Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València

Exploiting context modeling for frequent pattern interpretation

Master final work

Máster Universitario en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

Author: Diego Aineto García

Tutor: Eva Onaindía de la Rivaherrera
Laura Sebastiá Tarín

Course 2016-2017

Resumen

La minería de patrones frecuentes consiste en desarrollar algoritmos de minería de datos con el objetivo de descubrir patrones interesantes, inesperados y útiles en las bases de datos. Normalmente, los patrones interesantes están relacionados con patrones que ocurren frecuentemente en base a información no semántica, como soporte y confianza. Identificar información semántica nos permitirá construir el modelo de contexto de los patrones y extraer indicadores de contexto informativos, transacciones representativas y otros patrones semánticamente similares.

Nosotros proponemos utilizar conocimiento adicional proveniente de los datos estructurados del dominio, como ontologías y bases de datos, para enriquecer el modelo de contexto. Concretamente, nuestra propuesta se basa en la extracción de grafos que representan las relaciones semánticas de los patrones y definir el contexto de los patrones mediante la aplicación de algoritmos de inferencia para grafos. Esta novedosa estrategia produce modelos de contexto enriquecidos que ayudan a interpretar el significado de los patrones y a explorarlos a diferentes niveles de abstracción.

Este método proveerá a los patrones con una anotación semántica que mejorará su entendimiento y usabilidad. Además, esta propuesta descubre relaciones ocultas dentro de los patrones y detecta similaridad entre los patrones a un mayor nivel de abstracción, lo que permite obtener un conjunto de patrones frecuentes más general y compacto.

Palabras clave: minería de datos, patrones frecuentes, modelado de contexto, anotación semántica, grafo

Abstract

Frequent pattern mining consists of developing data mining algorithms to discover interesting, unexpected and useful patterns in databases. Typically, interesting patterns are associated to patterns that occur frequently on the basis of non-semantic information such as support and confidence. Identifying semantic information will allow us to construct the context model of the patterns and extract informative context indicators, representative transactions of the pattern as well as semantically similar frequent patterns.

We propose to exploit additional knowledge from domain-dependent structured data, such as databases and ontologies, to enrich the context model. Specifically, our proposal is based on extracting the graphs that represent the semantic relationships of the patterns and define the context of the patterns through the application of graph inference algorithms. This novel scheme yields enriched context models that help interpret the meanings of the pattern and further explore them at different levels of abstraction.

The results will provide patterns with a semantic annotation that will ultimately enhance the understanding and usability of these patterns. Furthermore, this approach uncovers hidden relationships within the patterns and showcases the similarity of different patterns at a higher level of abstraction allowing for a more general yet compact set of patterns.

Key words: data mining, frequent patterns, context modeling, semantic annotation, graph

Contents

List of Figures

List of Tables

CHAPTER 1
Introduction

In this first chapter, we describe the significance of the problem this Master thesis deals with as well as the objectives it aims to accomplish. We also detail the structure of this document.

1.1 Motivation

During the past years, storing data electronically has become extremely cheap and now many organizations own huge datasets which need to be analyzed to benefit from them. The sheer volume of data makes the analysis using traditional methods inviable and has given rise to a new research field known as *data mining*. Data mining is a set of techniques designed to automatically analyze data and extract interesting patterns, trends or other useful information.

One of the most prominent subfields of data mining is frequent pattern mining. This subfield comprises a set of techniques able to uncover hidden patterns from the data. Patterns are interesting because they present correlations between items and these are very useful decision making in companies and organizations. For example, if one knows two products are often bought together, one could offer a sale in one of the products and increase the price of the other product, a common marketing strategy.

One of the most well-known examples of application of such techniques is the one by Wal-Mart stores (a popular American multinational retailing corporation). Some years ago, Wal-Mart decided to extensively mine the sales data they had collected over the years, and many correlations were found. Some discoveries were trivial like, for example, people buying gin also buy tonic; but some other revealed interesting patterns. One of the most unpredictable patterns found was that the people who buy diapers also buy beer. This pattern is interesting because it is unexpected to think of these two products as related. However, later analysis revealed that the reason for this pattern was that people who buy diapers have sons, so they cannot go out drinking and buy the beer to drink at home.

Research on this field until now has focused mostly on solving the issue of uncovering frequent patterns hidden in the data. This has been done by developing more efficient algorithms able to process data faster or by extending the applicability of these algorithms to other data structures, such as sequences and graphs. There has been, however, little work done on explaining why a pattern is frequent and most related research has focused on developing metrics to measure the interest of a pattern. This presents a serious limitation, as knowing the reasons of why a pattern happens can be key to exploit them for decision-making.

In this work, we try to go beyond simply assigning some statistical measure to a pattern. We do this by providing patterns with a *semantic annotation*, that is, contextual information that helps explain why a pattern happens and under what circumstances. In order to accomplish this, we propose a method that incorporates domain knowledge to semantically annotate frequent patterns, thus providing patterns with an enriched description that represents both their meaning and usage.

1.2 Objective

The main objective of this Master thesis is to develop a method for the semantic annotation of frequent patterns using domain knowledge.

To achieve this objective, the following specific objectives will need to be fulfilled:

- Define the task of semantic annotation of frequents patterns and its main subtasks, as well as related concepts.
- Design the context of a pattern and develop a context modeling method that allows us to incorporate knowledge into the context of a pattern.
- Develop a method to semantically annotate frequent patterns using their context.
- Use our method to build the semantic annotation of the frequent patterns extracted from different datasets.
- Evaluate the results obtained in the previous point.

1.3 Structure

In this first chapter we have showcased the relevance of the issue this work addresses and the objectives we aim to fulfill.

The second chapter will provide a discussion on related work published on the topic of frequent pattern mining. As this is an extensive field, we present this section as an introduction to the topic and limit ourselves to present the most popular algorithms. We will also use this chapter to present prior work done concretely on the semantic annotation of frequent patterns.

Chapter 3 will introduce the problem by presenting an illustrative example of the task. This chapter will also contain the formal definitions of the task, the subtasks it is divided into, and other related concepts.

The fourth chapter will be dedicated to present our approach, that is, we will explain how to perform each one of the subtasks defined in the previous chapter. We will complement the explanations with examples of the application of our method to the test datasets.

In the fifth chapter we will evaluate the semantic annotations obtained for the frequent patterns extracted from the test datasets. This will be done by analyzing each one of the features of the semantic annotation separately.

The last chapter, chapter 6, will present the conclusions of this Master thesis. We will also comment on the limitations of the model and propose future improvements to the method.

CHAPTER 2
Related work

In this section we introduce the problem of frequent pattern mining. In order to do this, we explain in Section 2.1 the related concepts and the three main subproblems, going over the main techniques developed for each one. Section 2.2 will be devoted to discuss previous approaches to the task of semantic annotation of frequent patterns.

2.1 Frequent pattern mining

Pattern mining is a subfield of data mining which aims to develop algorithms capable of finding interesting patterns in large datasets that are interpretable. Being able to interpret these patterns means being able to understand important bits hidden in huge databases and use them for decision-making. Depending on the type of data, different types of patterns can be defined, but, generally speaking the three major types are itemsets, subsequences and substructures. When the frequency of a pattern in a dataset surpasses a user-specified threshold it is referred to as a *frequent pattern.*

There is abundant work published about this topic since its beginnings in the 1990s. Some recent surveys summarize the current state of the art of frequent pattern mining [14, 11, 16].

2.1.1. Frequent itemset mining

The frequent itemset problem is often known as the "market-basket" problem as it defines a many-to-many relationship between items and transactions (or baskets) an tries to discover those sets of items that can often be found together. This problem was first introduced in [1] which analyzed customer buying habits by finding associations between different items.

Given $I = \{i_1, i_2, ..., i_n\}$, a set that contains all of the items in a database D, and a k-itemset α, which consists of k items from I, α is said to be frequent if α can be found in no less than a number of s transactions of D, where s is an user-specified threshold known as *minimum support.*

Considering that computing the possible k-itemsets in a database with n distinct items generates $\binom{n}{k}$ combinations, developing scalable algorithms for mining frequent itemsets in a large database is a challenging task. [2] proposed the Apriori algorithm utilizing an interesting property they named the Apriori property. This property says that a k-itemset is frequent only if all its sub-itemsets are frequent, meaning that for a 2-itemset to be frequent both its elements must be frequent 1-itemsets. The Apriori property gave place to a wide family of algorithms that uses a bottom-up approach in two steps. In the first step a candidate list of k-itemsets is built from frequent itemsets of size $k-1$. In the second one the database is scanned, computing the frequency of each candidate k-itemset and removing those that do not meet the minimum support. This process is iteratively executed until no more frequent k-itemsets are found.

A different approach was presented later in [15]. This approach introduced a new frequent pattern tree (FP-tree) structure, an extended prefix tree structure for storing information about frequent patterns. Using this structure they developed a new mining algorithm known as FP-growth which does not make use of candidate generation. FP-growth scans the database once and uses the frequency of the items to build a FP-tree that is recursively mined in a divide-and-conquer way. FP-growth addresses some of the issues with Apriori-based methods, such as repeated database scans and the costly generation of huge candidate sets. The FP-growth algorithm is able to extract the complete set of frequent patterns and is about an order of magnitude faster than Apriori-based methods.

There is one last family of mining methods that use a different database representation. Instead of the traditional horizontal format, where a set of items is associated to a transaction identifier (TID), a vertical format is used. The vertical representation associates to each item the set of transactions in which it can be found (TID set). The most well-known algorithm that uses this representation is Equivalence CLASS Transformation (Eclat) [24]. Eclat uses an incremental approach building the TID sets for $(k+1)$-itemsets by intersecting the TID sets of frequent k-itemsets. This methods takes advantage of the Apriori property and removes the need to scan the database repeatedly, thus improving the performance.

The concept of itemset patterns naturally leads to a different representation in the form of if-then rules named association rules. Association rules follow a $I \rightarrow j$ structure, where I is an itemset and j is an item. The intrepretation of this rules is that if I happens it is likely that j will also happen. A confidence measure for each rule can be computed using the formula $\frac{sup(I \cup j)}{sup(I)}$. The confidence of a rule can be seen as the posterior probability of item j given I ($P(j|I)$) and therefore association rules are very interesting for business decision-making.

2.1.2. Sequential pattern mining

In some domains, the ordering of events or elements is important and frequent itemset mining techniques, which ignore this ordering, may fail to find important patterns in the data or present patterns which are not useful due to this omission. The task of sequential pattern mining was first proposed by R. Agrawal and R. Srikant, the fathers of frequent pattern mining in 1996, to address the issue of uncovering interesting subsequences in a sequence database.

A sequence $\alpha = \langle a_1, a_2, ..., a_k \rangle$ is an ordered list, where a_i is an itemset comprised of items having the same timestamp. A sequence α contains a subsequence $\beta = \langle b_1, b_2, ..., b_k \rangle$ ($\beta \sqsubseteq \alpha$) if there exist integers $1 \leq i_1 < i_2 < ... < i_n \leq m$ such that $a_1 \subseteq b_{i_1}, a_2 \subseteq b_{i_2}, ...,$ and $a_m \subseteq b_{i_m}$. The support of a sequence α in a sequence database $D = \{s_1, s_2, ..., s_p\}$ is the number of sequences containing α.

The algorithms for sequential pattern mining share many similarities with the frequent itemset mining ones, as they were designed as extensions built upon them. As such, three major families of algorithms can be found. AprioriAll [3] and its revision, GSP [21], are inspired by the Apriori algorithm and work in a similar way by generating candidates for subsequences of length k using subsequences of length $k-1$.

The Spade algorithm [25], an extension of the Eclat algorithm, utilizes a vertical representation of the data that indicates the itemsets and the sequences in which item i can be found. This information exists for every distinct item in the database and is called IDList. This representation has two important advantages. The first one is that the support of any given pattern can be computed directly from its IDList. The second one is that candidate generation can be done by joining the IDLists of two subsequences, thus avoiding scanning the original database.

