Automated Extraction of Product Comparison Matrices From Informal Product Descriptions

Sana Ben Nasr[a], Guillaume Bécan[a], Mathieu Acher[a], João Bosco Ferreira Filho[b], Nicolas Sannier[c], Benoit Baudry[a], Jean-Marc Davril[d]

[a] *Inria / IRISA, University of Rennes 1, France*
[b] *Department of Computer Science of the Federal University of Ceara, Brazil*
[c] *SNT Center for Security, Reliability and Trust, University of Luxembourg, Luxembourg*
[d] *University of Namur, FUNDP, Faculty of Computer Science, Belgium*

Abstract

Domain analysts, product managers, or customers aim to capture the important features and differences among a set of related products. A case-by-case reviewing of each product description is a laborious and time-consuming task that fails to deliver a condense view of a family of product.

In this article, we investigate the use of automated techniques for synthesizing a product comparison matrix (PCM) from a set of product descriptions written in natural language. We describe a tool-supported process, based on term recognition, information extraction, clustering, and similarities, capable of identifying and organizing features and values in a PCM – despite the informality and absence of structure in the textual descriptions of products.

We evaluate our proposal against numerous categories of products mined from BestBuy. Our empirical results show that the synthesized PCMs exhibit numerous quantitative, comparable information that can potentially complement or even refine technical descriptions of products. The user study shows that our automatic approach is capable of extracting a significant portion of correct features and correct values. This approach has been implemented in *MatrixMiner* a web environment with an interactive support for automatically synthesizing PCMs from informal product descriptions. *MatrixMiner* also maintains traceability with the original descriptions and the technical specifications for further refinement or maintenance by users.

1. Introduction

Domain analysis is a crucial activity that aims to identify and organize features that are common or vary within a domain [1, 2, 3]. A feature can be roughly defined as a prominent or distinctive user-visible aspect, quality or characteristic of a product [4]. At their respective level, domain experts, product managers, or even customers on their daily life activities need to capture and understand the important features and differences among a set of related products [5]. For instance, the motivation for a customer is to choose the product that will exhibit adequate characteristics and support features of interest; when several product candidates are identified, she or he will compare and eventually select the "best" product. In an organization, the identification of important features may help to determine business competitive advantage of some products as they hold specific features.

Manually analyzing a set of related products is notoriously hard [6, 7, 3]. As the information is scattered all along textual descriptions, written in informal natural language, and represent a significant amount of data to collect, review, compare and formalize; a case-by-case review of each product description is labour-intensive, time-consuming, and quickly becomes impractical as the number of considered products grows.

Given a set of textual product descriptions, we propose in this article an approach to automatically synthesize *product comparison matrices (PCMs)*. PCMs are tabular data describing products along different features [8]. Our approach extracts and organizes information despite the lack of consistent and systematic structure for product descriptions and the absence of constraints in the writing of these descriptions, expressed in natural language.

Numerous organisations (e.g., Wikipedia), companies, or individuals rely on tabular representation to present some discriminant features of a product compared to another [8, 9]. With the extraction of PCMs, organizations or individuals can obtain a synthetic, structured, and reusable model for the understanding and the comparison of products. Instead of reading and confronting the information product by product, PCMs offer a *product line view* to practitioners. It is then immediate to identify recurrent features of a domain, to understand the specific characteristics of a given product, or to locate the features supported and unsupported by some products. PCMs are also an interesting potential step stone for further analysis such as: (1) formalization and generation of other domain models (e.g., feature models [10, 7, 11, 12, 13]), (2) feature recommendation [6], (3) automatic reasoning (e.g., multi-objective optimizations) [14], (4) derivation of automatic comparators and/or configurators [9].

Numerous techniques have been developed to mine variability [10, 15, 16] and support domain analysis [17, 18, 19, 20, 7, 6, 3, 21, 22, 23], but none of them address the problem of structuring the information in a PCM.

Our automated approach relies on Natural Language Processing (NLP) and mining techniques to extract PCMs from text. We adopt an *contrastive analysis* technology to identify *domain-specific terms* (single and multi-word) from the textual descriptions. The proposed method takes the descriptions of the different products as input, and identifies the linguistic expressions in the documents that can be considered as terms. In this context, a term is defined as a conceptually independent expression. Then, the method automatically identifies which terms are actually domain-specific. We also rely on information extraction to detect *numerical information*, defined as domain relevant multi-word phrases containing numerical values. The task of building the PCM involves computing terms (resp., information) similarity, terms (resp., information) clustering, and finally features and cell values extraction.

This approach has been implemented in a tool, *MatrixMiner*: It is a web environment with an interactive support for automatically synthesizing PCMs from informal product descriptions [24]. *MatrixMiner* also maintains traceability with the original descriptions and the technical specifications for further refinement or maintenance by users. This article is a significant extension of our ESEC/FSE tool demonstration, 4-pages paper [24]. We provide an in-depth description of the automated extraction process as well as substantial empirical results, including a user study with MatrixMiner. We rely on previous work in which we have defined a rich and expressive *format* capable of formally encoding PCMs [8].

We evaluate our tool against numerous categories of products mined from BestBuy [25], a popular American company that sells hundreds of consumer electronics. Overall, our empirical study shows that, given a supervised and necessary scoping (selection of products), the synthesized PCMs exhibit numerous quantitative and comparable information: 12.5% of quantified features, 15.6% of descriptive features, and only 13.0% of empty cells. The user study shows that our automatic approach retrieve 43% of correct features and 68% of correct cell values in one step and without any user intervention. We also show that we have as much or more information in the synthesized PCMs than in the technical specifications for a significant portion of features (56%) and cell values (71%).

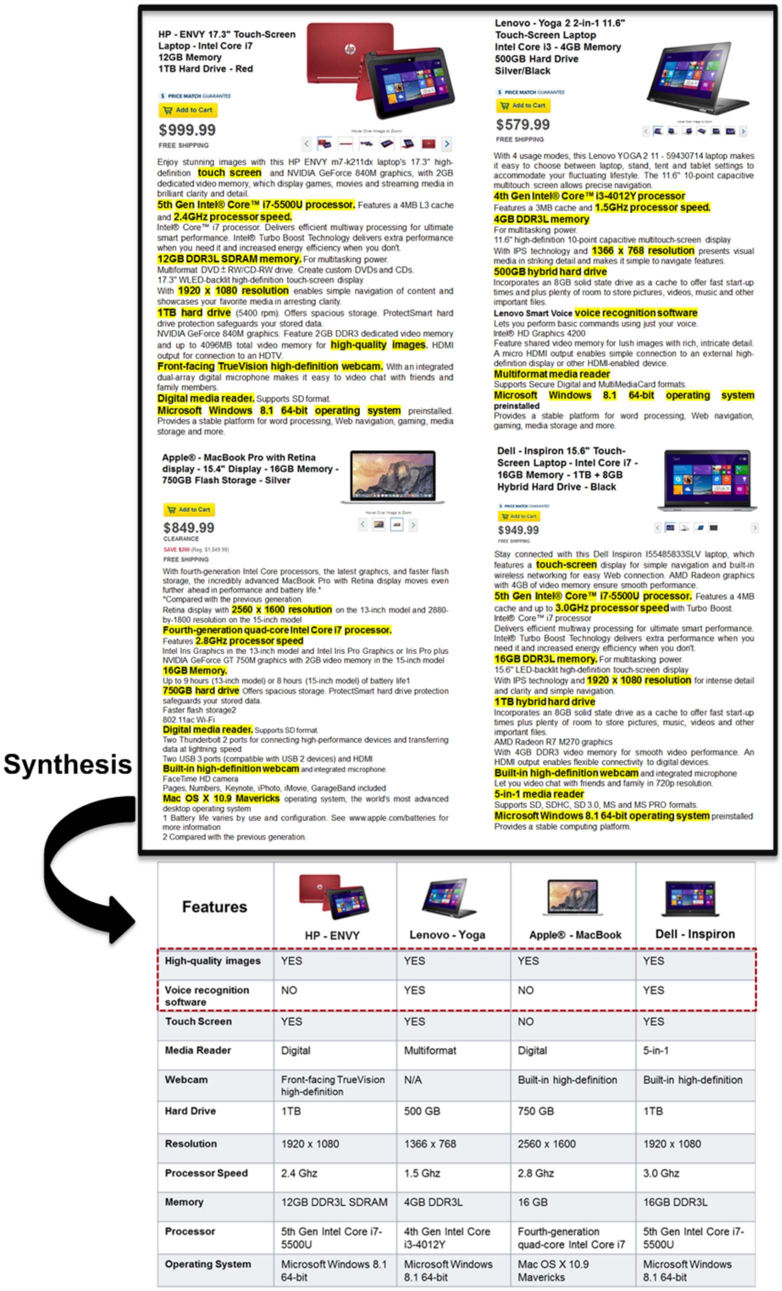
The remainder of the article is structured as follows. In Section 2, we provide additional background on PCMs and elaborate on the PCM synthesis challenge. Section 3 gives a general overview of our approach. Sections 4 and 5 describe the main steps of our approach, namely terms and information extraction, and subsequent construction of the PCM. In Section 6, we describe and illustrate the integration of the synthesis techniques into the *MatrixMiner* environment. Sections 7 to 9 successively present our case study and analyse the results of an empirical evaluation and a user study. In Section 10, we discuss threats to internal and external validity while in Section 11 we point out the differences and synergies between existing works and our proposal. Section 12 concludes the article and presents future research directions.

2. Background and Motivation

Organizations describe the products they sell on their website using different categories of text forms. It goes from plain text in a single paragraph, formatted text with bullets, to matrices with product specifications. There is a spectrum of product descriptions ranging from structured data (matrices) to informal descriptions written in natural languages. Both have strengths, weaknesses, and have the potential to comprehensively describe a set of products. BestBuy provides descriptions for hundreds of thousands of products, including: (1) *products overviews*, texts describing features of products using natural language (see Figure 1); (2) *technical specifications*, which describe the technical characteristics of products through feature lists (see Figure 2).

Figure 1 illustrates the common scenario in which a customer needs to buy a laptop on BestBuy website and has to decide among a diversity of products. He/she has to go through many textual descriptions (product overviews) and reasons over the different features of the product. A typical question is to figure out if a particular feature is supported by existing products (if any) and what are the alternatives. In domain analysis, the biggest challenge is related to the number of products and the number of features an analyst has to gather and organize. The more assets and products, the harder the analysis. Our goal is to automate the manual task of analyzing each product with respect to its textual description and clustering information over several products, and provides a reader with an accurate and synthetic *product comparison matrix (PCM)*, as shown in Figure 1.

The manual elaboration of a PCM from textual overviews can be done as follows. First, it requires the ability to detect from the text the potentially relevant domain concepts expressed as single or multi words including domain specific terms and numerical information, such as those that are highlighted in the text of Figure 1. Once detected, multiwords have to be split between the feature name and its value. We observed different value types for features in a previous work [26]. Each of these value types imply a different interpretation for the feature. For instance, the feature "Touch Screen" means the availability of the feature, which has to be interpreted as as a YES/NO value (see the PCM of Figure 1). Feature values can also mix letters and numbers, for instance the following snippet: "5th Gen Intel Core i7-5500U". Consequently, determining features and their related values is not a trivial problem.



HP - ENVY 17.3" Touch-Screen Laptop - Intel Core i7 12GB Memory 1TB Hard Drive - Red

PRICE MATCH GUARANTEE

Add to Cart

$999.99

FREE SHIPPING

Hover Over Image to Zoom

Enjoy stunning images with this HP ENVY m7-k211dx laptop's 17.3" high-definition touch screen and NVIDIA GeForce 840M graphics, with 2GB dedicated video memory, which display games, movies and streaming media in brilliant clarity and detail.
5th Gen Intel® Core™ i7-5500U processor. Features a 4MB L3 cache and 2.4GHz processor speed.
Intel® Core™ i7 processor. Delivers efficient multiway processing for ultimate smart performance. Intel® Turbo Boost Technology delivers extra performance when you need it and increased energy efficiency when you don't.
12GB DDR3L SDRAM memory. For multitasking power.
Multiformat DVD ± RW/CD-RW drive. Create custom DVDs and CDs.
17.3" WLED-backlit high-definition touch-screen display
With 1920 x 1080 resolution enables simple navigation of content and showcases your favorite media in arresting clarity.
1TB hard drive (5400 rpm). Offers spacious storage. ProtectSmart hard drive protection safeguards your stored data.
NVIDIA GeForce 840M graphics. Feature 2GB DDR3 dedicated video memory and up to 4096MB total video memory for high-quality images. HDMI output for connection to an HDTV.
Front-facing TrueVision high-definition webcam. With an integrated dual-array digital microphone makes it easy to video chat with friends and family members.
Digital media reader. Supports SD format.
Microsoft Windows 8.1 64-bit operating system preinstalled. Provides a stable platform for word processing, Web navigation, gaming, media storage and more.

Apple® - MacBook Pro with Retina display - 15.4" Display - 16GB Memory - 750GB Flash Storage - Silver

Add to Cart

$849.99

CLEARANCE

SAVE $200 (Reg. $1,049.99)

FREE SHIPPING

Hover Over Image to Zoom

With fourth-generation Intel Core processors, the latest graphics, and faster flash storage, the incredibly advanced MacBook Pro with Retina display moves even further ahead in performance and battery life.*
*Compared with the previous generation.
Retina display with 2560 x 1600 resolution on the 13-inch model and 2880-by-1800 resolution on the 15-inch model
Fourth-generation quad-core Intel Core i7 processor.
Features 2.8GHz processor speed
Intel Iris Graphics in the 13-inch model and Intel Iris Pro Graphics or Iris Pro plus NVIDIA GeForce GT 750M graphics with 2GB video memory in the 15-inch model
16GB Memory.
Up to 9 hours (13-inch model) or 8 hours (15-inch model) of battery life1
750GB hard drive Offers spacious storage. ProtectSmart hard drive protection safeguards your stored data.
Faster flash storage2
802.11ac Wi-Fi
Digital media reader. Supports SD format.
Two Thunderbolt 2 ports for connecting high-performance devices and transferring data at lightning speed
Two USB 3 ports (compatible with USB 2 devices) and HDMI
Built-in high-definition webcam and integrated microphone.
FaceTime HD camera
Pages, Numbers, Keynote, iPhoto, iMovie, GarageBand included
Mac OS X 10.9 Mavericks operating system, the world's most advanced desktop operating system
1 Battery life varies by use and configuration. See www.apple.com/batteries for more information
2 Compared with the previous generation.

Lenovo - Yoga 2 2-in-1 11.6" Touch-Screen Laptop Intel Core i3 - 4GB Memory 500GB Hard Drive Silver/Black

PRICE MATCH GUARANTEE

Add to Cart

$579.99

FREE SHIPPING

Hover Over Image to Zoom

With 4 usage modes, this Lenovo YOGA 2 11 - 59430714 laptop makes it easy to choose between laptop, stand, tent and tablet settings to accommodate your fluctuating lifestyle. The 11.6" 10-point capacitive multitouch screen allows precise navigation.
4th Gen Intel® Core™ i3-4012Y processor
Features a 3MB cache and 1.5GHz processor speed.
4GB DDR3L memory
For multitasking power.
11.6" high-definition 10-point capacitive multitouch-screen display
With IPS technology and 1366 x 768 resolution presents visual media in striking detail and makes it simple to navigate features.
500GB hybrid hard drive
Incorporates an 8GB solid state drive as a cache to offer fast start-up times and plus plenty of room to store pictures, videos, music and other important files.
Lenovo Smart Voice voice recognition software
Lets you perform basic commands using just your voice.
Intel® HD Graphics 4200
Feature shared video memory for lush images with rich, intricate detail. A micro HDMI output enables simple connection to an external high-definition display or other HDMI-enabled device.
Multiformat media reader
Supports Secure Digital and MultiMediaCard formats.
Microsoft Windows 8.1 64-bit operating system preinstalled
Provides a stable platform for word processing, Web navigation, gaming, media storage and more.

Dell - Inspiron 15.6" Touch-Screen Laptop - Intel Core i7 - 16GB Memory - 1TB + 8GB Hybrid Hard Drive - Black

PRICE MATCH GUARANTEE

Add to Cart

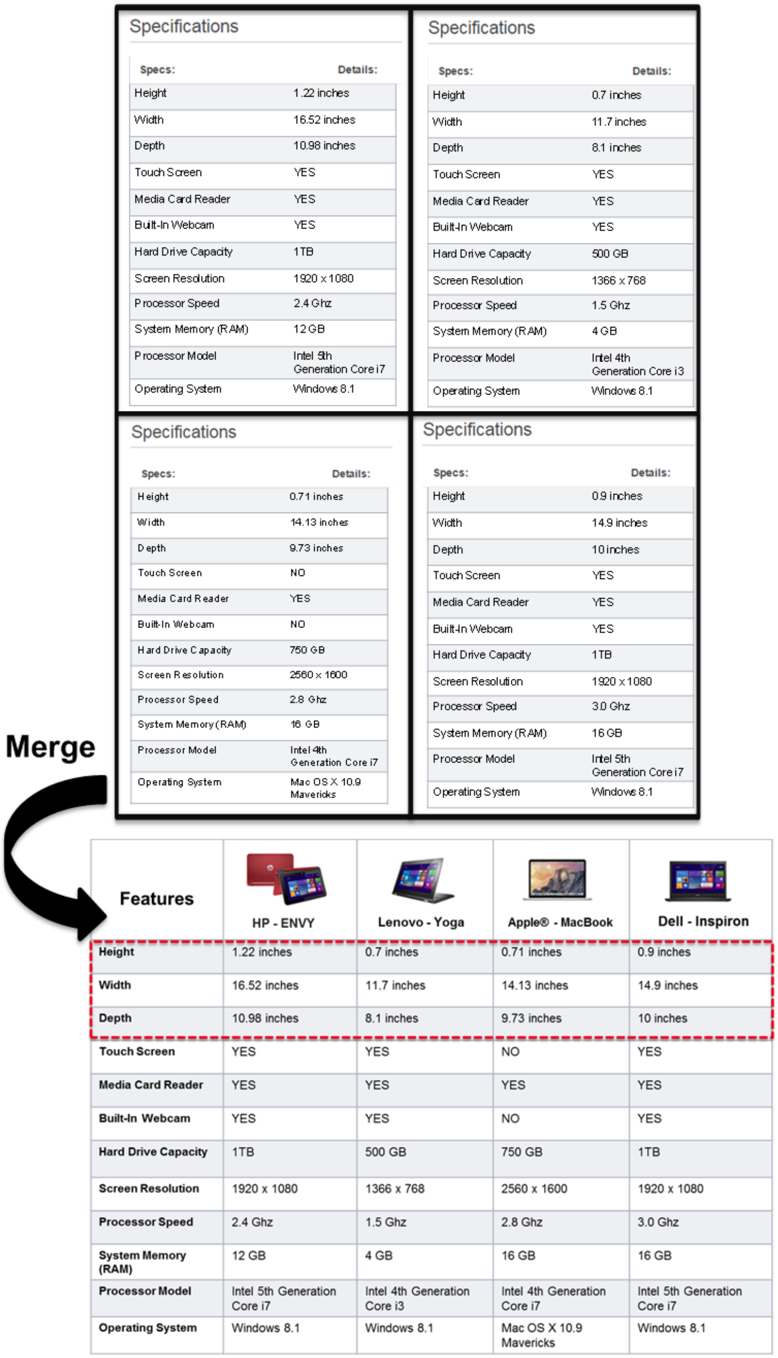
$949.99

FREE SHIPPING

Stay connected with this Dell Inspiron I55485833SLV laptop, which features a touch-screen display for simple navigation and built-in wireless networking for easy Web connection. AMD Radeon graphics with 4GB of video memory ensure smooth performance.
5th Gen Intel® Core™ i7-5500U processor. Features a 4MB cache and up to 3.0GHz processor speed with Turbo Boost.
Intel® Core™ i7 processor
Delivers efficient multiway processing for ultimate smart performance. Intel® Turbo Boost Technology delivers extra performance when you need it and increased energy efficiency when you don't.
16GB DDR3L memory. For multitasking power.
15.6" LED-backlit high-definition touch-screen display
With IPS technology and 1920 x 1080 resolution for intense detail and clarity and simple navigation.
1TB hybrid hard drive
Incorporates an 8GB solid state drive as a cache to offer fast start-up times plus plenty of room to store pictures, music, videos and other important files.
AMD Radeon R7 M270 graphics
With 4GB DDR3 video memory for smooth video performance. An HDMI output enables flexible connectivity to digital devices.
Built-in high-definition webcam and integrated microphone
Let you video chat with friends and family in 720p resolution.
5-in-1 media reader
Supports SD, SDHC, SD 3.0, MS and MS PRO formats.
Microsoft Windows 8.1 64-bit operating system preinstalled
Provides a stable computing platform.

Features	HP - ENVY	Lenovo - Yoga	Apple® - MacBook	Dell - Inspiron
High-quality images	YES	YES	YES	YES
Voice recognition software	NO	YES	NO	YES
Touch Screen	YES	YES	NO	YES
Media Reader	Digital	Multiformat	Digital	5-in-1
Webcam	Front-facing TrueVision high-definition	N/A	Built-in high-definition	Built-in high-definition
Hard Drive	1TB	500 GB	750 GB	1TB
Resolution	1920 x 1080	1366 x 768	2560 x 1600	1920 x 1080
Processor Speed	2.4 Ghz	1.5 Ghz	2.8 Ghz	3.0 Ghz
Memory	12GB DDR3L SDRAM	4GB DDR3L	16 GB	16GB DDR3L
Processor	5th Gen Intel Core i7-5500U	4th Gen Intel Core i3-4012Y	Fourth-generation quad-core Intel Core i7	5th Gen Intel Core i7-5500U
Operating System	Microsoft Windows 8.1 64-bit	Microsoft Windows 8.1 64-bit	Mac OS X 10.9 Mavericks	Microsoft Windows 8.1 64-bit

Figure 1: Automatic synthesis of a PCM from 4 textual product descriptions. Portions of texts in yellow have been identified and exploited to synthesize features' names and cell values. High-quality images and voice recognition (in red) are features not described in the technical specifications of the same 4 products (see Figure 2).



Specifications

Specs:	Details:
Height	1.22 inches
Width	16.52 inches
Depth	10.98 inches
Touch Screen	YES
Media Card Reader	YES
Built-In Webcam	YES
Hard Drive Capacity	1TB
Screen Resolution	1920 x 1080
Processor Speed	2.4 Ghz
System Memory (RAM)	12 GB
Processor Model	Intel 5th Generation Core i7
Operating System	Windows 8.1

Specifications

Specs:	Details:
Height	0.7 inches
Width	11.7 inches
Depth	8.1 inches
Touch Screen	YES
Media Card Reader	YES
Built-In Webcam	YES
Hard Drive Capacity	500 GB
Screen Resolution	1366 x 768
Processor Speed	1.5 Ghz
System Memory (RAM)	4 GB
Processor Model	Intel 4th Generation Core i3
Operating System	Windows 8.1

Specifications

Specs:	Details:
Height	0.71 inches
Width	14.13 inches
Depth	9.73 inches
Touch Screen	NO
Media Card Reader	YES
Built-In Webcam	NO
Hard Drive Capacity	750 GB
Screen Resolution	2560 x 1600
Processor Speed	2.8 Ghz
System Memory (RAM)	16 GB
Processor Model	Intel 4th Generation Core i7
Operating System	Mac OS X 10.9 Mavericks

Specifications

Specs:	Details:
Height	0.9 inches
Width	14.9 inches
Depth	10 inches
Touch Screen	YES
Media Card Reader	YES
Built-In Webcam	YES
Hard Drive Capacity	1TB
Screen Resolution	1920 x 1080
Processor Speed	3.0 Ghz
System Memory (RAM)	16 GB
Processor Model	Intel 5th Generation Core i7
Operating System	Windows 8.1

Features	HP - ENVY	Lenovo - Yoga	Apple® - MacBook	Dell - Inspiron
Height	1.22 inches	0.7 inches	0.71 inches	0.9 inches
Width	16.52 inches	11.7 inches	14.13 inches	14.9 inches
Depth	10.98 inches	8.1 inches	9.73 inches	10 inches
Touch Screen	YES	YES	NO	YES
Media Card Reader	YES	YES	YES	YES
Built-In Webcam	YES	YES	NO	YES
Hard Drive Capacity	1TB	500 GB	750 GB	1TB
Screen Resolution	1920 x 1080	1366 x 768	2560 x 1600	1920 x 1080
Processor Speed	2.4 Ghz	1.5 Ghz	2.8 Ghz	3.0 Ghz
System Memory (RAM)	12 GB	4 GB	16 GB	16 GB
Processor Model	Intel 5th Generation Core i7	Intel 4th Generation Core i3	Intel 4th Generation Core i7	Intel 5th Generation Core i7
Operating System	Windows 8.1	Windows 8.1	Mac OS X 10.9 Mavericks	Windows 8.1

Figure 2: Basic aggregation of 4 technical specifications into a PCM. Height, width, and depth (red) are features not described in the textual descriptions of the 4 products; the other features overlap and are also contained in synthesized PCM

2.1. Toward Automatic Extraction of PCMs

Our objective is to automate the identification of features, their values, and collect information from each product to create a complete PCM. This comes with a set of challenges, mostly due to the informal and unstructured nature of textual overviews.

First, the representation aims to provide a *structured* view of all available products and all available features. From a parsing and natural language processing perspective, plain text and PCMs have different organizations schemes. On the one hand, text is grammatically organized but may not been organized in terms of feature definitions nor description. As being part of open initiatives such as consumer associations, mainstream initiatives like Wikipedia, or e-commerce websites, one cannot rely on the quality of the textual descriptions, in terms of both wording and organization. For instance, textual descriptions may present features in different orders as to put emphasis on a particular one, or may have different authors that do not share the same writing patterns. On the other hand, a PCM is clearly organized as a set of products, features, and associated values. If a product description provides for free the product's name, it is not trivial to determine its features and their values, which have to be mined from the description, as stated previously.

Second, it is not only a matter of parsing products features and their respective values. It is also a matter of making the most *synthetic* and relevant PCM to enable *comparison*. The number of features depends on both (1) the textual description length, precision, and quality, and (2) the capability to cluster features as they share the same meaning but different names. Finding the right name for a feature can have an impact on the number of features. Being generic (for instance, "processor") increases the possibility to have different values for this feature whereas a series of too specific features ("5th Gen Intel... processor") will only have a YES/NO value with a high risk of feature explosion. Ideally we would rather like to extract a feature (*e.g.* processor) together with a value (*e.g.* 5th Gen Intel...) out of an informal text.

2.2. Complementarity between Product Overviews and Technical Specifications

Another interesting observation is the nature of relationship that can exist between product overviews and product specifications. Again, with the same example, but now considering technical specifications, we automatically compute the output PCM (see Figure 2).

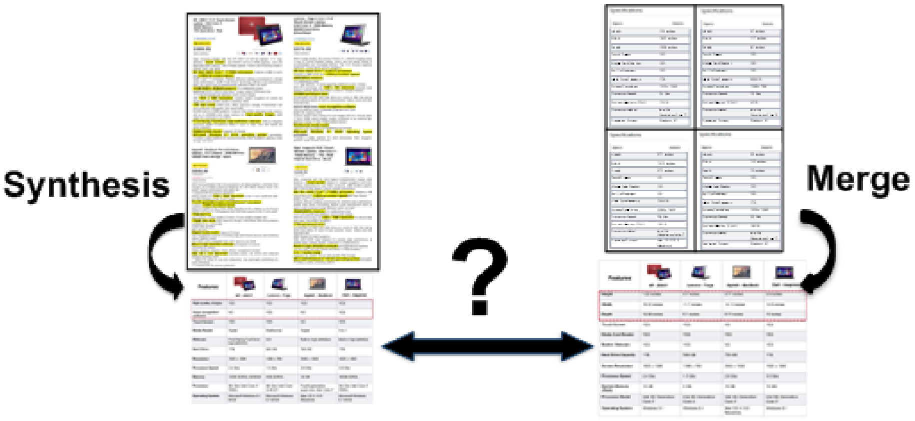


Figure 3: Complementarity between synthesized PCMs (textual overviews, left-hand side) and technical specifications (right-hand side)

With our automated extraction from overviews, there is also a potential to complement or even refine technical specifications of products (see the two PCMs in Figure 1 and Figure 2). Considering the verbosity aspect of natural language, the overview can contain information that refines the information of the specification. If we compare the cell values of the same feature or two equivalent features in the overview and the specification, we observed that the cell value in the overview PCM can refine the cell value in specification PCM.

For example, "Media Reader" exists in both overview PCM and specification PCM of laptops. In the first case, it has "Digital", "Multiformat", "5–in–1" as possible values, while in the second case, it is simply a boolean feature. "Webcam" is also boolean in specification PCM and non boolean in overview PCM ("Front–facing TrueVision..." and "Built–in high–definition"). In the specification PCM, "Memory" has "12 GB" as a possible value, while in the overview PCM, the value contains also the type of memory: "12GB DDR3L SDRAM". At the same time, "Operating System" has "Windows 8.1" as a possible value in the specification PCM, however it includes also the architecture in the overview PCM ("Microsoft Windows 8.1 64-bit").

Furthermore, in an overview PCM, we can obtain additional features that could refine features existing in specification PCM. For instance, "High–quality images" and "Voice Recognition Software" are two features in the overview PCM. However, they do not exist in the specification PCM. Hence, overviews can also complement the information of technical specifications.

Similarly, the specification PCM can contain information that refine those in the overview PCM. For instance, the following features "height", "width” and "depth” in the specification PCM refine "size" in the overview PCM. The mapping of features can be one-to-one or arbitrarily complex. Overall the user can get a complete view through the PCM and the aggregation of information coming from both overviews and specifications.

3. Overview of the Automatic Extraction

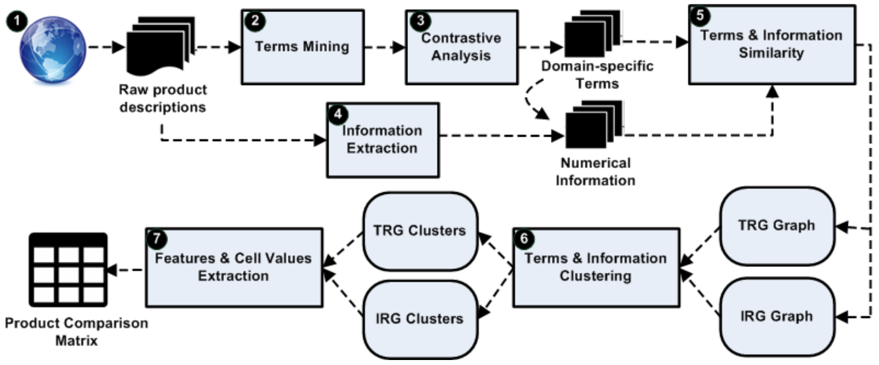


Figure 4: Approach Overview

Our approach is summarized in Figure 4 and consists of two primary phases. In the first phase, *domain specific terms* and *numerical information* are extracted from a set of informal product descriptions (steps ❶ to ❹), while in the second phase the PCM is constructed (steps ❺ to ❼). For step ❶, the raw product descriptions are extracted along different categories of products. We provide means to either (1) manually select the products to be included in the comparison; or (2) group together closest products within a category.

We outline in the following the rest of the procedure.

Mining Domain Specific Terms. Steps ❷ and ