*Dedico a concretização desta etapa, finalizada por esta dissertação aos meus Pais, Mário Luiz e Maria Edite…*

*“Always look on the bright side of life!”*

*Monty Python*

Contents

[1 Introduction 1](#_Toc393020761)

[1.1 Motivation 2](#_Toc393020762)

[1.2 Vision 2](#_Toc393020763)

[1.3 Goals 3](#_Toc393020764)

[1.4 Development context 3](#_Toc393020765)

[1.5 Dissertation Structure 3](#_Toc393020766)

[2 State of the Art / Related Work 5](#_Toc393020767)

[2.1 Concept Relation Quantification 5](#_Toc393020768)

[2.2 Ontology Learning 5](#_Toc393020769)

[2.2.1 Association rules 5](#_Toc393020770)

[2.3 Building & Construction 5](#_Toc393020771)

[3 Theoretical and Technical Foundation 7](#_Toc393020772)

[3.1 Association Rules 7](#_Toc393020773)

[3.1.1 Frequent Pattern Growth 7](#_Toc393020774)

[3.1.2 The (Association Rules) Algorithm 13](#_Toc393020775)

[3.1.3 (Association Rules) Metrics 13](#_Toc393020776)

[3.2 Vector Space Model 14](#_Toc393020777)

[3.2.1 Term Weighting – The TF-IDF 15](#_Toc393020778)

[3.3 Similarity Measure in Information Retrieval 16](#_Toc393020779)

[3.3.1 Cosine Similarity Measure Algorithm 18](#_Toc393020780)

[4 Building & Construction Domain Ontology 20](#_Toc393020781)

[4.1 Ontology 20](#_Toc393020782)

[4.1.1 Construction Methodology 21](#_Toc393020783)

[4.1.2 Ontologic enrichment Dynamics 21](#_Toc393020784)

[4.1.3 The E-COGNOS Project – Ontology in Building and Construction 21](#_Toc393020785)

[5 Design and Implementation 22](#_Toc393020786)

[5.1 Tools and Technologies 22](#_Toc393020787)

[5.2 Conceptual & Technical Architectures 23](#_Toc393020788)

[5.2.1 Document Analysis 23](#_Toc393020789)

[5.2.2 FP-Growth 25](#_Toc393020790)

[5.2.3 Association Rules 25](#_Toc393020791)

[5.2.4 Frequent Itemset Mapping 25](#_Toc393020792)

[5.2.5 Ontology Enrichment 26](#_Toc393020793)

[5.3 Front end 26](#_Toc393020794)

[6 Evaluation 28](#_Toc393020795)

[6.1 Use cases 28](#_Toc393020796)

[6.2 Cientific publications 28](#_Toc393020797)

[7 Conclusion and Future Work 30](#_Toc393020798)

[8 Bibliography 32](#_Toc393020799)

[9 Appendices 34](#_Toc393020800)

[Figure 3.1 - An FP-Tree example for the items in the transaction Table 3‑1. 10](file:///F:\Os%20meus%20documentos\Universidade\Dissertação\Escrita\working\inprocess\Dissertação%20v4%20(Tudo).docx#_Toc393020752)

[Figure 3.2 - The three initial trees at the end of the first three transactions 11](file:///F:\Os%20meus%20documentos\Universidade\Dissertação\Escrita\working\inprocess\Dissertação%20v4%20(Tudo).docx#_Toc393020753)

[Figure 3.3 – Prefix sub-paths for all frequent items 12](file:///F:\Os%20meus%20documentos\Universidade\Dissertação\Escrita\working\inprocess\Dissertação%20v4%20(Tudo).docx#_Toc393020754)

[Figure 5.1 - Rapidminer Main Process 22](#_Toc393020755)

[Figure 5.2 - Conceptual Architecture 22](#_Toc393020756)

[Figure 5.3 - Vector Creation 23](#_Toc393020757)

[Figure 5.4 - Document Analysis Pipeline 24](#_Toc393020758)

[Figure 5.5 - Entity Relation Model 26](#_Toc393020759)

[Table 3‑1 – Transaction table for frequent items in database 9](#_Toc392953758)

[Table 5‑1 – Numerical to Binomial regulation 23](#_Toc392953759)

[Table 5‑2 - Association Rules Database Structure 24](#_Toc392953760)

SYMBOLS AND NOTATION

|  |  |  |
| --- | --- | --- |
|  | **AI** | **A**rtificial **I**ntelligence |
|  | **AR** | **A**ssociation **R**ules |
|  | **FP** | **F**requent **P**attern |
|  | **KD** | **K**nowledge **D**iscovery |
|  | **TID** | **T**ransaction **ID**entification |
|  | **UML** | **U**nified **M**odelling **L**anguage |
|  | **XML** | e**X**tensible **M**arkup **L**anguage |
|  | **FI** | **F**requent **I**tem |
|  |  |  |
|  |  |  |
|  |  |  |

# Introduction

In the modern world nowadays, information value is huge. The amount available to everyone is such as the ease that it reaches to every person. The global and exponential growth use of computers by people and businesses, and the rise of the internet changed the way to look to this information and the forms to collect, select and distribute. Every day, in most human daily tasks related to a computer, information records are created. It is thus more important to arrange techniques to represent such information in the form of knowledge representation understandable for the machine and human at the same time. The discovery of these techniques presents a big challenge to engineers. They have to know that the knowledge must be in a way that is searchable, with quick access to minimize the time of the access, making this task almost transparent to users. If there is a way to discover the knowledge desired, objectively and efficient, more time would be available to other tasks as or more important to everyday life.

Today, is almost impossible to live without information and their several representations, if it is a simple daily task or a simple supermarket visit, or even a more complex situations like constructing a building, the information is always present. In any situation also time is very important, so it becomes fundamental for engineers to create means to reduce the access time to knowledge. Through models that respond to this needs, like text and data mining models.

But to achieve an objective result of a search, the knowledge must have some kind of organization. It is not enough to have the knowledge as it is received as raw material. It will be necessary to process it in any way. Like separate it by themes, or measure its similarity to a central subject, but the important is to make some processing to help achieve the right results faster.

Através da extracção de conhecimento, a matéria prima passa por três processos: extracção de palavras, filtragem de expressões regulares e criação do vector estatístico.

A criação de uma ontologia aparece da necessidade de detecção, extracção e relacionamento dos conceitos das diferentes áreas, através de uma forma de classificação. Estas ontologias normalmente são criadas e mantidas de uma forma pela acção de um humano. Reconhecer um conceito, saber se já existe na ontologia, definir um relacionamento, atribuir uma classificação são as tarefas de um responsável pela manutenção de uma ontologia. Isto apresenta-se como um trabalho muito exaustivo, quanto maior for a base de conhecimento ou a ontologia. Algumas questões surgem de imediato. Como reconhecer um conceito, como verificar se ele existe na base de dados, como saber qual a relação que o conceitos tem com outros conceitos, como saber se se relaciona mais com um conceito A do que com um conceito B, como definir uma métrica, como saber se a relação existente é a adequada. As dúvidas que surgem são muitas na cabeça do engenheiro de ontologias?

Uma das motivações deste estudo é a criação de algum dinamismo uma ontologia, no meu entendimento, aparece

Esta é uma das motivações deste trabalho. Ajudar a que uma pesquisa seja o mais objectiva possível para que se possa descobrir rapidamente o conhecimento que deseja e deixar mais tempo para o que é mesmo importante.

Torna-se assim importante falar nas técnicas de web mining, mais concretamente Text mining, e knowlege discovery.

Antes de saber como se descobrir novo conhecimento é necessário saber como é que os sistemas de informação reconhecem o conhecimento quando este está na forma de texto. Existem algumas formas de representação do conhecimento, uma delas chama-se ontologia.

## Motivation

Information is everywhere. Nowadays, every area have a database or repository with information. As the IT systems grow, and the time passes the information also grows, and the complexity of the information sometimes reach sizes that humans do not imagine, neither can deal with them. Besides knowing the human brain is a “machine” that can store lots of knowledge inside, there is no one that have all the information in the world.

With the appearance of Internet and computers, arise the opportunity to store knowledge and share it with others, making the human more aware of the world around. One can be, for instance in Australia, and get an information of Portugal without travelling to the country.

Storing the information makes new challenges for the engineers. With the help of the improving of technology, and the massification of knowledge, the issue of storing information get to a point where was necessary to organize it.

In the competitive engineering world, a good organized system could be a key to reach success. The need of getting objective results from a search may be the difference in making a contract. Each day, engineers work with lots of information in their systems. The importance to have good systems, and to reach the information needed quickly grows.

(Enquadramento do problema actualmente. Falar na resolução de problemas pelas TI, e nomeadamente a resolução de problemas de pesquisa, organização e recolha de informação)

## Vision

The issue of information retrieval in a society where the organization, and indexing of information itself is very useful, and even sometimes it is critical, it becomes important to develop systems and processes that eases the complication and challenges that information has.

Organizing the information in databases is one of the steps for these challenges. Organizing in a way that information systems can easily retrieve, trying to discover relevant information, related to the search pretended.

(Será este projecto uma futura solução para automatização de processos? Em que contextos irá ajudar? “Eu vejo o mundo …” ou “Eu vejo no futuro que possa existir … ”.)

## Goals

The information in the databases of the systems, tend to get bigger after time. It is very important to organize such information in a way that anyone can reach it objectively. One of the goals of this work is to purpose a technique so that the information is searchable and the results intended can be related with a query.

It is a way to help discover patterns inside the documents, and even discover related concepts in text.

One will try to help discover frequent patterns of text in a corpus of document files, and with the help of an ontology, will try to map with similar concepts, and also try to find some association of those concepts with the context of the knowledge.

(Que situações/problemas a solução proposta irá resolver? Qual o objectivo principal/secundário)

## Development context

The civil industry is no exception when the subject information appears. Like any other area, the quantity of information is growing in large scale. It is thus necessary that the ways to store knowledge, and in a form that can be understandable to an engineer, and to a machine. Text is a good form of representation of knowledge, that has search capabilities and also easy understandable.

When a civil engineer starts a project, normally works in a collaborative way with other actors, like constructors, employees, other engineers. It becomes a necessity to have a system that has all information gathered, and at same time can be scalable. This scalability also brings new challenges. How to get the information for a specific project when all projects are in the system? And if the necessity of searching documents arises? How to get the documents that are similar to the subject one search for?

This work tries to answer such questions, proposing a solution to help get the more approximate results of a query, with a technique called Association Rules.

(A solução no contexto da engenharia civil. Falar da integração dos )

## Dissertation Structure

(Estrutura da dissertação. Resumo de cada capítulo)

# State of the Art / Related Work

(Apresentar aqui alguns trabalhos desenvolvidos na área, como o antecessor deste(do Paulo); dar também uma perspectiva dos trabalhos existentes com ontologias, aprofundar alguns trabalhos que reforçam a aplicação de association Rules; e mais especificamente apresentar alguns trabalhos desenvolvidos na construção civil, dentro do tema)

## Concept Relation Quantification

(Falar com o Paulo)

## Ontology Learning

### Association rules

## Building & Construction

# Theoretical and Technical Foundation

## Association Rules

In semantic systems, there are several ways to reach the goals. One presented in this work and also much studied in the scientific world is the Association rules technique. The main goal of this technique is to help taking conclusions about the relations of the words in the texts and build a new ontology or improving an existing one without any knowledge of the taxonomy present in it. It only relies on the original documents. How can this situation be achieved? After the first preparation of the text transforming it to concepts, before the rules can be discovered, one big step has to be made. From the pre-processed corpus of data, one has to recognize frequent patterns in the concepts in it. For this step there are several algorithms that deal with this problem, ECLAT, Apriori and FP-Growth are the most known and studied. Apriori and FP-Growth are the most used of all three. This section is going to be based on the foundations of the Association Rules with the description of the algorithm used in this work, the FP-Growth. It will be presented a definition and an explanation of such utility. Then, the algorithm will be explained and some discussion will be made around it. It will be compared with the two main competitors presented earlier. What is the best? And the fastest? How does it work? What are the one who are the best for low data structures? And for big ones? In last section about FP-Growth it will be discussed its importance before the discovery of Association Rules.

In the following lines the algorithm to discover the rules used in this work will be discussed in more detail. Furthermore, also some discussion around it will be presented. How does this technique works and what is the method used? What is a rule? How can one define a rule? Subsequently is the metric discussion. How can a rule be measured? What metrics are known? What metric is the best? What makes a strong rule, and what are the metrics that are most used.

All this questions will be answered by the author of this work along with some discussion around some other studies in the field of Association Rules and FP-Growth.

### Frequent Pattern Growth

Before rules of association can be found, the database must be mined to see which of the items are frequent. There are several processes in the academic community for this purpose who, given a set of database transactions can search it and return all the frequent item above some kind of measure to prove that represents the frequency of each item. Apriori, Eclat, FP-Growth are the ones that are most used and discussed by researchers. (Han et al., 2004) is recognized as a first introduction of the FP-Growth approach. It compares FP-Growth with Apriori, one of the initial and most used processes.

There are some characteristics that an algorithm should have to be classified as a good one. Namely time performance, usability in large databases and small databases, scalability, etc. In the next sub sections, the arguments are in favour of FP-Growth, in which the author of the present work identifies, based on the research, as being the best for the present case.

#### FP-Growth - A Definition

FP-Growth stands for Frequent Pattern Growth, it represents an algorithm to discover frequent patterns in data and specifically used in text mining. This algorithm is currently one of the fastest ones to mine association rules. It can also be defined as the first step in the path of item recognition that appears frequently. These items are called Frequent Patterns, meaning some text that appears in the database of transactions and are considered frequent above some minimum threshold value. This value is chosen by the engineer handling this process. It is more or less an arbitrary choice based on try and error method. More work can be done in this step to improve results, such that an artificial intelligence process could find a method to dynamic choose the best value for the intended use of the algorithm.

Before FP-Growth the processes to discover frequent patterns in databases of text were mainly Apriori-like based algorithms. Such processes are known to be very costly in large databases. Its times to search will exponential grow as the database will also grow. On the other side stands FP-Growth, a *divide-and-conquer method* (Han et al., 2004). It is based in a prefix tree representation, called FP-Tree. This tree holds the frequent patterns found in the transaction database. With the divide and conquer method this can be seen as a recursive elimination process. It will separate the frequent items from the ones that are not frequent inside a database.

Due to the popularity, effectiveness and performance of this algorithm, it was much appreciated in many investigations in the academic and scientific community. Also many changes proposals to the original were studied and presented. For example, in (Wang et al., 2002), the author proposes improvements and upgrades for the algorithm. One of these proposals is the Top Down FP-Growth algorithm. This work’s author debates a different process to search frequent patterns. It searches the FP-Tree from the top to the bottom and not generating conditional FP-Trees to each item. This method processes the nodes of the tree at upper levels before processing the ones on the lower levels. This is different from original FP-Growth, in which it mines the tree from bottom up, from the item to its prefixes, and creating several conditional trees for each item.

(Korczak and Skrzypczak, 2012) illustrates an example of discovering customer frequent patterns in an online store with the help of FP-Growth to discover association rules between the transactions of the customers. In other project, (Bonchi and Goethals, 2004) are inspired by the known small Japanese bonsai tree and tries to apply its broad concept in the FP-Tree of the FP-Growth algorithm. This study examines the reduction of the tree by a technique that is based on pruning specific “leaves” (nodes) resulting in smaller compressed trees.

Another study related to this subject is the one presented in (Zeng et al., 2010). In this paper, the discussion presented is a process to weight association rules based on an FP-tree. It proposes a new method called FP-WAR, FP-Weighted Association Rules where outlines the importance of getting a technique to weight association rules and give them different *interestingness*.

#### FP-Growth Algorithm (Fp-Tree building, projecting and pruning)

The FP-Growth algorithm is made in two steps. The first is an elimination phase scheme and building of an FP-Tree, and the second step is a frequent itemset generation.

Table 3‑1 – Transaction table for frequent items in database

|  |  |
| --- | --- |
| TID | Items |
| 1 | architect, designer |
| 2 | designer, engineer, analyst |
| 3 | architect, engineer, analyst, professor |
| 4 | architect, analyst, professor |
| 5 | architect, designer, engineer |
| 6 | architect, designer, engineer, analyst |
| 7 | architect |
| 8 | architect, designer, engineer |
| 9 | architect, designer, analyst |
| 10 | designer, engineer, professor |

**Step 1 – Infrequent items elimination**

The elimination scheme is where the initial data is mined to separate the frequent from the non frequent items in the database. It uses a recursion process to make a kind of elimination scheme. It compresses the data set by determining the frequent items and deleting all that are not frequent. These infrequent elements are the ones that are found mainly alone in the database, this means, that such frequency is below the above threshold level. On the other side, the frequent items that are above this level are called frequent items, and are the ones that it will be kept, and are the base for the next step.

As already referred, the deleting process is based on a user-defined minimum offset, called support, in which below it, all items are eliminated, and above it the items are the ones to be stored. This value along others is explained in detail in sub section 3.1.3 below when the discussion will be around how to measure an item. The result of this process is a modified set of transactions with only frequent sets of single-items. In the end, the frequent items are ordered for better organization search purpose. Table 3‑1 presents an example of a resulting set of transactions after this elimination and ordering process be made. Each line of the table represents a transaction from the database. The resulting items are the ones found in the database that matches the requirements of a threshold value above the user defined value. These items are the ones considered frequent in our initial data.

Figure 3.1 - An FP-Tree example for the items in the transaction .

Following the elimination scheme and frequent itemset filtering, a tree has to be built, the so called FP-tree. This tree is a rooted acyclic graph with vertices not labelled and a root node null valued. It is constructed with transaction scans like the ones on Table 3‑1, one transaction at a time. The main idea is to map the graph such that a new path for each unique transaction will be drawn. Each node represents each frequent item found earlier. If the search discovers a common prefix on the item set, it will overlap and remove it, and if a suffix exists, creates a new node in the graph and connect to its previous item. If this common prefix is shared along two transactions, these are merged into the corresponding node. Each node holds a counter. This counter represents the frequency of the node in the respective path and always starts with the value of 1 in each node creation. When a transaction shares a node along the same path, the counter is then incremented, and it goes to the next transaction. This will ensure that each frequent item only needs one path for each item in the tree. The chances of which the common prefixes can be shared are higher if the frequent items have been sorted by its frequency, from top to bottom order.

Example: let’s suppose that there is a database that after applying the elimination and ordering task of FP-Growth presents the following set of items S, and the previous transaction Table 3‑1 with all 10 transactions of this database scan:

**Building the FP-Tree**

The algorithm to build the FP-tree will have to deal with each of the transactions, one at a time. First it should be created a node to represent a root with the value of *null*. For the first transaction TID1, the first item is *architect*. This item will create a new node in the tree with the value of *architect* and a counter associated to it with the initial value of 1. The second item in the TID1 is *designer*, as this is also a new one, a node should be created in the next level of *architect* node with counter equal to 1. Figure 3.2a) represents the situation after TID1 where one can see the FP-tree constructed until this moment. For TID2 the items {*designer, engineer, analyst*} are the ones to consider. As the first one is *designer* and in the current tree there is no first level node with such designation, a new node should be created a connected to *null* node for the item *designer*. As this is a new node its counter is set to 1. Then the node for engineer must be created and connected to designer with the counter at 1. The next item on the transaction is *analyst* that it is also a new one in the path, so a node must be created with its name and the counter equal to 1. The moment at the end of this transaction TID2 is represented in the Figure 3.2b). One can see that there is two individual paths for the transactions, both sharing a node with the same item: *designer*. In this case both should be linked to recognize this situation and further evaluation. The dotted line represents this linkage. TID3 includes items {*architect, engineer, analyst, professor*}. One can easily see that the first item of this transaction already is connected to the *null* node. In this case, there is no need to create a new node that would be repeated, instead the respective counter should be incremented by 1, totalling now 2, that represents the two paths that starts with *architect*. For *engineer*, *analyst* and *professor*, the procedure should be similar for new nodes in the same path. As *engineer* and *analyst* already exist in the tree, they should be linked with its equals also in the same way *designer* was above. An illustration of this transaction is Figure 3.2c). One can see that three different paths exists if counting the last leaves of the branches or if totalling the sum of the nodes and counters connected to *null* node.

Figure 3.2 - The three initial trees at the end of the first three transactions

**a) TID = 1**

**b) TID = 2**

**c) TID = 3**

This process shall continue through all transactions until all transactions table lines be evaluated for the actual process. For this example the table has 10 transactions, which are all represented in Figure 3.1 above. From this figure it can be seen all the paths from the transaction table. To be noted that node *architect* is the one that starts most of the paths of the itemsets, exactly 8, making him the most frequent item in the database. As we can also infer from the figure, is that *designer* is the second most frequent, but it alternates in its position in the paths, five in the first position and 5 in the second position. It can also be noted in the tree that all items nodes with the same designation in the different paths are linked to each others.

The main purpose of this step is a filtering and organizing step with the objective to facilitate the search of frequent items. With this kind of structure the speed of search will decrease significantly.

**Step 2 – Frequent Itemset Generation**

The next step in FP-Growth algorithm, as noted, is the evaluation and recognition of the most frequent items. It is an extraction process that is called **Frequent Itemset Generation**. The main idea of this technique is to extract frequent items from the earlier built FP-tree. From a Bottom up perspective, the technique will individually process each ending bottom node and separates in an exclusively separate tree which is called **prefix path sub-tree**.

Figure 3.3 – Prefix sub-paths for all frequent items

**a) Prefix path sub-tree for *professor***

**b) Prefix path sub-tree for *designer***

**c) Prefix path sub-tree for *engineer***

**d) Prefix path sub-tree for *architect***

**e) Prefix path sub-tree for *analyst***

Find an example of this trees in Figure 3.3 where each of the items in the database will have an exclusive prefix path sub-tree. It is the *divide and conquer* method, separating each sub-tree individually for faster performance of the main tree. The paths included in each sub-tree are the ones that have the respective item as a leave node (ending bottom node). Therefore, for a frequent item X and its ancestor Y and Z from an FP-tree, the resulting sub-tree will be used to extract itemsets ending in X, subsequently will extract the ones ending in YX, and after it the ones ZYX, continuing traversing through the path and being processed recursively until it analyses all paths and reaches the most top node, also called null node.

Table 3‑2 - Paths table for frequent items

|  |  |  |
| --- | --- | --- |
| Item | # | Paths |
| Architect | 1 | architect |
| Professor | 1 | architect, engineer, analyst, professor |
| 2 | architect, analyst, professor |
| 3 | designer, engineer, professor |
| Designer | 1 | architect, designer |
| 2 | designer |
| Engineer | 1 | architect, designer, engineer |
| 2 | architect, engineer |
| 3 | designer, engineer |
| Analyst | 1 | designer, engineer, analyst |
| 2 | architect, engineer, analyst |
| 3 | architect, analyst |
| 4 | architect, designer, engineer, analyst |
| 5 | architect, designer, analyst |

Using the previous example, a) to e) represents the prefix sub-path trees for itemset S. For each of the individual items in S, one sub-path tree was divided for further processing. In a), the prefix path sub-tree for item *professor* presents 3 paths described in . Consequently, the *divide and conquer* approach makes the problem easier to evaluate.

As observed in the first step, for a node be considered frequent it has to hold a support threshold value (minSup = 2 in previous example). That was a requirement to search the items in the database and eliminate the ones that did not have at least another equal item in it. With this minimum support in mind, one has to traverse from the bottom to find each frequent items. For this task, if one wants to know if X is a frequent item, it has to follow the dotted lines in the prefix sub-tree and sum the counters associated with item X, and thus calculating the support for X, denoted by (Vo and Le, 2009).

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

Equation (1) represents the mathematical expression to consider an item X a frequent item in the database. If this situation is true, X can be extracted as a frequent item and it can be found frequent items ending in X. For that a table as has to be considered holding all paths for each of the extracted frequent items.

In previous the example, considering the item *professor*, one can easily sees the 3 paths for it. They are {architect:8, engineer:1, analyst:1}. {architect:8, analyst:1} and {designer:2, engineer:2}, these all lead to an ending node *professor*. In case of the third path, the set {designer, engineer} appears twice in database, however with *professor*, they only appear once. Similarly, architect in the other two paths is shared among them with a support of , appearing only twice together with *professor*. Therefore, to evaluate the set that appears together with *professor*, the nodes from the corresponding prefix path sub tree have to be updated as follows:

S1={architect:1, engineer:1, analyst:1}, S2={architect:1, analyst:1} and S3={designer:1, engineer:1}. Furthermore, as *designer* has, it's not considered FI with *professor*. Thus S3={engineer:1}.

Figure 3.4 - Conditional FP-Tree for item *professor*

These three prefix paths of *professor* {S1, S2, S3} constitute the sub pattern-base called, according to (Han et al., 2004), conditional pattern base, consequently, an FP-Tree based in this is called **conditional FP-tree**. A visual representation of the conditional FP-Tree for item *professor* is illustrated in .

includes all frequent items found in this example, considering all items in database.

Table 3‑3 - Frequent itemsets discovered for all items

|  |  |
| --- | --- |
| Item | Frequent Itemsets discovered |
| professor | {professor}, {analyst, professor}, {architect, analyst, professor}, {engineer, professor}, {architect, professor} |
| analyst | {analyst}, {engineer, analyst}, {designer, engineer, analyst}, {architect, engineer, analyst}, {designer, analyst}, {architect, designer, analyst}, {architect, analyst}, |
| engineer | {engineer}, {designer, engineer}, {architect, designer, engineer}, {architect, engineer} |
| designer | {designer}, {architect, designer} |
| architect | {architect} |

#### FP-Growth vs APRIORI and ECLAT –

When the knowledge discovery in databases theme arises, FP-Growth is not the only method to find frequent patterns. There are some other approaches for its discovery in artificial intelligence. The most comparable to FP-Growth are ECLAT and APRIORI.

ECLAT means

APRIORI is probably the first effective process to discover frequent patterns.

One of the advantages of FP-Growth, when comparing with the competitors, is that it does not create huge amount of frequent itemsets and a small database of transactions.

#### FP-Growth before Association Rules

### The (Association Rules) Algorithm

The association rules are intended to reinforce frequent patterns in the text. The main responsibility is to give an extra push for the matching of concepts.

### (Association Rules) Metrics

The Association Rules algorithm recognizes association between frequent patterns resulting from a frequent pattern recognition algorithm, like FP-Growth, or Apriori. But how is its relevance measured? How can anyone know that a rule is better than another one? How can anyone simply know that the rule is valid? To answer these questions, the author describes some metrics that classify each rule that is found in the process of Association Rule discovery (Azevedo and Jorge, 2007).

#### Confidence

This metric represents an estimation of the probability of observation Concept B given Concept A. The result values are between [0,1]. When a rule is received, one can immediately classify the relationship of the corresponding concepts.

#### Lift

*Lift*, also known as *Interest* is a measure to indicate the independence of Concept A from Concept B. The result values are inside the interval [0,+∞[. This measure is only used for co-occurrence and not with measure implication. The higher the value, the higher the rule is interesting. (A *lifted* B). If a Concept A is far from 1 it means that there is some relevant information about Concept B. If the value is 1, they are both considered independent, without any relation.

#### Conviction

This is a measure to support confidence and lift metrics. Is used for implication, it matters the distance that it happens. A=>B ≠ B=>A. The value represents the value of implication, as the higher the value, the higher the relationship value between both concepts. As Lift, if the value is 1 the concepts are independent, meaning no relation. The values are included in [0,+∞[.

#### Support

*Support* or *frequency* is a statistic metric defined as “the support of a set of items, represents the percentage of transactions from database that contains such items”. As this is a statistic measure, the values are between [0..1]. Higher the value, the more frequent are the concepts.

#### Rule Interest

*Rule Interest*, *PS*, *Novelty* or *leverage*, is a measure obtained from the difference between the real support and the expected support. The values are inside of [-0.25..0.25]. If a rule receives the value 0, it means that Concepts are independent. As the value grows, it grows the rule significance and interest.

#### Laplace

A “confidence estimator”, Laplace is also a statistic measure indicating that when the support of a Concept A decreases, the relevance of such concept also decreases. The values are inside [0,1[

## Vector Space Model

In Text Mining, a Vector Space Model (VSM) is an Information Retrieval statistical model that tries to make the assumption that each document is represented by a point in space in a group of documents. As more near the points are, bigger is the semantic similarity and as more apart the points are, less representative in the semantics they are(Turney et al., 2010).

The VSM is a form to explain to computer systems the semantics of human language. It was created for the SMART information retrieval system by its developer Gerard Salton and his team. (Salton, 1971)

VSM has several properties, one of which is that given a corpus it will extract knowledge automatically. The majority of today search engines use VSM as a model because of its good performance in preparing the raw data to measure the similarity between documents, phrases and words (Manning et al., 2008). Queries made by the engines have a good performance presenting relevant results to the query author. Some of the most known algorithms for semantic relatedness(Pantel and Lin, 2002; Rapp, 2003; Turney et al., 2003) and semantic relation similarity (Lin and Pantel, 2001; Nakov and Hearst, 2008; Turney, 2006) also use VSMs as a base technology for preparation of the data.

There are some hypotheses that VSM tries to answer, they all begin from the main one, the *statistical semantic hypothesis*, that states that if statistical patterns are used on human word syntactic formation and usage of natural language terms, the possibility to understand the meaning of human speech is real.(Turney et al., 2010) The above hypothesis is the converging point of the following ones: bag of words, distributional, extended distributional and latent relation. In the following lines, the author of the present work will give a brief explanation of each of them.

* **Bag of words hypothesis:** By representing the documents on the corpus and the query as a bag (or collection) of words, one can estimate the relevance of these documents to a query. This can be explained as the word frequency that exists in the documents tends to represent the document relevance faced to a query. Each bag of words can be represented by a Term-Document Matrix. Salton et al. (Salton et al., 1975) wrote that this hypothesis is the foundation of VSM application in information retrieval systems; The authors of this hypothesis believes that each column vector of this matrix represents in some way a subject or meaning of the document. (Salton et al., 1975)
* **Distributional hypothesis:** When the subject is similar contexts, the distributional hypothesis says that the words in those contexts are also similar in their meanings. (Harris) The data is organized in a Word-Concept Matrix. When one wants to measure the word similarity, this hypothesis is the reason for the application of the VSM;
* **Extended distributional hypothesis:** This hypothesis was proposed by Lin and Pantel (Lin and Pantel, 2001), the co-occurence of patterns in similar pairs, will lead to similar meanings. The co-occorence of X:Y similar pairs is a tendency of patterns like “X solves Y” or “Y is solved by X”. When this happens one can be lead to think that these kind of patterns present the similar meanings; (Lin and Pantel, 2001) The representation of this pairs results in a pair-pattern matrix.
* **Latent relation hypothesis:** The last hypothesis is the inverse of the extended distributional hypothesis described before. It covers the pairs of words, when these co-occur in similar patterns. In this case one can say they have similar meaning.(Turney, 2006)

### Term Weighting – The TF-IDF

When dealing with large raw data, these can be represented by vectors in a matrix, the Term-Document Matrix. This matrix is prepared in such way that the terms are arranged in row vectors and the documents are arranged in the column vectors. Each entry in the matrix corresponds to a weight of each term in a document. This process orders the terms in a document by their relevance in each document and in corpus by a tf-idf (term frequency – inverse document frequency) weighting normalized scheme. This scheme is presented by the following:

(1)

– represents the term frequency of term i in document j.

– represents the number of documents that contains term i.

The result is the matrix (2) with the weight or relevance of each term.

(2)

By this weighting process, the system may know the relevance of each term in the context and which one is more or less representative.

This form of representation is called a bag or multiset, and supports the bag of words hypothesis discussed earlier. This way, one can discover the tendency of the proximity of a document to a subject, by this frequency of words in the document. Salton et al (Salton et al., 1975) states that this hypothesis is the foundation of VSM application in Information Retrieval.(Salton et al., 1975)

The VSM is not the only way to represent text, as seen the several hypothesis represent more ways to represent and weight relevance of the terms. But this is not exclusively, for example, proposes another form of weighting the terms and documents in a corpora. This point of view, which is called TWEAK, uses labels to learn the terms weight related to its importance in the subject as a parametric function, where the model parameters are learned from the labelled data. (Yih, 2009). Instead of being an independent weighting scheme like tf-idf, that it does not take consideration the previous analysis or other kind of past similarity calculus, this TWEAK is dependant of the previous analysis as this considers the model parameters in the evaluation. Meaning that the previous labelling and classifying of the terms in the corpus are included in the next weighting, making this process influenced for the actual subject of the text data.

## Similarity Measure in Information Retrieval

(A importância da Similarity Measure na área de Information Retrieval, talvez falar de algumas medidas que existam. A Semelhança entre o quê? Documentos e queries! Word similarity; Document similarity; Context similarity)

Before one can understand what to measure, it is important to understand what similarity is, and what is it role in Information Retrieval (IR). Similarity is the state or the fact of being similar.(Oxford University, 2006). To understand what Similarity means, it is important to understand that each word or concept has a(many) meaning(s)/subject(s) that can be related to. How similar is each word to a subject? Lin even presents a Similarity Theorem to explain it:

“*The similarity between A and B is measured by the ratio between the amount of information needed to state the commonality of A and B and the information needed to fully describe what A and B are."* (Lin, 1998)

In other words, a similarity of two words is a quantification of their differences and similarities. It is a measure of how much their meanings are close. How much information do they share, and how much information they do not.

Even in words with similar meaning, similarity measures are important to find what the word that best fits in a particular context is. Or just to know what the word that is more similar to other is.

In the previous chapter, the author presented an approach on how the raw data can be organized for further evaluation. For it there are some ways to do it, several take vector models as starting point. The VSM is one of the best known and applied method. As VSM states, to better understand the meaning of a word or concept, there must be a measure in the semantic relations of each word, given a set of documents. One must know how similar each word is, therefore it can try to figure out what is the best approximation to the main subject of each document, and thus give the possibility for the machine understand its meaning. For example, given the words *Architect*, *Engineer* and *house*, how can they be related to each other? The reader will obviously know that an *Architect* is more related to a *house* than is an *Engineer*. For the reader is easy to know the meaning but for an AI system, is it? How can it understand the relation to each word? This is called a *Similarity Measure*, on other words, it is a way to measure the semantic relation of each of the words or concepts, and see if they are more related to ones, than others. Several paths are possible to consider and achieve the previous.

Similarity measure can be used for more than similarity of words and concepts. One of the most known applications of similarity is between a user query and some document pages like a web search engine (“Google.com,” 2013) But this is not an exclusively use of similarities, also for instance, one can measure a document similarity between scientific papers, or measure similarity of unstructured data texts so a context or domain can be found. It can be used to aid ontology construction, fortify relations between concepts, make ontology dynamic and capable of learning.

To start a similarity measurement some initial thoughts must be made. What is it going to be measured? The corpora documents? Words? Vectors? The type of the initial corpora is important. Other thoughts must be made on if the domain is known, is the data already structured? Is there any ontology to support the process or any previous similarity measurement to help? Are their semantic similarities important? What is the similarity measure process to use?

There are much similarity measures in the scientific community that are being discussed.

One of the most known is the Cosine Similarity Algorithm.

### Cosine Similarity Measure Algorithm

# Building & Construction Domain Ontology

(Falar das ontologias, o que são, para que servem, como funcionam.

Falar da importância das ontologias.

Falar da importância das ontologias no domínio da construção civil, e especificamente na construção de edifícios.)

## Ontology

The term “Ontology” origins from early 18th century, from the modern Latin *ontologia*, from the Greek work *onto*, which means “being”, plus *logia* which means study, science, theory. (Oxford University, 2006). It is, thus, a sub domain of philosophy that studies the being, to be. In its origin, is the study of the nature of being, existence and the way this being relates with others. Also represents the categories of the being and such relations.

In information systems, the ontology is the study of the representation of knowledge. It is a form of definition and organization of the knowledge domain. Ontology can be defined as a structure or a form of data organization. For a better understanding of this knowledge, an ontology is divided in *classes*. These classes present *relations* between them. Each class includes a set of *concepts* and their relations.

The ontology is this way, a structured representation of a data set of knowledge that aims for a better understanding of the data presented.

In IT systems, W3C created a standard to define a representation and organization of a domain that can be “easily” read by a human. This standard is named OWL, which means Web Ontology Language. (W3C, 2004)

One of the main challenges that a set of raw text data presents is the difficult to translate it to a language that computer might better understand and even further process. This challenge is somewhat solved with the OWL. This language appeared to facilitate the interface between a raw data, unstructured that is not easily read and a structured data that human eye could make conclusions about it.

### Construction Methodology

#### Types of Ontologies.

#### OWL Ontology Language (and RDF)

### Ontologic enrichment Dynamics

#### Update relations

#### Creation of new concepts and relations

#### Relation Weighting

### The E-COGNOS Project – Ontology in Building and Construction

# Design and Implementation

## Tools and Technologies

This work proposes a solution following the structure presented in Figure 5.1.

Each block represents a sub-process and was developed with a specific technology. ‘Document Analysis’, ‘FP-Growth’ and ‘Association Rules’ blocks were developed with Rapidminer processes. After this processing the results were delivered to the application by a rapidminer interface (Figure 5.1), coded in Ruby Programming Language technology.



Figure 5.1 - Rapidminer Main Process

The databases created were developed in MySQL and implemented in MySQL Workbench, a specific tool to develop databases. Both run on top of an Apache server.

For the communication with the ontology is made with Jena Semantic Framework Ontology, a Java API that supports OWL language, in which the ontology is coded.

The Front End application is developed in NetBeans environment, and coded in Java Language with ASP features. Some of the technologies used can be found in Appendix X.

Protegè software was also used to aid the author to refer to the ontology structure.

ASSOCIATION

RULES

FREQ. ITEMSET MAPPING

ONTOLOGY

ASSOCIATION RULES DATABASE

DOCUMENT ANALYSIS

FP-GROWTH

Figure 5.2 - Conceptual Architecture

## Conceptual & Technical Architectures

The idea of this work is to implement a solution to help discover ontologic relations between concepts. Specifically, find Rules of Association for the concepts. The approach proposed lies on a conceptual architecture composed by several blocks, each one representing a process (see Figure 5.2). Namely ‘Document Analysis’, ‘FP-Growth’, ‘Association Rules’, ‘Frequent Itemset Mapping’ and the blocks representing two support databases ‘Ontology’ and ‘Association Rules Database’. In the following lines, each block will be explained in detail. This architecture is, in the author opinion, the one that best suits the problem.

### Document Analysis

Before one can discover the earlier ontologic relations, the source text documents must be prepared in such way, so that can be understandable by this architecture. Some organization in them is necessary, along with some processing as the text is in a raw state. To achieve the former, the documents pass by some pre-processing in rapidminer (see Figure 5.3).



Figure 5.3 - Vector Creation

The first step of the preparation of the files is Tokenization. The Tokenization operation is responsible for separate the full text into a sequence of tokens. Tokens can be understood in several ways, for the purpose of this work, one can consider token as a set of letters. Everything that does not have letters, it is discarded.

The second step is the transformation of all tokens to lower case, and the third step is the filtering of Stopwords. These stopwords are the words that have no semantic importance for the context, like “*the*”, “*each*”, “*a*”, etc. All stopwords are removed of the set of tokens.

After this, the next step is the Stemming algorithm. The stemming algorithm has the responsibility to transform the word in its stem, aka common morphological root. In this project the stemming algorithm used is the Snowball algorithm. [] This process can be optional, but one thinks it is of a great value, as it reduces the words to its stem, gathering the words from the same family to enrich its value in a document. Meaning that as more words are grouped for it stem, more representative is the stem in the document, and thus, reinforcing a better context in it. Also it reduces the size of the data, augmenting the precision of each stem.

The following step is to discard all tokens (words) that are lower than 4 and higher than 50 characters. This process is necessary to remove unnecessary tokens that have no taxonomic relevance for the study, like chain of random letters, thus the author chose this interval as a fair number.

The last step of the document analysis is the generation of n-grams. The n-grams generation is the creation of sequences of 1 to N words, being for this case N=3, using unigrams, bigrams (eg. Waste Management) and trigrams (e.g. Electric Power Product). The purpose for this generation is a first try to find concepts and groups of words that represents concepts.

DOCUMENT ANALYSIS

Tokenize

Transform Cases

(lower cases)

Filter

Stopwords

Stemming

(Snowball)

Filter Tokens

(4-50)

Generate

n-grams (n≤3)

Figure 5.4 - Document Analysis Pipeline

The output of the analysis is saved into a temporary database for easement of processes. In the interface between the analysis and the FP-growth process, the results enters afterwards in a sub-auxiliary process Numerical to Binomial, whose function is to change the nominal values of the vector to binomial values, which changes to false every value inside an interval, and to true the ones outside. This means that words that have no significant ontologic meaning are filtered out of the document corpus.

Table 5‑1 – Numerical to Binomial regulation

|  |  |  |  |
| --- | --- | --- | --- |
| **NumBinMax** | **Support** | **Confidence** | **Association Rules** |
| 0,012 | 0,25 | 0,01 | 18 |
| 0,012 | 0,25 | 0,60 | 18 |
| 0,012 | 0,25 | 0,70 | 18 |
| 0,013 | 0,20 | 0,01 | 102 |
| 0,013 | 0,25 | 0,70 | 12 |
| 0,014 | 0,20 | 0,01 | 92 |
| 0,015 | 0,20 | 0,01 | 92 |

For the purpose of this work, the interval values were choose as follows: Min – 0.0 Max 0.013, as the shows, some tests were made to get the configuration of this module that gives a wider number of Association Rules to examine.

### FP-Growth

This block is responsible to find Frequent Patterns from the source documents that the author is analysing. The process uses the FP Growth algorithm, described earlier in this document. This process is the base for the recognition of concepts that appear in the text more frequently than others.

### Association Rules

(Mostrar alguns exemplos de aplicação das Association Rules no trabalho;)

Table 5‑2 - Association Rules Database Structure

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| # | Premise | Conclusion | Confidence | Conviction | Gain | Laplace | Lift. | Ps | Total Support |
| 1 | Concept A | Concept B | Val A | Val B | Val C | Val D | Val E | Val F | Val G |

### Frequent Itemset Mapping

(Explicar o que representa este bloco; Falar de onde é que aparece este procedimento, e baseado em quê; Explicar o que são os elementos que se apresentam em baixo)

The Frequent Itemset Mapping is a module that is executed after the rapidminer processing. The main objective is to map the concepts reaching of the association rules results with the concepts directly from the respectively ontology.

The mapping is processed by the similarity level calculus between frequent itemsets and ontology equivalent terms.

#### Frequent Itemsets

Frequent Itemsets are concepts that are found to be a frequent pattern from the source documents. This is a direct result from the FP-Growth algorithm.

#### Ontology equivalent terms

#### Similarity

#### Candidate concepts

(Concepts that are related to each other by the keywords, in the ontology)

#### Mapping

(Transforming each frequent itemset in a concept that can be used in the ontology)

This process is responsible to map each frequent itemset from the source documents with the keywords that are associated with the concepts in the ontology.

### Ontology Enrichment

(Falar do processo de enriquecimento da ontologia, com a adição de novos conceitos, actualização dos já existentes; Como detectamos, o que actualizamos, etc)

(Falar o OWL como ferramenta de ajuda para visualização da ontologia)

## Front end

(Arranjar um nome para a aplicação)

(Falar da ligação entre Rapidminer->FrontEND<-JENA<-Owl(XML) )

(Inserir algumas imagens e explicar o funcionamento da aplicação através de diagramas, talvez Use Cases e/ou sequence diadrams[UML])



Figure 5.5 - Entity Relation Model

# Evaluation

(Falar da avaliação da aplicabilidade e importância do trabalho;)

## Use cases

## Cientific publications

(Falar da aprovação de publicações deste trabalho pela comunidade científica)

Incluir papers (e talvez incluir referência à ligação com o trabalho do Ruben e do Paulo)

# Conclusion and Future Work

# Bibliography

Azevedo, P.J., Jorge, A.M., 2007. Comparing Rule Measures for Predictive Association Rules, in: Kok, J.N., Koronacki, J., Mantaras, R.L. de, Matwin, S., Mladenič, D., Skowron, A. (Eds.), Machine Learning: ECML 2007, Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 510–517.

Bonchi, F., Goethals, B., 2004. FP-Bonsai: The Art of Growing and Pruning Small FP-Trees, in: Dai, H., Srikant, R., Zhang, C. (Eds.), Advances in Knowledge Discovery and Data Mining, Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 155–160.

Google.com [WWW Document], 2013. URL https://www.google.pt/?gfe\_rd=cr&ei=EwS7U-L4Mcbe8gf90oFg (accessed 7.7.14).

Han, J., Pei, J., Yin, Y., Mao, R., 2004. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. Data Min. Knowl. Discov. 8, 53–87.

Korczak, J., Skrzypczak, P., 2012. FP-Growth in Discovery of Customer Patterns, in: Aberer, K., Damiani, E., Dillon, T. (Eds.), Data-Driven Process Discovery and Analysis, Lecture Notes in Business Information Processing. Springer Berlin Heidelberg, pp. 120–133.

Lin, D., 1998. An information-theoretic definition of similarity., in: ICML. pp. 296–304.

Lin, D., Pantel, P., 2001. DIRT@ SBT@ discovery of inference rules from text, in: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, pp. 323–328.

Manning, C.D., Raghavan, P., Schütze, H., 2008. Introduction to information retrieval. Cambridge university press Cambridge.

Nakov, P., Hearst, M.A., 2008. Solving Relational Similarity Problems Using the Web as a Corpus., in: ACL. Citeseer, pp. 452–460.

Oxford University, 2006. Oxford Dictionary of English. Oxford University Press, London.

Pantel, P., Lin, D., 2002. Discovering word senses from text, in: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, pp. 613–619.

Rapp, R., 2003. Word sense discovery based on sense descriptor dissimilarity, in: Proceedings of the Ninth Machine Translation Summit. pp. 315–322.

Salton, G., 1971. The SMART retrieval system—experiments in automatic document processing.

Salton, G., Wong, A., Yang, C.-S., 1975. A vector space model for automatic indexing. Commun. ACM 18, 613–620.

Turney, P., Littman, M.L., Bigham, J., Shnayder, V., 2003. Combining independent modules to solve multiple-choice synonym and analogy problems.

Turney, P.D., 2006. Similarity of Semantic Relations. Comput. Linguist. 32, 379–416. doi:10.1162/coli.2006.32.3.379

Turney, P.D., Pantel, P., others, 2010. From frequency to meaning: Vector space models of semantics. J. Artif. Intell. Res. 37, 141–188.

Vo, B., Le, B., 2009. Mining traditional association rules using frequent itemsets lattice, in: International Conference on Computers Industrial Engineering, 2009. CIE 2009. Presented at the International Conference on Computers Industrial Engineering, 2009. CIE 2009, pp. 1401–1406. doi:10.1109/ICCIE.2009.5223866

W3C, 2004. OWL Web Ontology Language Overview [WWW Document]. OWL Web Ontol. Lang. URL http://www.w3.org/TR/2004/REC-owl-features-20040210/ (accessed 7.7.14).

Wang, K., Tang, L., Han, J., Liu, J., 2002. Top down FP-Growth for association rule mining. Springer.

Yih, W., 2009. Learning term-weighting functions for similarity measures, in: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2. Association for Computational Linguistics, pp. 793–802.

Zeng, B., Jiang, X.-L., Zhao, W., Luo, C., 2010. The improvement of weighted association rules arithmetic based on FP-tree, in: Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on. IEEE, pp. V4–549.

# Appendices