The last family of algorithms is based on the pattern-growth approach used in the FP-growth algorithm and its most representative algorithm is PrefixSpan [19]. This algorithm works in a divide-and-conquer way by recursively appending items to sequential patterns to build larger ones.

All of the aforementioned sequential pattern mining algorithm use two basic operations to generate $(k+1)$-sequences from k-sequences known as s-extension and i-extension. An s-extension consist in adding an item as a new itemset as the last itemset of the sequence. For example, $\langle\{a\},\{c\}\rangle$ is an s-extension of $\langle\{a\}\rangle$ with item c. On the other hand, an i-extension consists in adding an item to the last itemset of a sequence so, for example, $\langle\{a,c\}\rangle$ is an i-extension of sequence $\langle\{a\}\rangle$.

Similarly to association rules for frequent pattern mining, rules can also be defined for sequential patterns in the form $X \rightarrow Y$. These rules are named sequential rules and can be interpreted as if some items X appear in a sequence, it will be followed by items Y with a given confidence. Note that differently to the association rules, sequential rules imply a total order relationship between the items, meaning that for Y to happen, X must have happened before at some point in a sequence of events. For example, the rule $\{a\} \rightarrow \{b,c\}$ means that if a customer buys item a, he will later buy items b and c in any particular order.

2.1.3. Structured pattern mining

Itemsets and sequences are not always enough to model the data relationships in some domains. For these cases, more complex data structures are needed, like graphs, trees and lattices. The task of finding patterns among these is called structured pattern mining. The most representative problem in structured pattern mining is frequent subgraph mining, which purpose is finding all frequent subgraphs in a given data set.

Although more complex, the algorithms follow a similar approach to those previously seen in this paper. In a first step, candidate subgraphs are generated by expanding previously found frequent subgraphs. After that, the frequency of the candidates is computed. Counting the occurrences of a subgraph in a data set is particularly difficult since graph isomorphism must be taken into account and much research has been done to efficiently detect such isomorphisms [20, 22, 8].

Frequent subgraph mining techniques can be divided into Apriori-based approaches and pattern growth-based approaches. The Apriori-based algorithms use a breadth-first search strategy to explore the search space by levels, meaning that before considering subgraphs of size k, all subgraphs of size $k-1$ must be previously computed. On the other hand, pattern growth algorithms use a depth-first search approach to recursively find all supergraphs of a given subgraph.

2.1.4. Applications

Frequent pattern mining techniques have been applied to many fields. In this section, some of the most successful applications for each of the three main subproblems are discussed here.

Starting with frequent itemset mining, the most well-known example of application is the analysis of market baskets. In this case, the items are the different products being sold by a particular retailer and a transaction can be seen as the set of products bought together by a user. A frequent pattern here means that there is a strong connection between two or more products and offers very interesting information for marketing decision-making. One famous example of such patterns is {beer, diaper}, two items that apparently have nothing in common but have been proven to be tightly related. Although the market-basket problem is the most popular and the starting point in pattern mining, this is not the only possible application. Some other examples are:

- Plagiarism: Considering documents as items and sentences as transactions, plagiarism can be detected by finding pairs of documents which are frequent. If this happens it means that both documents have some sentences in common which usually mean that one of them has been plagiarized.
- Biomarkers: This is a very interesting application in which two types of items are used: diseases and biomarkers. A transaction in this example is the medical data about a particular patient. If a pattern consisting of one disease and one or more biomarkers is found, the biomarkers can be considered as symptoms of the disease. This methodology has been used to develop tests to detect diseases in patients [4].

Sequential pattern mining techniques also have many real-life applications since sequences are usually found in many domains. Some examples of applications of these techniques are:

- Click-stream analysis: A click-stream is the path a user follows when browsing a web page. The analysis of this path is very interesting from the point of view of e-commerce since it allows to determine the effectiveness of a site as a channel to the clients. In [10] a particular form of sequential rule is proposed to analyze click-streams.
- Bioinformatics: String mining is a particular case of sequential pattern mining characterized by limited alphabets and very long sequences. DNA sequences and protein sequences can be seen as an example of string mining [23]. Sequential pattern mining algorithms are able to identify structural units inside biological sequences which provide information about their properties.

Given the computational difficulties of frequent subgraph mining, the use of these techniques is not as widespread as the previous ones. Nonetheless, it has been very successfully applied in the field of biological science and medicine to discover relationships between the structure of chemical molecules and biological functions. In [9], frequent subgraph mining algorithms are used to predict the toxicity of unknown molecules.

2.1.5. Open research topics

Frequent pattern mining is a very popular topic for both academics and business companies. As such, it has been extensively researched to the point where it can be used as a solution to several problems. However, there are still many critical issues that need further work. Following, we expose some open research topics:

- Developing scalable and efficient algorithms: pattern mining is very computationally expensive and, although there are more than one hundred algorithms designed, there is still room for improvement. One possible way to achieve this is through the development of parallel algorithms able to operate on distributed file systems, which are commonly used on big data environments. A second approach to this problem is to sacrifice completeness of the algorithms for a compact set of frequent patterns that still holds the relevant information.

- Understanding and interpretation of patterns: In order to be able to use frequent patterns for decision-making and take advantage of them, the information they provide must be understood. This is not always a simple task since algorithms can output huge sets of frequent patterns and the relationship between correlated items is not easy to understand. It is interesting then, to develop mechanisms that go a step beyond pattern mining by providing deeper information about the patterns like, for example, meaning of the pattern, context in which the pattern is found and synonym patterns.
- Applications: Finding new uses for frequent pattern mining methods is an interesting topic. Since most domains can be modeled by either itemsets, sequences or graphs, there are still many opportunities for utilizing these techniques for new applications.

2.2 Semantic annotation of frequent patterns

As previously mentioned, most research in frequent pattern mining has focused its attention on developing efficient algorithms to discover various kinds of frequent patterns, but little attention has been paid to the next step: interpreting the discovered patterns. The problem of semantic annotation of frequent patterns is proposed in [17] to address this issue.

The goal of the semantic annotation of frequent patterns is to annotate frequent patterns with in-depth, concise and structured information that presents the meaning of a pattern. Given that this is a novel problem, the work in [17] proposes a general approach to generate the semantic annotation of frequent patterns as a way to define the meaning of a pattern.

The method proposed in [17] utilizes only elements present in the transactional database such as items, transactions and other patterns to model the context of a pattern. Concretely, they use micro-clustering to select a subset of those elements, which they call *context indicators*, and represent the context of a pattern as a vector measuring the relevance of each context indicator with respect to the given pattern.

This method has wide applicability since it does not need other input than that required by the frequent pattern mining process itself, but it is limited to the information found within the transactional database. This is a serious limitation, as transactions are comprised of identifiers or names of items, and only statistical information can be extracted from these elements. As such, the approach we present in this work draws upon their proposal and introduces a semantic annotation method able to incorporate knowledge from external sources to enrich the context of a frequent pattern.

CHAPTER 3

Problem formulation

In this chapter, we present the formalization of the problem we are aiming to solve. We will first introduce an illustrative example that will be used throughout the chapter to support the explanation of the different concepts and notions. The example is presented in section 3.1, and section 3.2 is devoted to give a formal definition of the concepts related to the semantic annotation of frequent patterns.

3.1 Illustrative example

We introduce an illustrative example for a better understanding and interpretation of a frequent pattern and its semantic annotation. In this example we will present a simplified version of the semantic annotation process to help the reader understand the basic concepts of this task and, on the following sections, we will elaborate on this process with the method we propose.

The annotation of a pattern should present the semantics of a pattern in a compact and structured way. That is, we can interpret the semantic annotation of a pattern as the contents of a dictionary entry.

Term: "enterprise"

Definitions:

1. A project or undertaking, especially a bold or complex one.
2. A business or company

Example sentences:

1. A joint enterprise between French and Japanese companies
2. A state-owned enterprise

Synonyms:

Company, venture, campaign, plan, etc

Figure 3.1: Example of a dictionary entry

Figure 3.1 shows a typical dictionary entry that represents the pattern 'enterprise'. The entry is structured in three parts. First, a group of definitions suggesting the different meanings of the term are given. Second, a set of example sentences showcasing the usage of the term. Last, some synonyms or semantically similar terms are shown, which share similar definitions.

Let us now consider the transactional database of Figure 3.2. This is a small database with only three transactions. Each transaction corresponds to the movies seen by a particular user so, for example, the interpretation of t_1 would be that user 1 has seen the movies *Terminator*, *Alien* and *Memento*. The frequent patterns that can be mined from this database with a minimum support $\sigma = 2$ are four, indicating that at least two users have seen the movies in the corresponding pattern: 1) {Terminator}, 2) {Matrix}, 3) {Alien}, and 4) {Terminator, Alien}. Of course, given that the database we are working with is a transactional database, the patterns obtained are itemsets formed by one or more items.

Transactional database
t_1 = {Terminator, Alien, Memento}
t_2 = {Matrix, Terminator, Alien, Fight Club}
t_3 = {Matrix, 12 Monkeys}
Frequent patterns ($\sigma = 2$)
p_1 = {Terminator}
p_2 = {Matrix}
p_3 = {Alien}
p_4 = {Terminator, Alien}

Figure 3.2: Illustrative example: transactional database and frequent patterns extracted

Our goal is to extract semantic information about the frequent patterns found in Figure 3.2 and annotate them in an structured way, similarly to the dictionary entry shown in Figure 3.1. The annotation will uncover hidden information about the pattern and provide the users with a semantic description that will help them better interpret the pattern.

The next step of the semantic annotation process is to locate a source from which to extract semantic information for our patterns. This cannot be done directly, but luckily enough, semantic information about the items comprising a pattern can be found in numerous sources like ontologies, knowledge bases and databases. For example, a retailer will not only have information about their sales or transactions, but also about their products, such as category and price; even if that is not the case, the time-stamp of the sale itself and the users who bought a particular product can be used as semantic information source. In this work, we will refer to the sources containing semantic information about the items as ***knowledge data sources***.

For our illustrative example, let us also consider the following knowledge data source. This is an ad-hoc data source built with data extracted from IMDb (Internet Movie Database), an online database of information related to films and television programs [1]. The table shows some extra information about the movies of our transactional database, particularly their genres.

Movie	Genres
Terminator	Action, Sci-fi
Alien	Horror, Sci-fi
12 Monkeys	Adventure, Drama, Mystery
Memento	Mystery, Thriller
Fight Club	Drama
Matrix	Action, Sci-fi

Table 3.1: Knowledge data source for the transactional database in Figure 3.2

Using the information from the knowledge data source in Table 3.1, we can build the context of the four frequent patterns we previously extracted (see Figure 3.2). Each

[1] http://www.imdb.com/

different genre can be considered a ***context unit***, and the genres associated to the movies of a particular frequent pattern the ***context indicators*** of such frequent pattern. Using this methodology we can model the context of the four patterns as illustrated in Figure 3.3.

Context units
Action, Sci-fi, Horror, Adventure, Drama, Mystery, Thriller
Context indicators
p_1: {Action, Sci-fi}
p_2: {Action, Sci-fi}
p_3: {Horror, Sci-fi}
p_4: {Horror, Action, Sci-fi}

Figure 3.3: Context modeling of the four frequent patterns in Figure 3.2

The context of a pattern can be used for different purposes:

1) Identify the context indicators that better represent the meaning of a pattern, which can be interpreted as the definition of the pattern.

2) Find the most representative transactions showcasing the usage of the pattern.

3) Measure the similarity between patterns and detect semantically similar patterns.

All of these features can be used to annotate frequent patterns, building something analogous to a dictionary entry as shown in Figure 3.4.

Pattern: {Terminator}
Context indicators:
Action, Sci-fi
Representative transactions:
t_1 = {Terminator, Alien, Memento}
t_2 = {Matrix, Terminator, Fight Club}
Similar patterns:
p_2 = {Matrix}

Figure 3.4: Semantic annotation of a frequent pattern

In Figure 3.4, the frequent itemset {Terminator} is annotated with its strongest context indicators, Action and Sci-fi, which for this trivial example are the whole set of context indicators. The semantic annotation also features the most representative transactions, t_1 and t_2; and a semantically similar pattern, {Matrix}, which can be considered a synonym since it shares the same context indicators as {Terminator}. The semantic annotation of a frequent pattern provides a dictionary-like description of a frequent pattern that can help understand why a pattern is frequent and under which circumstances it occurs. It also helps identify patterns with a similar meaning that can be considered redundant and, thus, can be used to compact potentially large lists of frequent patterns.

3.2 Semantic concepts

In this section, we introduce the formal definitions of the main concepts of the semantic annotation of frequent patterns. The definitions presented here are adapted from the ones proposed in [17] to consider the particularities of our proposal, that is, the usage of knowledge data sources to enrich the context of a pattern.

Let $I = \{i_1, i_2, ..., i_n\}$ be the set that contains all of the items of a transactional database $D = \{t_1, t_2, ..., t_n\}$, where each transaction t_i is an itemset consisting of items from I. Let U be the set of all concepts from a knowledge data source K so that $I \subseteq U$.

Definition 1. (Context indicator of a pattern). Given a frequent pattern α, a context indicator of α is any concept $u_i \in U$ which holds a direct or indirect relationship with an item of α.

With this definition, a context indicator may be any concept from K which holds some semantic information about the items found in the frequent patterns retrieved from D.

Generally, semantic relationships among concepts are represented in graphs, so knowledge bases and ontologies, which have this underlying representation, are the most likely knowledge data sources to be used. Nonetheless, relational databases can also be represented as graphs as we show in Figure 3.5. In this figure we represent the relational database of Table 3.1 as a graph, using nodes for both movies and genres, and edges to relate them. By doing so, this method is general enough to be applicable to most scenarios.

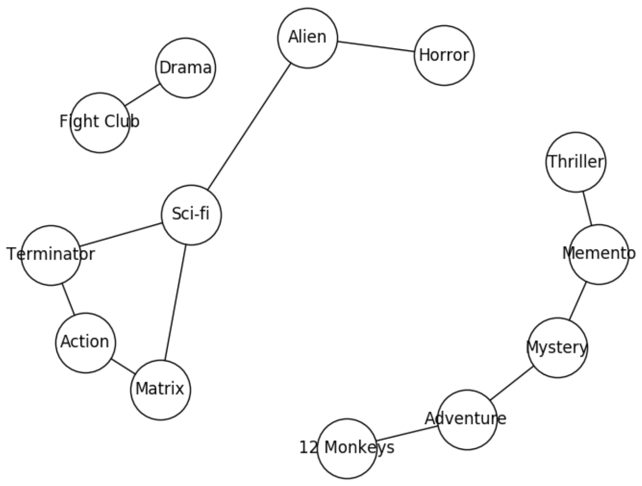

Figure 3.5: Graph representation of the relational database in Table 3.1

Definition 2. (Context of a pattern). Let α be a frequent pattern and U_α the set of context indicators of α. The context of α is a tuple $< G_\alpha, v_\alpha >$ where G_α is a graph whose nodes are concepts in U_α and the edges depict the relationships among those concepts, and v_α is a vectorial representation derived from the graph G_α. We will refer to G_α as the ***context graph*** and to v_α as the ***context vector*** of pattern α, respectively.

The context graph and context vector each convey different information about the semantics of a pattern and is better suited for different purposes: graphs are powerful tools to describe relationships among concepts, so we will use the context graph to extract the semantics of a pattern; the context vector, on the other hand, measures the relevance of the concepts in K for a given pattern and will be used to efficiently compute similarity between patterns. Keeping both representations allows us to leverage the strengths of each data structure during the semantic annotation process.

Let us consider again the illustrative example introduced in the previous section and the graph in Figure 3.5. Figure 3.6 shows an example of context graph and context vector for the pattern {Terminator}. The context graph is built as a subgraph of the knowledge data source K which contains only the context indicators of α. For the vectorial representation, a one-hot vector is used, that is, a vector with one component for each concept of U, where the component takes the value 1 if it is a context indicator of α; i.e., if the component represents an element of U_α. The context vector comprises 13 elements corresponding to the concepts of U of the knowledge data source shown in Table 3.1 (movies and genres). Particularly, the three vector components that take the value 1 are the components that represent a) the name of the movie; b) the genre Action; and c) the genre Sci-fi. Given the simplicity of this example, both the context graph and context vector are representing the same information, but this will not necessarily be true in all cases. We will be able to check the differences in the information represented in the context graph and context vector once we introduce our method to build these two structures in Chapter 4.

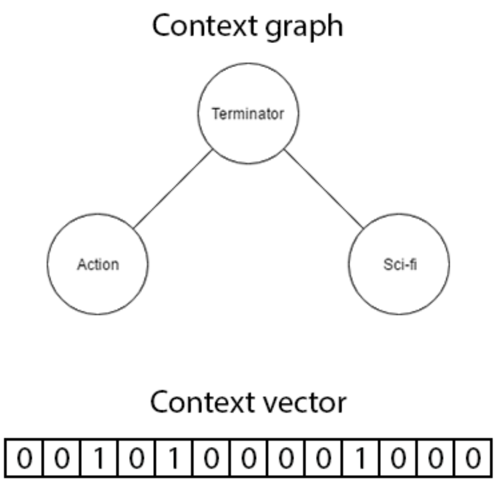

Figure 3.6: Example of context of the pattern {Terminator}

Given the previous definitions, we can now define the process of semantic annotation and its subtasks.

Definition 3. (Semantic Annotation of a pattern). Given a pattern α and the set of frequent patterns P_D retrieved from D, the semantic annotation of α is the task of building a dictionary-like description of α. This description consists of: 1) the subset of context indicators $U'_\alpha \subseteq U_\alpha$ that better describe α; 2) a subset of frequent patterns $P_\alpha \subseteq P_D$ with contexts that are semantically similar to the context of α under some similarity measure; and 3) the set of transactions $T_\alpha \subseteq D$ most representative of α.

Definition 4. (Context Indicators Extraction). Given G_α, the graph representing the context of α, the extraction of context indicators consists of finding the subset $U'_\alpha \subseteq U_\alpha$, s.t. $\forall u' \in U'_\alpha$ and $\forall u \in U_\alpha - U'_\alpha$, $r(u, G_\alpha) \leq (u', G_\alpha)$, where $r(\cdot\,, G_\alpha)$ is a function that measures the relevance of a node with respect to the graph G_α

Definition 4 states that the most representative context indicators of a pattern α are the concepts of U_α which satisfy some measuring condition in the graph G_α.

Definition 5. (Similar Patterns Extraction). Given P_D, the set of frequent patterns of the database D, the problem of extracting semantically similar patterns consists of finding the subset $P_\alpha \subseteq P_D$, s.t. $\forall \gamma \in P_\alpha$, $sim(v_\gamma, v_\alpha) \geq c$, where $sim(\cdot\,,\cdot)$ is a similarity measure between two context vectors and c is the minimum similarity threshold, specified by the user.

According to Definition 5, the semantically similar patterns of a pattern α are those frequent patterns of D that satisfy some similarity condition. This is calculated by pairwise comparison of the context vector of α against the context vector of another frequent pattern.

Definition 6. (Representative Transaction Extraction). Given the transactional database D and a pattern α, the task of extracting the representative transactions for α consists in defining a similarity measure $sim(t_i, v_\alpha)$ between a transaction and the context vector of α, and to find the set of transactions $T_\alpha \subseteq D$, s.t. $\forall t_i \in T_\alpha$, $sim(t_i, v_\alpha) \geq c$, with c being a user-specified threshold.

The definitions introduced in this section are general enough for representing the semantic annotation of frequent patterns in different application domains. The relevance and similarity functions of the three definitions will be selected accordingly to the specific characteristics of the domain.

CHAPTER 4

Semantic annotation method

In this chapter, we will explain our approach to the task of semantic annotation of frequent patterns. We will start by giving a brief overview of the whole process in section 4.1 and presenting the datasets used in our experiments in section 4.2. The remaining sections of this chapter (4.3, 4.4, 4.5, 4.6), will provide in-depth explanations about the different subtasks identified in this process.

4.1 Overview

We will start this chapter with a high-level explanation of the proposed semantic annotation method. This is illustrated in Figure 4.1, an scheme that shows the different subtasks involved in the process of semantic annotation of frequent patterns. The figure also shows the inputs and outputs of each subtask.

The input of the semantic annotation process is a dataset split into a transactional database and a knowledge data source. The transactional database is simply a collection of transactions like, for example, the sales of a particular shop or the movies a user has seen, while the knowledge data source contains semantic information about the items comprised in the transactions. As an example of the kind of semantic information a knowledge data source could contain, we can mention the *price*, *product category*, *time-stamp* of the sales of a shop, or the *movie genres* in the case of the example presented in the preceding chapter. The knowledge data source is depicted as a graph in Figure 4.1 to emphasize that the semantic relationships among concepts of the data source are represented as graphs. Nevertheless, other type of structures such as a relational relational database could be used as a knowledge data source as well, since the contents of the database are representable in a graph.

Before starting the semantic annotation of frequent patterns, these must be extracted from the transactional database. This is done by using frequent itemset mining techniques. Given that these techniques are not the focus of this Master thesis, we will not deepen on this issue. A discussion about the most popular frequent itemset mining techniques can be found in section 2.1.1 of this document. We will just say that, in our experiments, we are using the Eclat algorithm [24] with a relative minimum support of 10%.

The semantic annotation process starts with the context modeling of the frequent patterns; i.e., generating their context graph and context vector. This subtask takes as input the set of frequent patterns and the knowledge data source and produces the graph and vector associated to each pattern (see Definition 2). As we mentioned in the previous chapter, the context graph describes the semantic relationships between the context indicators of the pattern, while the vector is used to measure how relevant the

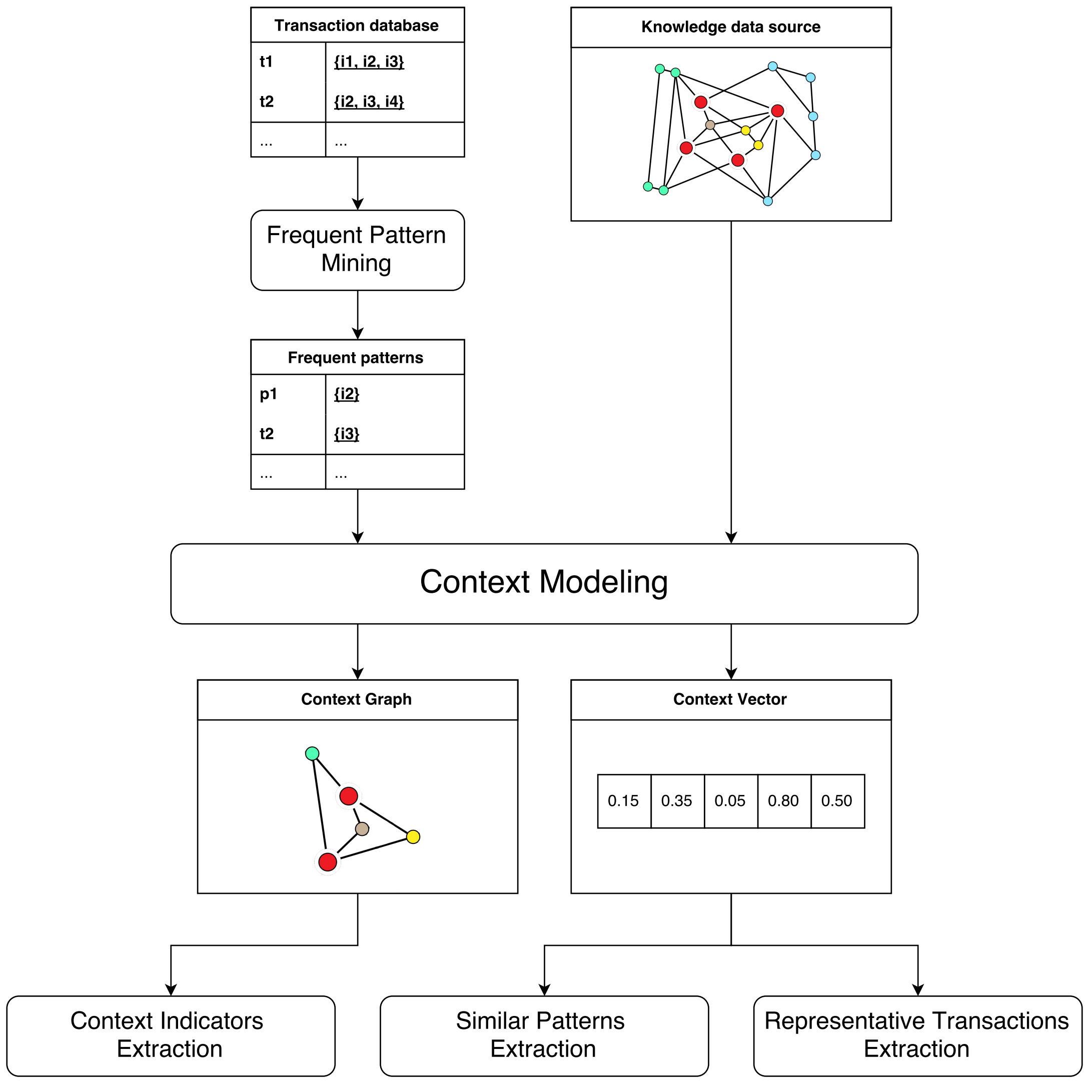

Figure 4.1: Scheme of the semantic annotation process

semantic concepts found in the knowledge data source are for the pattern. Section 4.3 will provide an explanation on how these two representational structures are built.

From this point on, three independent subtasks, one for each of the features of our dictionary-like semantic annotation, are executed. No real ordering is needed for these subtasks, but we will start with the one we consider more important, the context indicators extraction. This subtask takes the context graph of a pattern, and using the method described in section 4.4, identifies the most relevant context indicators that better define the pattern. The remaining subtasks, the extraction of semantically similar patterns and representative transactions, work similarly. They both take the context vectors as inputs and apply similarity measures on the vector space in order to find patterns with similar meanings and the transactions that better reflect the usage of the patterns. This will be explained in detail in sections 4.5 and 4.6, respectively.

4.2 Datasets

As commented in section 4.1, the input data of the semantic annotation of a frequent pattern is a dataset formed by a transactional database and a knowledge data source. In this work, we have used two different datasets: *MovieLens* and *eTourism*. The former is a well-known dataset used as benchmark for many applications, specially for recommendation systems. The latter, on the other hand, is a small dataset comprised of a transactional database and an independent ontology. By using these two distinct datasets, we want to stress that our semantic annotation method is applicable to different kinds of datasets. In the following, we will describe the transactional database and knowledge data source of both *MovieLens* and *eTourism* datasets.

4.2.1. MovieLens

MovieLens is a collection of rating data sets from the *MovieLens* web site [1] made available by the GroupLens Research group. Of the many available datasets, we are using the *MovieLens* 20M Dataset, a stable benchmark dataset with over 20 million ratings and 465 000 tags applied to 27 000 movies by 138 000 users. This dataset is a conventional relational database with the following structure:

- **Ratings data**: all ratings are contained in a table, each row representing the rating given to one movie by one user. Ratings are made on a 5-star scale, with half-star increments on a range from 0.5 to 5.0.

- **Movies data**: movie information is split across two tables, one containing the identifiers and titles of the movies and another relating movies and movie genres.

- **Tags data**: tags are user-generated metadata about movies. Each tag is typically a single word or short phrase. Tag information is contained in a table where each row represents one tag of one movie given by one user.

- **Tag genome data**: the tag genome is a subset of 1100 tags computed through a machine learning algorithm on user-contributed content including tags, ratings, and textual reviews. The tag genome encodes how strongly movies exhibit particular properties represented by tags. Tag genome data are stored in a table that can be seen as a dense matrix, where each movie has a relevance score for every tag in the genome.

In order to use this dataset for the semantic annotation process, we first need to decide what parts of the dataset are going to be considered as the transactional database and as the knowledge data source. For the transactional database, we are using the ratings of a user as a transaction. Concretely, only ratings with a score of 4 stars or higher are considered, so a transaction is regarded as the set of movies a user has mostly liked.

For the knowledge data source, we are using all other tables from the dataset; i.e., the *movies data*, the *tag data* and the *tag genome data*. For instance, the information shown in Table 3.1 depicts the type of *movies data* (movies and genres) that can be found in the dataset. Additionally, Table 4.1 shows an excerpt of the data comprised in the *tag genome* table of *MovieLens*.

[1] http://movielens.org

Movie	Genome tag	Relevance
Pulp Fiction	absurd	0.60925
Pulp Fiction	action packed	0.46725
Pulp Fiction	alien invasion	0.02125
Pulp Fiction	amazing cinematography	0.49075
Pulp Fiction	art	0.5935
Pulp Fiction	assassin	0.70325

Table 4.1: Excerpt from the tag genome table of *MovieLens*

4.2.2. eTourism

The *eTourism* dataset contains two independent sources of data. The first one is a transactional database collected through the Twitter Streaming API [2]. This database contains the spots visited by tourists during their stay in the city of Valencia. Each transaction of this database represents the tourist places visited by a user, such as monuments, museums, and other Points of Interest (POIs), from where a tweet was posted during the user's visit to Valencia. The database contains a total of 1062 transactions and 42 different POIs.

The knowledge data source is a tourism ontology [13] describing the taxonomy of the POIs of Valencia. This taxonomy offers information about the type of POI (museum, open space, etc.) and architectural style (baroque, neoclassic, etc.), and assigns a weight to each category. Figure 4.2 shows a fragment of the ontology we are using where it can be seen how each POI is associated with a small number of categories. For example, the POI 'Valencia Cathedral' is classified as 60% 'Gothic' and 90% 'Church'.

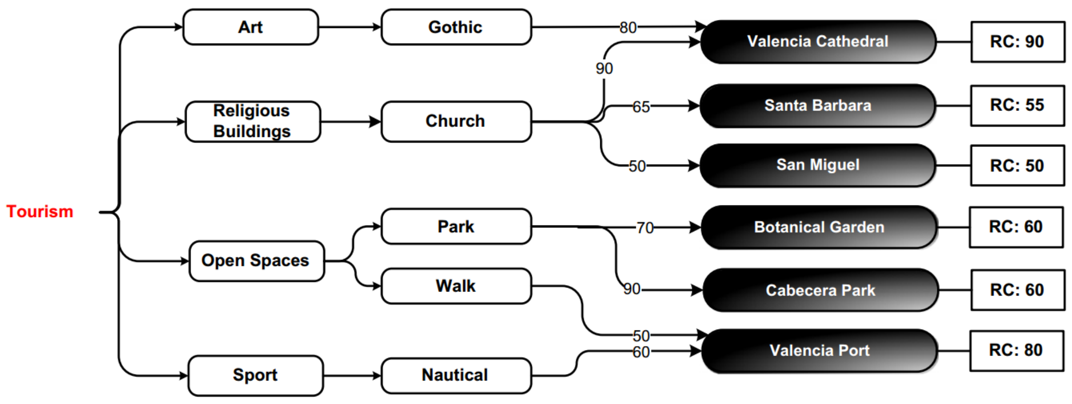

Figure 4.2: Fragment of the tourism ontology used as knowledge data source

As this dataset is already split into transactional database and knowledge data source, there is no need to further differentiate between them and can be used as is.

4.3 Context Modeling

The context of a pattern α, as previously defined, is a tuple $< G_\alpha, v_\alpha >$ composed of a context graph and a context vector. The ***context graph*** provides a representation able to deal with multiple concepts of different nature and the relationships among them.

[2]https://dev.twitter.com/streaming/overview

Consequently, G_α is built as a semantic network; that is, a network that represents semantic relationships between concepts. A semantic network takes the form of a directed or undirected graph where nodes represent concepts or entities, and edges are the semantic relationships between concepts. Figure 4.3 shows an example of a semantic network where different concepts are related to each other.

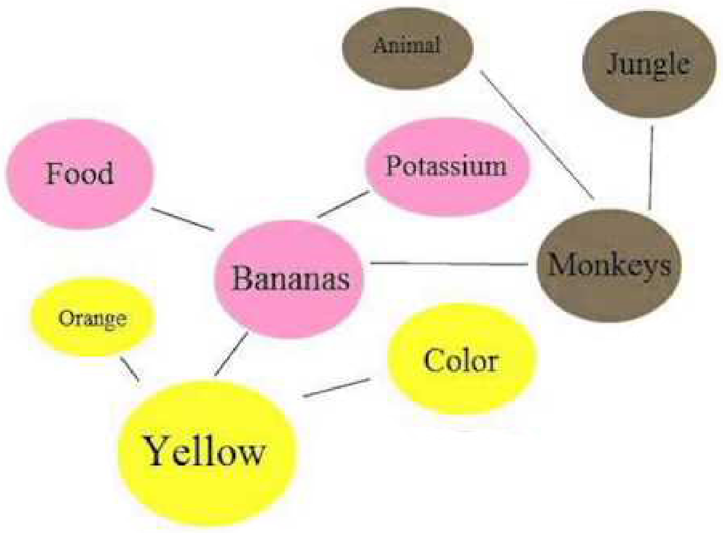

Figure 4.3: Example of semantic network

The ***context vector***, on the other hand, simplifies the task of pattern comparison and allows this process to be done efficiently. Vectorial representations are commonly used in many disciplines such as natural language processing and information retrieval. In a vector space, the similarity between two elements can be computed as the distance between the vectors of both elements. Thus, the context vector provides us with a representation that is easily usable to compare pattern contexts and compute similarities, something that will be needed later in order to extract semantically similar patterns and representative transactions.

In the following sections, we will explain how to build the context graph of a pattern α from its set of context indicators U_α and how to obtain an accurate vectorial representation from the semantic network that models the context of a pattern.

4.3.1. Building a context graph

Building the context graph of a pattern $\alpha = \{i_1, i_2, ..., i_k\}$ is a procedure that takes as inputs the pattern α itself and the knowledge data source K, and iteratively produces the graph G_α. The result of this procedure can be seen as the subgraph of K that will comprise the context indicators U_α more closely related to α. In Algorithm 1, we present the pseudo-code for this procedure.

This algorithm starts by adding the items of α as nodes to G_α and iteratively expands the graph by increasing the distance from the items to the context indicators, understanding distance as the number of edges we have to traverse in K to reach the context indicators from an item. In each iteration, we look for the context indicators at distance d (U_d in the algorithm), and add them as nodes to G_α. The algorithm stops once no more

Algorithm 1 Algorithm to build the context graph of a pattern

```
$d = 0$
$U_d = \alpha$
for $u \in U_d$ do
  add $u$ as a node to $G_\alpha$
end for
while $size(U_d) \neq 0$ AND $d < D$ do
  $U_{d+1} = \{\}$
  for $n \in U_d$ do
    $U' \subseteq U_\alpha$ s.t. $\forall u \in U'$ exists a link between $n$ and $u$ in $K$
    for $u \in U'$ do
      add $u$ as a node to $G_\alpha$
      draw an edge between $u$ and $n$ in $G_\alpha$
    end for
    $U_{d+1} = U_{d+1} \cup U'$
  end for
  $d = d + 1$
end while
return $G_\alpha$
```

context indicators have been found in an iteration or when we reach a maximum distance D, specified by the user.

Let us illustrate this process with an example. If we were to build the context graph of $p_1 = \{\text{Terminator}\}$ from the illustrative example of section 3.1, we would perform the following steps:

- **Initialization**: Add "Terminator" as a node to the context graph G_{p_1}
- **Iteration 1**: Look for the context indicators at distance 1 from "Terminator", which are $U_1 = \{$"Action", "Sci-fi"$\}$, and add them to the graph drawing the edge between them and "Terminator" as well.
- **Iteration 2**: In the second iteration, we would find "Matrix" at distance 1 from "Action", and "Matrix" and "Alien" at distance 1 from "Sci-fi", so $U_2 = \{$"Matrix", "Alien"$\}$.
- **Iteration 3**: In this iteration, we would only find "Horror" as a context indicator connected to "Alien".

The algorithm will stop after the third iteration, since no more new context indicators will be found. This example is illustrated in Figure 4.4, where the nodes and edges added in each iteration are painted in blue.

Before building the context graph, it is recommended to select the context indicators that will be used to model the context of the patterns. Of course, the whole set of context indicators can be used, but depending on the available data and on the information we are interested in, it may turn out to be more useful to run a prior selection of context indicators. For our two datasets, *MovieLens* and *eTourism*, we followed different approaches.

In the case of *MovieLens*, the available types of context indicators are *users*, *genres*, *tags*, *genome tags* and other *movies*. *MovieLens* is a good example of how our method can be applied to most common databases, but this dataset has some peculiarities which arise from its utilization in recommendation systems: most tables in the *MovieLens* dataset can

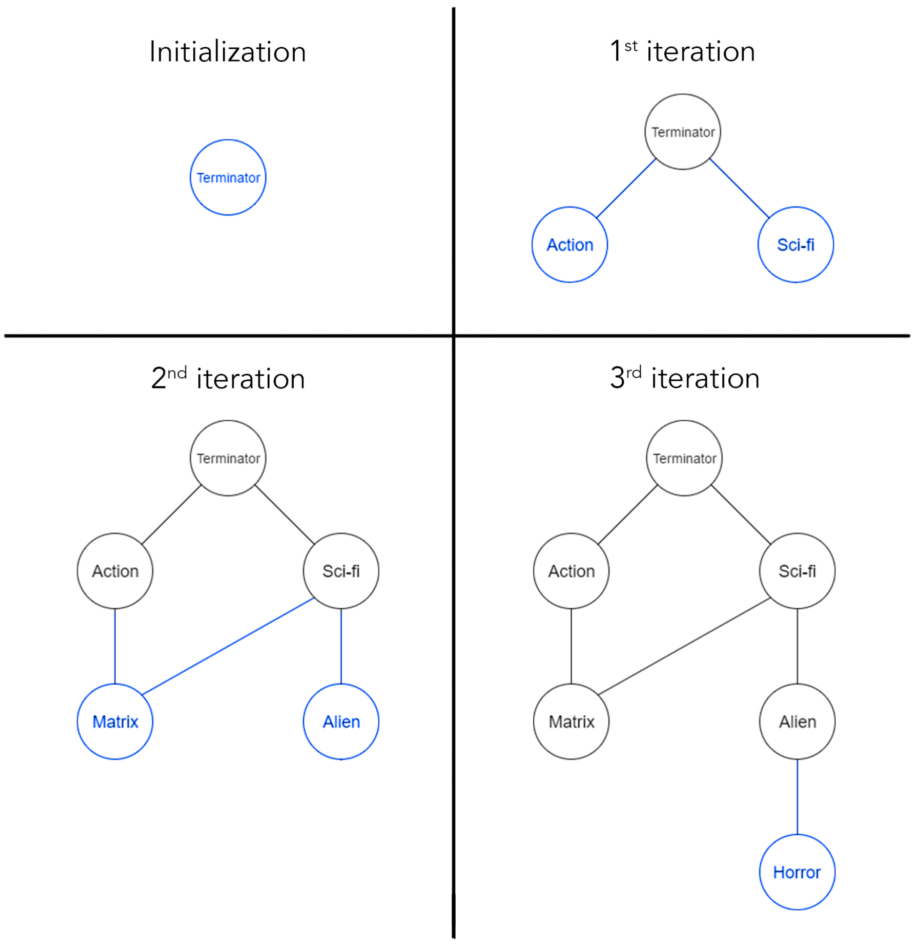

Figure 4.4: Building the context graph of pattern {Terminator}

be seen as dense matrices. That is, as we saw in section 4.2.1, each movie is associated to every *genome tag* along with the relevance score of each tag to the movie. If we used the *genome tags* as context indicators without any prior filtering, all the *movies* of a pattern would be linked to the whole set of *genome tags*, and each *genome tag* linked to all the *movies* in the dataset. This would lead to context graphs that containing most of the concepts of the knowledge data source and with little discriminative power. A similar situation would arise when using *users* as context indicators, since *MovieLens* is a recommendation database, and most users in the database have a large number of rated movies. For our experiments, we have chosen to use *movies* and *genome tags* as context indicators, and only those *genome tags* with a relevance score no less than 0.95. This can be seen as using only the *genome tags* that better describe a *movie*. Also, since this filtering implies we are only considering score values in the range 0.95 to 1, there would be little variability in the weight of the edges of the graph, so we are using unweighted graphs for the case of *MovieLens*.

For the *eTourism* dataset, we are using an ontology as the knowledge data source, which is independent from the transactional database. This ontology presents a taxonomy of POIs, and holds information about the degree of adequacy of each POI of the city of Valencia to a particular category (e.g., park, church, gothic, etc.). Contrary to the case of *MovieLens*, the data in *eTourism* are very sparse, with each POI being related to only a couple of categories. Then, that there is no need to filter the context indicators used. Also, the degree of adequacy to a category can be used as the weight of an edge, which allows us to test our method on weighted graphs. Figure 4.5 shows an example of the context graph of one of the frequent patterns mined from *eTourism*.

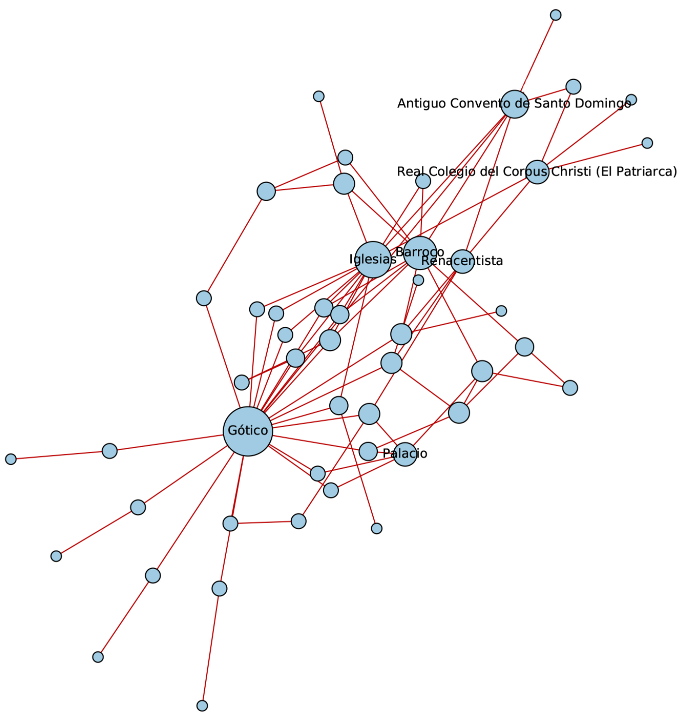
Antiguo Convento de Santo Domingo
Real Colegio del Corpus Christi (El Patriarca)
Iglesias
Barroco
Renacentista
Gótico
Palacio

Figure 4.5: Context graph of the frequent pattern {Catedral}

In both cases, the maximum distance of a pattern item to a context indicator is set to 3. This is because we want to balance direct and indirect relationships. Shorter distances favor more direct relationships in the context graph so the semantics uncovered with short distances may not be very interesting. On the other hand, longer distances promote the appearance of context indicators weakly related to the pattern items or with no significant connection, which leads to pattern contexts cluttered with irrelevant information. Hence, each particular dataset must be analyzed so as to choose the proper distance that would allow us to uncover not only direct relationships, but also relevant secondary relationships.

4.3.2. Building a context vector

The context vector v_α of a pattern α is a vectorial representation derived from its context graph G_α but, as we said in Chapter 3, the information comprised in the vector should not necessarily be the same as in the graph. The vectorial representation assigns numerical values to the concepts in the knowledge data source K, effectively allowing us to measure how closely related these concepts are to α. The number of concepts in a knowledge data source is usually too large to build a vector with one component for each concept, so a way to reduce its dimensionality while maintaining the semantic relationships is needed.

In order to build the context vector, we propose to use the communities that can be inferred from the knowledge data source K. Communities are sets of highly interconnected nodes that are found in complex networks. Hence, communities found on a semantic network can be understood as sets of strongly related concepts from the semantic standpoint and can be used to reduce the dimensionality of the context vector.

First, let us briefly explain the problem of community detection. This problem consists in partitioning a network in communities of densely connected nodes, with sparse connections to nodes of other communities. The quality of a partition is measured by its modularity, which is a scalar value between -1 and 1 that measures the density of links inside communities as compared to links between communities. The modularity is defined as

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j)$$

where A_{ij} represents the weight of the edge between i and j; $k_i = \sum_j A_{ij}$ is the sum of the weights of the edges attached to vertex i; c_i is the community to which vertex i is assigned; the δ-function $\delta(u, v)$ is 1 if $u = v$ and 0 otherwise; and $m = \frac{1}{2} \sum_{i,j} A_{ij}$.

Several algorithms exist in the literature of community detection, but the one we have chosen is the hierarchical method proposed in [6]. This algorithm is initialized by assigning each node in the network a different community, so there are as many communities as nodes. Afterwards, the following two phases are repeated iteratively:

- **1st phase**: The gain in modularity from moving one node to an adjacent community is computed for each node and the node is assigned to the community with bigger improvement in modularity. If no positive gain is possible, the node remains in its original community.

- **2nd phase**: A new graph is built, with a node for each community resulting from the previous step.

In each iteration, the two phases are applied to this new network, which means that smaller communities are merged together into bigger ones. It is at this point where the algorithm adopts its hierarchical nature. The process is repeated until there is no improvement in modularity during one iteration.

The algorithm does not guarantee an optimal partition and will generate slightly different partitions each time it is executed. This is because the output of the algorithm is dependent on the order in which nodes are considered. Nonetheless, authors claim that the bulk of the communities remain the same along different executions and we, too, have been able to verify this fact.

Let us explain now how this algorithm is used to build the context vector v_α. To begin with, we need to represent our knowledge data source K as a graph G_K. Note that G_K will contain all of the nodes and edges of the context graphs that we have previously built for our patterns. The community detection is performed over G_K, which is then split in several subgraphs G_{c_i}, one for each community. The context vector v_α is built comparing the context graph G_α of a pattern α to each community graph in the following way.

Given a context graph G_α and the set of community subgraphs $\{G_{c_i}, G_{c_2}, ..., G_{c_n}\}$ extracted from G_K, the context vector of pattern α is a vector $v_\alpha = \langle w_1, w_2, ..., w_n \rangle$, with each component w_i being computed as

$$w_i = \frac{|E_{G_\alpha} \cap E_{G_{c_i}}|}{|E_{G_{c_i}}|}$$

where E_{G_i} is the set of edges of graph G_i.

This way, each component of our context vector measures the degree of belonging of the pattern α to a community. As we previously said, a community can be understood as a set of closely related concepts, so the context vector is effectively measuring how relevant the concepts found in the knowledge data source K are for the given pattern. Figure 4.6 presents an example of a community graph extracted from the *MovieLens* dataset. This community graph is built around the concept "Science fiction" and many related context indicators can be found in it. To illustrate this, we have highlighted two sections of the

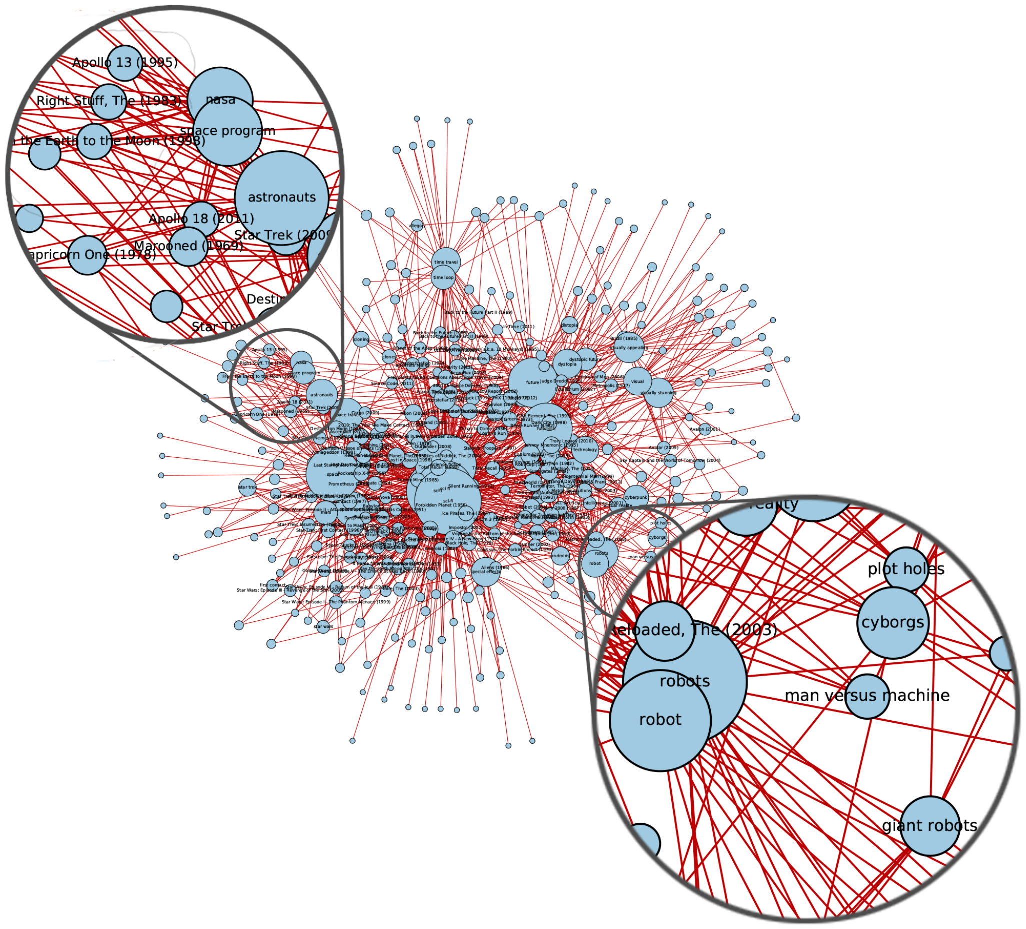

Figure 4.6: Example of community graph extracted from the *MovieLens* dataset

graph where context indicators such as "nasa", "space program", "astronauts", "robots", and "cyborgs" can be seen.

Note also that a patten can be a single item or a whole transaction, so this representation is used to compare patterns of any size independently of whether they are items, patterns or transactions. This representation will prove very useful in the subtasks to come, since it will allow us to efficiently compare patterns and transactions of different sizes.

4.4 Context indicators extraction

The most significant feature of the semantic annotation is the set of strongest context indicators, which serve as a definition of the pattern. In the context graph of a pattern, several context indicators can be found, but not all of them are equally relevant. The strongest context indicators are the most relevant nodes of the graph, which means that they are the nodes that best represent the semantics of the pattern.

In graph theory, the "relevance" of a node is referred to as the centrality of the node. The concept of centrality is based on three ideas: (1) a central node has more connections to other nodes, (2) the paths from the central node to other nodes are shorter, and (3) a central node can control the flow between other nodes. Based on these three features, the work in [12] proposed three different measures of node centrality: *degree*, *closeness*, and *betweenness*.

The *degree* of a node was the first centrality measure proposed, and, conceptually, it is the simplest measure. The *degree* is defined as the number of nodes that a node is connected to. This measure can be interpreted as the involvement of a node in the network. For example, in a graph representing friendship relationships in a social network, a node with high degree would indicate a popular person. Since only the local structure around a node is needed, the computation of this measure is simple and fast; but, the *degree* measure also has some shortcomings; e.g., it does not take into account the global structure of the graph. This means that a node connected to many but far from most of the nodes in a graph may be deemed important by this measure, while it is unable to efficiently reach other nodes.

To address the issue of *degree* centrality, a second measure was defined, *closeness*. The closeness of a node is the average length of the shortest path between the node and all the other nodes in the graph, and it is computed as the inverse sum of the shortest distances from the node to the rest of nodes in the graph. This means that the closeness of a node cannot be computed for graphs with disconnected components, which severely limits its applicability.

The last of the three measures, *betweenness*, measures the degree to which a node is found on the shortest path between two other nodes. This measure can be interpreted as the capability of a node to control the flow of information in a graph. This measure also takes into consideration the global structure of the network but, contrary to *closeness* centrality, it can be applied to graphs with disconnected graphs. Nonetheless, this measure also presents some limitations. *Betweenness* centrality is set to 0 for a node that does not lie on the shortest path between any two other nodes and, in general, this is found in a great proportion of nodes in a network, which reduces the discriminative power of this metric.

The three centrality measures were originally defined for unweighted graphs, but have since been generalized for weighted graphs. Degree centrality for weighted graphs is computed as the sum of the weights of the incident edges [5], while Dijkstra's algorithm of shortest paths is used for closeness [18] and betweenness [7] centrality.

Any of the presented variations of centrality can be used as the relevance function $r(\cdot, G_\alpha)$ (see Definition 4) to measure the relevance of a node with respect to the context graph G_α and extract the strongest context indicators. Each of these metrics is able to highlight different semantics of a pattern, so we will compare them in Chapter 5, discussing the advantages and disadvantages of each one.

Examples

We will now show some examples of the extraction of context indicators of a particular pattern.

For our first example we are using the frequent pattern {Godfather, The (1972) | Forrest Gump (1994)} extracted from the *MovieLens* dataset and its top 10 context indicators (Figure 4.7). Both movies in the pattern are very different in content, so it is interesting to see what information the context indicators will unveil. As we can see in the figure, instead of focusing on the contents of the movies, the majority of the context indicators are centered around the awards both movies won and other quality-related concepts such as "masterpiece", "great acting", and "imdb top 250". Therefore, the context indicators enable to showcase what these two movies have in common, which in this case reveal that the two movies are regarded as great movies.

Pattern: {Godfather, The (1972) | Forrest Gump (1994)}
Context indicators:
imdb top 250 | oscar (best supporting actor) | oscar (best directing) | crime
oscar (best picture) | highly quotable | masterpiece | great acting
organized crime | mafia

Figure 4.7: Context indicators of pattern {Godfather, The (1972) | Forrest Gump (1994)}

For our second example, we are going to pick a pattern comprised of only one item in order to see that our method is also able to describe single items as opposite to the previous example where the pattern is a two-item set and so the indicators describe the common features of both items. Figure 4.8 shows the context indicators of the frequent pattern {Jardines del Turia} from the *eTourism* dataset. The extracted context indicators show the type of the POI "Jardines del Turia" using concepts such as "Estadios y áreas deportivas", "Paseos", and "Parques". Moreover, other related POIs like "Estadio Mestalla" and "Bioparc" can also be found among the context indicators.

Pattern: {Jardines del Turia}
Context indicators:
Futbol | Estilos y periodos | Estadios y áreas deportivas | Estadio de Mestalla
Contemporáneo | Complejo Deportivo Cultural Petxina | Playas | Bioparc
Paseos | Parques

Figure 4.8: Context indicators of pattern {Jardines del Turia}

4.5 Similar patterns extraction

The next step in the process of annotating a frequent pattern is to find the patterns that are semantically similar to a given pattern α. In order to do this, we make the assumption that two patterns are similar if their context vectors are similar to each other under some similarity measure.

Numerous similarity measures have been proposed, each one with its advantages and limitations, applicable to different types of data. In the case under study, we are interested in finding the similarity between vectors of real numbers.

One of the most extended similarity measures is the cosine similarity. which is used to compute the similarity between two vectors. The cosine similarity is often used in text mining for document comparison. In this work, we use cosine similarity of two context vectors to measure the semantic similarity of two frequent patterns. The cosine similarity of the context vectors of two patterns, α and β, is defined as

$$sim(v_\alpha, v_\beta) = \frac{\sum_{i=1}^{n} a_i * b_i}{\sqrt{\sum_{i=1}^{n} a_i^2} * \sqrt{\sum_{i=1}^{n} b_i^2}}$$

where $v_\alpha = \langle a_1, a_2, ..., a_n \rangle$ and $v_\beta = \langle b_1, b_2, ..., b_n \rangle$.

Now that we have a way to compute the similarity between two frequent pattern, all that remains is to decide which patterns are similar under this measure. For this, we need to define a minimum similarity threshold c, that in our experiments is set to 0.9. Taking this threshold, two patterns, α and β, are semantically similar if $sim(v_\alpha, v_\beta) \geq 0.9$.

Examples

Following, we present two examples of the application of the similar pattern extraction method to our test datasets. Figure 4.9 shows a single-item frequent pattern from the *MovieLens* dataset and its similar patterns. The movie in the pattern belongs to the *The Lord of the Rings* (LotR) franchise and specifically shows the movie of 2002 entitled "LotR: The Two Towers". As can be observed in Figure 4.9, the similar pattern extraction method has detected that all the movies in the franchise are semantically similar. Hence, the similar patterns detected are those comprised of other movies in the franchise and their combinations.

Pattern: {LotR: The Two Towers (2002)}
Similar patterns:
{LotR: The Return of the King (2003)}
{LotR: The Fellowship of the Ring (2001)}
{LotR: The Two Towers (2002) | LotR: The Fellowship of the Ring (2001)}
{LotR: The Return of the King (2003) | LotR: The Two Towers (2002)}

Figure 4.9: Patterns similar to {LotR: The Two Towers (2002)}

The example of Figure 4.10 shows the similar patterns of the frequent pattern {Plaza del Ayuntamiento | Ciudad de las Ciencias} from the *eTourism* dataset. Considering that the knowledge available for this dataset is limited to the type of POI, we can infer some associations between items from this example. If we compare the frequent pattern with its similar patterns, we find that "Plaza del Ayuntamiento" can be exchanged by "Plaza de la Reina", and "Ciudad de las Ciencias" by "Oceanográfico" and still maintain the meaning of the pattern.

Pattern: {Plaza del Ayuntamiento | Ciudad de las Ciencias}
Similar patterns:
{Plaza de la Reina | Ciudad de las Ciencias}
{Plaza de la Reina | Plaza del Ayuntamiento | Ciudad de las Ciencias }
{El Oceanográfico | Plaza del Ayuntamiento}
{Ciudad de las Ciencias}
{El Oceanográfico}
{El Oceanográfico | Ciudad de las Ciencias }

Figure 4.10: Patterns similar to {Plaza del Ayuntamiento | Ciudad de las Ciencias}

4.6 Representative transactions extraction

The last feature of our semantic annotation is the representative transactions. A transaction is a set of items representing some information about a user. For example, in the case of *MovieLens* a transaction represents the movies a user has mostly liked, while in *eTourism* a transaction is the set of POIs a user has visited. We refer to the set of transactions that better represent the usage of a pattern α as the ***representative transactions*** of α. Since we are working with contexts, the representative transactions are the transactions whose contexts are closer to the context of a given pattern, even if that pattern is not found within the transactions items.

A transaction can be considered as a pattern, since it is an itemset. The main difference between frequent patterns and transactions lies in the size, which usually does not go beyond 3 for frequent patterns but it can be much larger for transactions. As such, the context modeling method we presented in Section 4.3 can also be used to model the context of a transaction.

Therefore, we build the context vector of a transaction the same way as we did for frequent patterns and we use the cosine similarity to compute the representativeness of a transaction, through a process very similar to the one shown in the previous section. The only difference is that instead of computing the similarity between the context vectors of two frequent patterns, we compute the similarity between the context vector of a frequent pattern and that of a transaction.

This method to build the context vector, however, may not be the most appropriate for transactions when dealing with large itemsets. Say, for example, we want to build the context graph of two transactions, where the first one has nine movies of genre Action and a single movie of genre Romance, and the second one has one movie of genre Action and one movie of genre Romance. Many context indicators will be shared among the movies of genre Action, so the context graph will not vary much between one transaction and the other even though one is clearly more oriented towards this genre than the other. Of course, this problem will also be observed in the context vectors, since they are derived from the context graphs.

To address this issue, we compute the context vector of a transaction as the average of the context vectors of the items in the transaction. In order to do this, we build the context graphs of the items in the transaction as if they were patterns of one single item, and compute their average. Formally, the context of a transaction $t_i = \{i_1, i_2, ..., i_k\}$ is a vector $v_{t_i} = \langle w_1, w_2, ..., w_n \rangle$ with

$$w_i = \frac{\sum_{j=1}^{k} v_{ji}}{k}$$

where v_{ji} is the component i of the context vector of item j, and k is the size of the transaction. This method provides a more accurate representation of the context of a transaction by taking into consideration the occurrences of the context indicators.

Having defined how to build the context vector of a transaction, we need now to define a minimum similarity threshold, that for consistency will remain 0.9, and compute the set of representative transactions. This can be done by finding all the transactions that meet $sim(v_\alpha, v_{t_i}) \geq 0.9$, in a similar way to the method described in the previous section.

Examples

Let us show the result of the extraction of representative transactions with two examples. The first example, shown in Figure 4.11, lists the two more representative transactions for the frequent pattern {Alien (1979)} from the *MovieLens* dataset. This is an interesting example because it shows two extreme cases of what is usually found among the representative transactions, which are small transactions containing the frequent pattern itself, and large transactions comprised of many items with similar contexts.

Pattern: {Alien (1979)}
Representative transactions:
{Alien (1979)}
{Night of the Living Dead (1968) | Howling, The (1980) | Alien (1979) | Psycho (1960) | Shining, The (1980) | Birds, The (1963) | Bride of Frankenstein, The (1935) | Jaws (1975) | Exorcist, The (1973) | Thing, The (1982) | King Kong (1976)}

Figure 4.11: Representative transactions for the frequent pattern {Alien (1979)}

Our second example, shown in Figure 4.12, displays much of the same behavior observed in the previous example. In this case the frequent pattern is {Playa de la Malvarrosa} from the *eTourism* dataset, and we can see once again how among the representative transactions we find the pattern itself and other items related to "Playa de la Malvarrosa" by the concept "open-spaces".

Pattern: {Playa de la Malvarrosa}
Representative transactions:
{Playa de la Malvarrosa}
{Playa de las Arenas}
{Playa de las Arenas | Jardines del Turia}
{Jardines del Turia}

Figure 4.12: Representative transactions for the frequent pattern {Playa de la Malvarrosa}

4.7 Conclusions

Before moving on to the next chapter, let us summarize what we have done in this process and what we have achieved in doing so.

1. **Extraction of context indicators**. This process takes the context graphs of the frequent patterns as inputs and, using centrality measures to compute the relevance of the nodes in the graph, elicits the strongest context indicators that better define the pattern.

2. **Extraction of similar patterns**. In this process, we obtain the patterns more similar to one given. This is done by pairwise comparison of the similarity between context vectors of frequent patterns.

3. **Extraction of representative transactions**. This process elicits the transactions of a dataset that are most representative to a given pattern. It is a process alike to the similar patterns but in this case the itemsets involved are much larger. This process tends to associate the transactions whose items share a more similar context to the given pattern.

CHAPTER 5

Relevance assessment of the results

This chapter is devoted to analyzing the results we have obtained from the application of the semantic annotation method described in the previous chapter. The chapter is divided in three sections, one for each of the features of the semantic annotation of a frequent pattern.

The first section, section 5.1, presents the results of the application of the three different centrality measures for the extraction of context indicators. We will use an example with three frequent patterns to compare the behavior of the three metrics and we will discuss the semantic information that is obtainable when using each metric.

Sections 5.2 and 5.3 offer a quantitative analysis of the quality of the similar patterns and representative transactions. For this analysis we are considering the ratings given by the users to the movies in the *MovieLens* dataset as ground truth and, based on this information, we present two methods to validate the results. Given that there is no available information in the *eTourism* dataset to verify the results of the semantic annotation process, this evaluation will be limited to the *MovieLens* dataset.

We will also analyze how the size of the context vector impacts the results. As the size of the context vector depends on the number of communities found in the context data source, we can modify this parameter by setting the level of the communities during the application of the community detection algorithm. For example, we can use bottom level communities by stopping the algorithm after the first iteration, or let it run all iterations to obtain top level communities.

5.1 Centrality measures for context indicators extraction

In this section, we are going to discuss the effect of the chosen centrality measure on the extraction of context indicators. This evaluation will be performed in a qualitative way, since this is a novel problem and, to the best of our knowledge, there exists no corpus that we can use to validate our results.

Before we start with the discussion we want to show some numerical values about the contexts graphs obtained in both datasets to put things into perspective. Table 5.1 shows the average, minimum, and maximum order (i.e., number of nodes) of the context graphs for each dataset. These results, paired with an average degree per node of 3.24 for *MovieLens*, and 1.29 for *eTourism*, shows that the contexts graphs in *MovieLens* are much larger and with many more connections than those of *eTourism*. This difference is given by the knowledge data sources, whereas we are considering over 20 000 context indicators for *MovieLens*, there are only a little over 100 context indicators for *eTourism*.

Dataset	Average	Minimum	Maximum
MovieLens	1470	127	2628
eTourism	61	6	161

Table 5.1: Order of the context graphs in the test datasets

In order to compare the three centrality measures presented in section 4.4, we are going to analyze how they behave on three single-item patterns. We have intentionally selected three similar movies with slight differences in order to test if these measures are able to obtain the context indicators that better define each pattern. The movies used in the example are "The Lion King", "Toy Story", and "Beauty and the Beast". All three are animation movies geared towards children, but they differ in content and animation style.

Let us start by analyzing the context indicators extracted with *degree* centrality. Table 5.2 shows the context indicators we have extracted using this centrality measure for the three pattern. The first thing that we notice is that almost all the context indicators are the same across the three patterns. Moreover, some context indicators like "computer animation" and "talking animals" appear in all three cases even though the first one should only appear for "Toy Story" and the second one for "The Lion King". The reason for this lies in the local nature of the *degree* centrality, which is unable to take into consideration the global structure of the context graph. As the three movies are similar, the context graphs contain mostly the same context indicators and the metric is not able to distinguish the indicators that are more important for each pattern. Hence, *degree* centrality favors more general concepts since those are the ones with more connections, independently of how related they are to the pattern. It also happens that the nodes with more connections are of the type *genome tag*, so the set of strongest indicators is comprised entirely of nodes of this type.

Pattern	Context indicators
{The Lion King)}	animation | cartoon | kids and family | animated | computer animation | talking animals | animals | disney animated feature | children | oscar (best animated feature)
{Toy Story}	animation | cartoon | computer animation | kids and family | animated | talking animals | disney animated feature | children | oscar (best animated feature) | animals
{Beauty and the Beast}	animation | cartoon | kids and family | animated | computer animation | fairy tale | disney animated feature | talking animals | oscar (best animated feature) | animals

Table 5.2: Context indicators obtained by using *degree* centrality

Moving now to *closeness* centrality, we can see in Table 5.3 that the context indicators are comprised mostly of other animation movies. This happens because the nodes found at the core of the structure of the context graph are context indicators of the same type as the item and, as the *closeness* centrality is based on the closest distance between nodes, these are the ones deemed more important by this measure.

Pattern	**Context indicators**
{The Lion King)}	animation | Lion King, The (1994) | Finding Nemo (2003) | Toy Story (1995) | Monsters, Inc. (2001) | Lilo & Stitch (2002) | 101 Dalmatians (One Hundred and One Dalmatians) (1961) | Charlotte's Web (1973) | Peter Pan (1953) | Stuart Little (1999)
{Toy Story}	animation | Finding Nemo (2003) | Toy Story (1995) | Monsters, Inc. (2001) | Lion King, The (1994) | Lilo & Stitch (2002) | Toy Story 2 (1999) | Stuart Little (1999) | 101 Dalmatians (One Hundred and One Dalmatians) (1961) | Ice Age (2002)
{Beauty and the Beast}	animation | Beauty and the Beast (1991) | Aladdin (1992) | Shrek (2001) | Sleeping Beauty (1959) | Pinocchio (1940) | Snow White and the Seven Dwarfs (1937) | Lion King, The (1994) | Finding Nemo (2003) | Shrek 2 (2004)

Table 5.3: Context indicators obtained by using *closeness* centrality

The last metric, *betweenness* centrality, is based on the ability of a node to control the flow of information. This metric assigns high values to nodes with many connections that are found in the thick of the graph and can be seen as a middle ground between the other two. The results of using *betweenness* centrality to extract the context indicators are presented in Table 5.4. We want to highlight two things about this centrality measure. The first one is that this metric is able to find the context indicators more suitable to the given pattern. Examples of this are "animals" and "talking animals" for "The Lion King", "computer animation" for "Toy Story", and "fairy tale" for "Beauty and the Beast". The second thing we want to highlight is that the context indicators are comprised of both *genome tags* and other *movies* offering a good mix between the *genome tags* that better describe the contents of the movie and other closely related *movies*.

Pattern	**Context indicators**
{The Lion King)}	animation | kids and family | children | cartoon | animals | kids | talking animals | Lion King, The (1994) | Finding Nemo (2003) | disney animated feature
{Toy Story}	animation | kids and family | children | cartoon | computer animation | kids | Finding Nemo (2003) | Tron (1982) | Toy Story (1995) | talking animals
{Beauty and the Beast}	animation | fairy tale | kids and family | cartoon | disney animated feature | Sleeping Beauty (1959) | Beauty and the Beast (1991) | Pan's Labyrinth (Laberinto del fauno, El) (2006) | Aladdin (1992) | Shrek (2001)

Table 5.4: Context indicators obtained by using *betweenness* centrality

Summarizing, *degree* centrality obtains general descriptions, *closeness* centrality describes the pattern in terms of items of the same type, and *betweenness* centrality offers a good balance between the other two while also offering more specific descriptions. In our tests, *betweenness* centrality has obtained the results that we believe are the most significant and meaningful, but the other two measures can be useful too depending on the needs of the problem.

5.2 Similar patterns evaluation

The similar patterns are the frequent patterns that share the same meaning as a given one. Similar patterns are very useful to summarize the output of frequent pattern mining algorithms, since we can group them together into clusters without loss of information. In doing so, we are compacting the output of this type of algorithms and making it easier for humans to process it, which is the motivation of this work.

In order to evaluate the quality of the extracted semantically similar patterns, we propose to use the confidence of the rule $\alpha \implies \gamma$, where α is a frequent pattern and γ is a pattern similar to α. The confidence of a rule is computed as

$$conf(\alpha \implies \gamma) = \frac{sup(\alpha \cup \gamma)}{sup(\alpha)}$$

with $sup(\alpha)$ being the support of α, that is, the number of occurrences of α in the transactional database; and $sup(\alpha \cup \gamma)$ the number of transactions that comprise both α and γ.

Given that γ is semantically similar to α, in other words, synonyms, we can assume that $\alpha = \gamma$ so $conf(\alpha \implies \gamma) = conf(\alpha \implies \gamma) = 1$. This means that the quality of the similar patterns extracted will be good when the confidence is close to 1.

Formally, the quality of a set of similar patterns is computed as follows. Given a pattern α and the set of its similar patterns P_α, the quality of P_α is computed as

$$qual(P_\alpha) = \frac{1}{N} \sum_{\gamma \in P_\alpha} conf(\alpha \implies \gamma)$$

where N is the size of P_α.

We have applied this method to evaluate the similar patterns we have extracted during the semantic annotation of the *MovieLens* dataset. In order to compute this metric, we consider $sup(\alpha)$ as the set of users (transactions) who have liked (i.e., 4-star or higher rating) the movies of a frequent pattern α and $sup(\gamma)$ as the set of users (transactions) who have liked the movies of a frequent pattern $\gamma \in P_\alpha$.

We have also tested the impact of the size of the context vector on the quality of the similar patterns. This parameter is dependent on the level of the communities used to build the context vector. In the hierarchical community detection algorithm we are using, higher level communities are built by merging lower level ones, which means that the higher we go in the hierarchy the less number of communities that are going to be found at the level. Since in this algorithm the level of the community is determined by the iteration they were found in, we can increase or decrease the size of the context vector by stopping our algorithm at specific iterations.

For our evaluation we have tested the following community levels:

- **Top level communities**: These are the communities found during the last iteration of the community detection algorithm. As they are high level communities, they usually agglomerate several more specific concepts into a more general one.
- **Bottom level communities**: These are the communities found after the first iteration of the community detection algorithm. Bottom level communities are comprised of context indicators very closely related and usually represent specific concepts.

Table 5.5 shows the average quality of the similar patterns of the 477 frequent patterns found in *MovieLens*. The first column shows the level of the communities used to build the context vector and on the second column we can find the number of communities found, which corresponds to the size of the context vector. Column 3 corresponds to the average size of P_α, and columns 4 and 5 show the average $qual(P_\alpha)$ for all frequent patterns when considering only the top-5 most similar patterns in P_α and all the similar patterns, respectively.

Level of the communities	Size of v_α	Average size of P_α	Top 5	All
Top level	38	46	0.8753	0.7788
Bottom level	645	3	0.9469	0.9412

Table 5.5: Quality of the similar patterns extracted from *MovieLens*

The results in Table 5.5 show that the quality of the similar patterns found increases with the size of the context vector. This is because the higher the size of the context vector, the lower the size of the communities, which means that each community represents more specific concepts, resulting in more accurate context vectors. An interesting observation can be made when comparing the difference between the fourth and fifth columns. We can see that the quality of the top-5 similar patterns is higher than the quality of the whole set of similar patterns. This is a good indicator that our ranking system is working properly, since the quality of the similar patterns that rank higher are better.

Looking at the third column, we see that the average size of the set of similar patterns P_α is much higher for smaller context vectors. In fact, we have been able to identify clusters of up to 129 similar patterns among the 477 frequent patterns of the *MovieLens* dataset when using top level communities, while the largest cluster was comprised of 19 similar patterns for bottom level communities. This means that the size of the context vector has an impact not only on the quality of the results, but also in the number of similar patterns found. Hence, there exists a trade-off between quality and compression power when it comes to similar patterns, and this can be balanced by adjusting the size of the context vector.

5.3 Representative transactions evaluation

The last step in the assessment of our results is evaluating the representative transactions extracted with our method. The representative transactions of a pattern are the transaction which contexts are closer to the context of the given pattern. In order to do this evaluation, we will measure the similarity between the context of a pattern and the context of a transaction through the similarity observed in the ratings given by the users to the movies of the pattern and the transaction.

For this purpose, we are going to build a new vector for each pattern α and its representative transactions T_α using the ratings that users have given to the movies in the pattern and transactions. We will assume that the results are good if there exists a correlation between the results obtained by using contexts with respect to those obtained by using ratings.

The process is as follows:

1. Given a subset of users $S = \{s_1, s_2, ..., s_m\}$ we will build, for each movie in the dataset, a rating vector $\langle r_1, r_2, ..., r_m\rangle$ where r_i is the rating user s_i has given to the movie, or 0 if he has not rated it.

2. Compute the rating vectors for the frequent patterns and representative transactions as the average of the rating vectors of the movies in the pattern or transaction, respectively.

3. Compute the cosine similarity between the rating vector of the frequent pattern and those of its respective representative transactions.

If the results are close to 1, it means that the correlation we have inferred at the context level is also observed when using the ratings, and our results are good.

In Table 5.6 we present the results of applying this methodology to the 477 frequent patterns in *MovieLens* and their corresponding representative transactions by using a subset of 10 000 users selected randomly. The first column shows the level of the communities used to build the context vector, the second column shows the size of the context vector, the third column shows the average size of the set of representative transactions T_α, and the fourth, fifth, and sixth columns show the average quality value obtained when evaluating the top-1 representative transaction, top-5 representative transactions, and all the representative transactions, respectively.

Level of the communities	Size of v_α	Average size of T_α	Top 1	Top 5	All
Top level	38	3591	0.7584	0.7349	0.7296
Bottom level	648	7	0.8863	0.8731	0.8710

Table 5.6: Quality of the representative transactions extracted from *MovieLens*

We can observe from the table that, similar to what happened with the similar patterns, larger context vectors obtain better results as they provide more accurate representations. Comparing the quality of the top-1, top-5 and all representative transactions we can see that the quality decreases when we take into consideration more representative transactions, which is once again a good indicator that our ranking system is working properly, as the representative transactions highly ranked present a greater correspondence between the context and the ratings. We also find the same relationship between size of the context vector and size of T_α that we observed during the evaluation of the similar patterns, where the larger the size of the context vector, the less number of representative transactions that are found.

CHAPTER 6

Conclusions and future work

Frequent pattern mining is one of the most popular research fields and has received lots of attention during the last twenty years. Many researchers have challenged this problem, resulting in over one hundred algorithms. This has led to faster algorithms able to mine more types of data, but with a limited utility since the output of these algorithms needs to be processed by humans and these algorithms can output huge sets of frequent patterns.

The solution we present in this Master thesis utilizes domain knowledge stored in structured datasets to automatically annotate frequent patterns. The domain knowledge is used to build the contexts of the patterns using the natural representation of semantic relationships that graphs provide. By exploiting the contexts of the patterns via different graph inference techniques, we have been able to build semantic annotations for the patterns containing information about their meaning, usage, and other similar patterns. In doing so, we are taking a first step towards the understanding of the patterns. One step that can be extremely difficult and time-consuming for humans.

The semantic annotation of a frequent pattern also provides the means to summarize the output of frequent pattern mining algorithms as a byproduct. Given that our semantic annotation includes a process for computing the similarity between patterns, we can use this similarity to form clusters of patterns with the same meaning. This can be used to compact the set of frequent patterns, which is another step towards helping humans understand and interpret the output of frequent pattern mining algorithms.

We have used our approach on two different datasets, *MovieLens* and *eTourism*, in order to test the applicability of our method to different types of datasets and, in both cases, have been able to successfully obtain the semantic annotations. Nonetheless, we have found the information provided by the semantic annotation of the patterns of *MovieLens* more interesting. We think this is due to a richer knowledge data source, given that the number of concepts found in *MovieLens* is orders of magnitude higher than in *eTourism*. This is an indication that our semantic annotation method is limited by the available domain knowledge, which is understandable. But, even with the very limited knowledge provided by the *eTourism* ontology, the information presented by the semantic annotation still showcased some interesting descriptions of the patterns and relationships.

In order to evaluate our approach we have analyzed each feature of the semantic annotation separately. For the context indicators we have compared the different centrality measures, finally obtaining the most meaningful results with *betweenness* centrality. The similar patterns and representative transactions, on the other hand, have been numerically evaluated achieving results that reveal high quality in both cases. This quantitative analysis have also served to test the impact of the context vector size in the quality of the found similar patterns and representative transactions.

Looking forward, we would like to further explore the possibilities of this method by implementing more sophisticated graph structures, like multigraphs and temporal graphs, during the modeling of the context. We think this could be used to build better contexts for the patterns and even extend the applicability of this method to other types of patterns, like sequential patterns. We would also like to test the performance of our method on more domains and we are specially interested in testing it on retail stores datasets.

Future Work

Regarding the practical application of the results, we envision that semantically enriched data will be very useful in *recommendation systems*. Particularly, we think that a context-based approach will be able to solve some of the most prominent problems in recommendation systems, like *cold start*, by finding similarities at a higher level of abstraction. Furthermore, our context modeling method is able to seamlessly integrate data of different types, blurring the frontiers between collaborative, content-based, and demographic recommendation systems.

We also expect our approach to be extensible to other research problems. An example of this is *plan recognition*, a subproblem of intelligent planning that aims to discover the goals behind a sequence of actions. As our method is able to uncover hidden relationships in the patterns, we expect it to perform similarly in finding the causality of a sequence of observed events.

All things considered, our method poses a distinguishable approach to the more common statistical approaches while achieving remarkable results, and sets the basis for an interesting research line based on enriched data with wide applicability.

Bibliography

[1] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993.*, pages 207–216, 1993.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pages 487–499, 1994.

[3] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proceedings of the Eleventh International Conference on Data Engineering, March 6-10, 1995, Taipei, Taiwan*, pages 3–14, 1995.

[4] G. Atluri, K. Padmanabhan, G. Fang, M. Steinbach, J. R. Petrella, K. Lim, A. M. III, N. F. Samatova, P. M. Doraiswamy, and V. Kumar. Complex biomarker discovery in neuroimaging data: Finding a needle in a haystack. *NeuroImage: Clinical*, 3:123 – 131, 2013.

[5] A. Barrat, M. Barthelemy, R. Pastor-Satorras, and A. Vespignani. The architecture of complex weighted networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(11):3747–3752, 2004.

[6] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[7] U. Brandes. A faster algorithm for betweenness centrality. *Journal of mathematical sociology*, 25(2):163–177, 2001.

[8] H. Bunke and G. Allermann. Inexact graph matching for structural pattern recognition. *Pattern Recognition Letters*, 1(4):245–253, 1983.

[9] L. Dehaspe, H. Toivonen, and R. D. King. Finding frequent substructures in chemical compounds. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98), New York City, New York, USA, August 27-31, 1998*, pages 30–36, 1998.

[10] P. Fournier-Viger, T. Gueniche, and V. S. Tseng. Using partially-ordered sequential rules to generate more accurate sequence prediction. In *Advanced Data Mining and Applications, 8th International Conference, ADMA 2012, Nanjing, China, December 15-18, 2012. Proceedings*, pages 431–442, 2012.

[11] P. Fournier-Viger, J. C.-W. Lin, R. U. Kiran, Y. S. Koh, and R. Thomas. A survey of sequential pattern mining. *Data Science and Pattern Recognition*, 1(1), 2017.

[12] L. C. Freeman. Centrality in social networks conceptual clarification. *Social networks*, 1(3):215–239, 1978.

[13] I. Garcia, S. Pajares, L. Sebastia, and E. Onaindia. Preference elicitation techniques for group recommender systems. *Information Sciences*, 189:155–175, 2012.

[14] J. Han, H. Cheng, D. Xin, and X. Yan. Frequent pattern mining: current status and future directions. *Data Min. Knowl. Discov.*, 15(1):55–86, 2007.

[15] J. Han, J. Pei, Y. Yin, and R. Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Min. Knowl. Discov.*, 8(1):53–87, 2004.

[16] C. Jiang, F. Coenen, and M. Zito. A survey of frequent subgraph mining algorithms. *Knowledge Eng. Review*, 28(1):75–105, 2013.

[17] Q. Mei, D. Xin, H. Cheng, J. Han, and C. Zhai. Generating semantic annotations for frequent patterns with context analysis. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 337–346. ACM, 2006.

[18] M. E. Newman. Scientific collaboration networks. ii. shortest paths, weighted networks, and centrality. *Physical review E*, 64(1):016132, 2001.

[19] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M. Hsu. Mining sequential patterns by pattern-growth: The prefixspan approach. *IEEE Trans. Knowl. Data Eng.*, 16(11):1424–1440, 2004.

[20] D. C. Schmidt and L. E. Druffel. A fast backtracking algorithm to test directed graphs for isomorphism using distance matrices. *J. ACM*, 23(3):433–445, 1976.

[21] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *Advances in Database Technology - EDBT'96, 5th International Conference on Extending Database Technology, Avignon, France, March 25-29, 1996, Proceedings*, pages 3–17, 1996.

[22] J. R. Ullmann. An algorithm for subgraph isomorphism. *J. ACM*, 23(1):31–42, 1976.

[23] J. Wang, J. Han, and C. Li. Frequent closed sequence mining without candidate maintenance. *IEEE Trans. Knowl. Data Eng.*, 19(8):1042–1056, 2007.

[24] M. J. Zaki. Scalable algorithms for association mining. *IEEE Trans. Knowl. Data Eng.*, 12(3):372–390, 2000.

[25] M. J. Zaki. SPADE: an efficient algorithm for mining frequent sequences. *Machine Learning*, 42(1/2):31–60, 2001.