in the case of CN, we profile the preferences of visitors using the concepts combined with ratings that they have provided in the past, and we profile installations using concepts combined with the ratings contributed by other visitors when they rated the installation. Therefore, in CF we calculate similarity between visitors based purely on the ratings they assigned installations in the past. Where as in CN similarity between a visitor and installation is based not on the rating but on the concept in proportion to the rating assigned by a visitor. Therefore in CN similarity is based on more data than for CF, and for this reason CN performs better than CF.

Regarding the influence of Top-3 and Top-5, we see that, in the case of CF and CN, the larger the $Top-n$, the more accurate the recommender is. However, in the case of CF the consistency of accuracy improves, whilst in the case of CN, the consistency level deteriorates. This is because there are only 7 installations but 27 visitors in our dataset. Given that a rating is a nominal scale of natural numbers from 1 to 5, visitor profiles will be more similar to each other as the range of ratings is so narrow. However, as there are only 7 installations, their profiles could be more distinct from each other. For this reason, when selecting a larger $Top-n$, the difference between the most similar visitor and the least similar visitor of the $Top-n$ selection will be less than the difference between the most similar and the least similar installation of the $Top-n$ selection. Therefore the variation (std_{avg}) in CF decreases whilst it increases in CN.

We also notice that an increase in $Top-n$ vastly increases the number of recommendations by CN. This is expected as we select the 5 most similar installation profiles to consider recommending, instead of 3. As there are only 7 installations, when you select 3 instead of 5 most similar installations you have less chance of there being an installation that has not been rated by the visitor for whom the recommendation is being formulated. Therefore there are less number of recommendations by CN when using Top-3 than when using Top-5.

In the case of CF and CF-CN, the increase in $Top-n$ has no effect on the number of recommendations. This is because as there are 27 visitors, whether 3 or 5 similar visitors are selected there is nearly equal chance that amongst the similar visitors there is a visitor who has rated an installation that the visitor for whom the recommendation is being formulated has not.

Disappointingly, the CF-CN performs the least in accuracy and consistency, and the larger the $Top-n$, the less accurate and consistent the recommendations. An explanation for this would be that comparing visitors with each other using a vector of concepts in proportion to the ratings assigned with that concept is not effective. This is because the vector allows for no distinction to be made as to which installation what rating or concept was assigned. Therefore visitor vectors could be identical even when visitors assigned different concepts and ratings to the same installation. This decreases the accuracy of the similarity measurements between visitors.

This problem could be mitigated by extending the $ContentBasedProfile(c)$ to store a rating for each combination of concepts and installations. In this way, we would be able to compare visitors on the difference in ratings and concepts, they assigned to the same installation.

We can conclude that using a CN algorithm will increase the accuracy of the recommendations formulated, as apposed to a CF algorithm using the rating history of visitors. This indicates that we are better able to find the similarity between a visitor and an installation using concepts and ratings, than we are able to find similar visitors in an instance of CF using only rating history.

9.3 Evaluation of Semantic Extension

As a large part of this thesis explores relating tag to concepts in an ontology, it is important to understand to what extent a semantic extension in a recommender algorithm would have an effect on the accuracy and possibly the consistency of the recommendations. We investigated this by running simulations of CN and CF-CN using tags (instead of concepts) and ratings from the STRP_dataset, for both Top-3 and Top-5 situations.

Table 9.3: Results of the Top-3 CN and CF-CN Simulation using Tags

	CN_T	CF-CN_T
Average mean of the difference between predicted ratings and actual ratings of visitor *c* ($mean_{avg}$)	0.96	1.72
The average of the standard deviation amongst visitors *C* (std_{avg})	0.72	0.96
The total number of recommendations formulated (tot_r)	30	74
The average number of recommendations formulated per visitor (avg_r)	1.5	3.22
The number of visitors for whom at least one recommendation was formulated out of 23 visitors (*coverage*)	10	23

Table 9.4: Results of the Top-5 CN and CF-CN Simulation using Tags

	CN_T	CF-CN_T
Average mean of the difference between predicted ratings and actual ratings of visitor *c* ($mean_{avg}$)	0.96	2.07
The average of the standard deviation amongst visitors *C* (std_{avg})	0.55	1.07
The total number of recommendations formulated (tot_r)	51	74
The average number of recommendations formulated per visitor (avg_r)	2.32	3.22
The number of visitors for whom at least one recommendation was formulated out of 23 visitors (*coverage*)	22	23

The results of CN and CF-CN using tags can be found in Tables 9.3 and 9.4, using Top-3 and Top-5 settings, respectively. We compare these results to the results of CN and CF-CN using concepts for Top-3 and Top-5 that are found in Tables 9.1 and 9.2.

We observe that CN with tags is marginally more accurate than CN with concepts. We could conclude that tags perhaps profile visitors and installations more accurately, and therefore CN and CF-CN perform better with tags. We must however realize the origins of the STRP_dataset that was used in these simulations. We obtained these tags through an on-line questionnaire where we asked participants to provide ratings and tags for installations, but also asked their opinion for the STRPValues that had been assigned to the installation by the STRP organizers. In hindsight we realize that we may have seeded the tags of the participants by asking their opinion of the STRPValues onto which we mapped their tags. It is therefore plausible that this dataset is too perfect, and therefore the recommendation algorithms with tags perform too well, as the dataset has been influenced.

The disadvantage to using tags is that a larger number of weights in the visitor profile needs to be compared when calculating similarity amongst visitors and installations. For example in STRP_dataset, there were 26 distinct concepts as apposed to the 394 distinct tags. When we calculated the cosine of a visitor's concept vector with the concept vector of an installation, we needed to calculate the distance between each corresponding element in both vectors. In the case of CF and CF-CN using concepts the vectors are 26 elements in size, where as with regards to CF and CF-CN using tags, the vectors are 394 elements in size.

Although we cannot deny that using concepts as apposed to tags in CN and CF-CN algorithms may deteriorate the accuracy and consistency of the recommendations, the calculation cost of recommendation formulation is reduced. With respect to semantic extension in CN and CF-CN algorithms we can conclude, that in the context of the STRP recommender we can accurately profile a visitor and an installation using a limited number of concepts, and the resulting recommendations from these profiles are comparable to the accuracy and performance of CN using tags.

Part IV

Conclusion

Chapter 10

Conclusions and Future Work

In this thesis we have, in Part I, presented the context and design of the STRP Recommender System. In Part II we evaluated mapping tags to ontological concepts in order to establish the meaning of these tags, and in Part III we investigated the incorporation of the meaning of tags in the recommendation formulation process. This part is our concluding part: In this chapter we will summarize our contributions in this thesis, in Section 10.1. In Section 10.2 we present our conclusions to our research question, and in Section 10.3 we reflect on the validity of our results for research in recommender systems, in general. Finally, in Section 10.4 we discuss future work regarding this topic.

10.1 Contributions

During the course of this thesis we explored what was necessary to formulate recommendations for visitors of STRP Festival. From this we identified a number of processes necessary for achieving this. Amongst which, we identified the need to map a tag to a concept from an ontology, for which we eventually used Relco. We also involved the organizers of STRP Festival in the creation of the STRPValues vocabulary used to categorize art pieces.

During our investigation of Relco we explored its different settings, and we implemented recursion in the semantic concept search function of Relco. We also implemented the possibility of stemming the tags before Relco executes the string matching function on them. Following that, we set-up experiments to test the different settings of Relco. We also investigated how we could automatically establish the STRPValues for the installation models of new art pieces.

Finally, we set about investigating how effective the recommendation formulation process was if we incorporated this semantic technology. In order to do this, we wanted to create a dataset of tags and ratings. We gathered this through means of an on-line questionnaire which we compiled and distributed. Using this data we simulated formulating recommendations using 10 different scenarios, and we measured the accuracy and consistency of these recommendations.

10.2 Conclusions

From this thesis we conclude that a recommender system, in the context of STRP Festival, will provide the organizers the means to learn about the preferences and experiences of visitors at the festival. We also conclude that Relco equips us with the means to map tags to concepts from the STRPValues Vocabulary. It is most effective when we stem the tags before we perform Levenshtein string matching. We also learned the importance of creating a vocabulary where each concept has only one string literal as this reduces the chance of mismatches occurring.

Through our investigation of incorporating the STRPValues of tags in the recommendation formulation process, we conclude that content-based recommendations are most accurate. We also conclude that semantic extension, as we have implemented it, does not improve the accuracy and consistency of recommendations to any great extent. We do, however, conclude that this semantic extension accurately profiles visitors and installations, whilst using a finite number of concepts. This will reduce the complexity of recommendation formulation calculations as the number of ratings and tags increases.

10.3 Reflection

In the previous section, we conclude that we can accurately profile visitors and art pieces with a finite number of concepts. Whilst our results have displayed this to be true in the context of STRP Festival; we cannot claim that content-based recommendation approaches using semantic extension will be as effective in other contexts. Firstly, we believe the simplicity of the STRPValues and EmotionValues allows for relatively accurate mappings, using Relco. A simple vocabulary may not be applicable in other contexts. Secondly, our conclusions are based on results of recommendation formulations using a relatively small dataset of users, art pieces, tags and ratings. We cannot exclude that on a larger scale the accuracy and consistency of the recommendations may decrease.

Ultimately, from this thesis, we can conclude that there is a scenario where a content-based recommendation approach with a semantic extension, using Relco, is feasible. This thesis brings to light the fragility of mapping tags to concepts in an ontology, and identifies factors, such as the choice of STRPValues vocabulary, that influence this and ultimately affect the accuracy of formulating recommendations.

10.4 Future work

As a result of our literature research, our evaluation of Relco and our investigation of different recommendation algorithms for the STRP Recommender System, we have established a number of extensions that could possibly improve the accuracy of recommendations:

Multidimensional Recommendations

In the past, recommendation systems have considered two aspects in their recommendation formulation: the users and the objects that can be recommended. Adomavicius [1] suggests recommendation systems should also take into account the context of the situation when the recommendation is required, for example, the time of year, place etc.

With respect to the STRP Recommender System, if we added the context we could enhance the experience of the visitor and elicit more contributions from the visitor. For example: if the STRP Recommender System took into account the congestion of people at certain art pieces or the starting time of live performances, visitors could be guided to art pieces that were less busy and they would not have to wait in queues or stand in crowds to view artworks, or they could be guided to performances they may perhaps miss as there are limited number of performances.

Multicriteria Ratings

The creation of a multicriteria of ratings [1], will elaborate the rating of a visitor, but it will increase our understanding of what a visitor appreciates in an art piece. The critical part of a multicriteria rating is the creation of this multicriteria; if it is too specific or ambiguous it will be ineffective in understanding visitors, and for visitor too taxing to provide.

Non-intrusiveness

A common issue with recommendation systems is that they require a high level of involvement and interaction from the user [1]. Investigation needs to be done into the incorporation of measuring sentimentality of tags instead of also requiring ratings from visitors. SentiWordNet[1] is a tool available for measuring the polarity of concepts from WordNet. The current obstacle is that the version of WordNet published in RDF is different to the version of WordNet on which SentiWordNet is based.

[1]SentiWordNet: http://sentiwordnet.isti.cnr.it/

Bibliography

[1] Gediminas Adomavicius and T. Tuzhilin. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, 17(6):6, June 2005.

[2] Mark van Assem, A. Gangemi, and G. Schreiber. RDF/OWL Representation of WordNet. Draft, World Wide Web Consortium (W3C), June 2006.

[3] Tim Berners-Lee. Linked-Data. available via <http://www.w3.org/DesignIssues/LinkedData.html>, accessed 28 October 2009, July 2006.

[4] G.A.R.M. Boschouwers. CHI Explorer building a community. Master's thesis, Eindhoven University of Technology, Department of Mathematics and Computer Science, 2008.

[5] Robin Burke. Hybrid Recommender Systems Survey and Experiments. *User Modeling and User-Adapted Interaction*, 12:331–370, 2002.

[6] Sam Chapman. String Similarity Metrics for Information Integration. url: http://staffwww.dcs.shef.ac.uk/people/S.Chapman/stringmetrics.html, 2006. Natural Processing Group, Department of Computer Science, University of Sheffield.

[7] William W. Cohen, P. Ravikumar, and A. Fienberg. A Comparison of String Distance Metrics for Name-Matching Tasks. In *Proceedings of IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03)*, 2003.

[8] Jonathon Collins. A Broader Definition of RFID. url: http://www.rfidjournal.com/article/view/4819, April 2009.

[9] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, May 1998.

[10] Christiane Fellbaum. *Theory and Applications of Ontology; Computer Applications*, chapter 10, pages 231–243. Springer, 2010.

[11] George W. Furnas, T. K. Landauer, L. M. Gomez, and S.T. Dumais. The vocabulary problem in human-system communication. *Communications of the ACM*, 11:964–971, 1987.

[12] Scott A. Golder and B. A. Huberman. Usage patterns of collaborative tagging systems. *Journal of Information Systems*, 32(2):198–208, 2006.

[13] RDFCore Working Group. RDF/XML Syntax Specification. Revised, World Wide Web Consortium (W3C), February 2004.

[14] Semantic Web Deployment Working Group. SKOS Simple Knowledge Organization System. Reference, Word Wide Web Consortium (W3C), August 2009.

[15] Tom Gruber. Collective Knowlegde systems: Where the Social Web meets the Semantic Web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6:4–13, 2007.

[16] Tom Gruber. Ontology of Folksonomy: A mash-up of Apples and Oranges. *International Journal on Semantic Web and Information Systems*, 3(1):1–11, 2007.

[17] Benjamin Heitmann and C. Hayes. Using Linked Data to Build Open, Collaborative Recommender Systems. In *Proceedings of AAAI Spring Symposium "Linked Data Meets Artificial Intelligence"*, 2010.

[18] G.J.A.M. Ketelaars. From Tag to Concept. Master's thesis, Eindhoven University of Technology, Department of Mathematics and Computer Science, February 2008.

[19] Hak Lae Kim, A. Passant, J. G. Breslin, S. Scerri, and S. Decker. Review and Alignment of Tag Ontologies for Semantically-Linked Data in Collaborative Tagging Spaces. In *Proceedings of the 2008 IEEE International Conference on Semantic Computing (ICSC 2008)*, 2008.

[20] Sheila Kinsella, A. Budura, G. Skobeltsyn, S. Michel, J.G. Breslin, and K. Aberer. From Web 1.0 to Web 2.0 and Back - How did your Grandma Use to Tag? In *Proceeding of the 10th ACM workshop on Web information and data management (WIDM 2008)*, pages 79–86, October 2008.

[21] Qingfeng Li and S. C.-Y. Lu. Collaborative Tagging Applications and Approaches. *IEEE*, 15(3):14–21, July-September 2008.

[22] Julie Beth Lovins. Development of a Stemming Algorithm. *Mechanical Translation and Computation Linguistics*, 11(1&2):22–31, March and June 1968.

[23] Deborah L. McGuinness and F. van Harmelen. OWL Web Ontology Language Overview. Recommendation, Word Wide Web Consortium (W3c), February 2004. http://www.w3.org/TR/2004/REC-owl-features-20040210/.

[24] Nikola Milikic, F. Radulovic, N. Kolundzija, and A. Svitlica. Smiley Ontology Specification. Working draft, University of Belgrade, November 2009.

[25] George A. Miller. Wordnet: A Lexical Database for English. *Communications of the ACM*, 38(11):39–41, November 1995.

[26] George A. Miller and Christiane Fellbaum. WordNet then and now. *Language Resources & Evaluation*, 41(2):209–214, October 2010.

[27] Bruno Oliveira, P. Calado, and H. S. Pinto. Automatic Tag Suggestion Based on Resource Contents. In *Proceedings of the 16th international conference on Knowledge Engineering (EKAW 2008)*, pages 255–264, 2008.

[28] Alexandre Passant and P. Laublet. Meaning Of A Tag: A Collaborative Approach to Bridge the Gap Between Tagging and Linked Data. In *Proceedings of the WWW 2008 Workshop Linked Data on the Web (LDOW2008)*, 2008.

[29] Martin F. Porter. An Algorithm for Suffix Stripping. *Program*, 14(3):130–137, July 1980.

[30] Mark Tribe and R. Jana. *New Media Art*. Taschen, 2006.

[31] Peter Willett. The Porter stemming algorithm: then and now. *Program:electronic library and information systems*, 40(3):219–233, 2006.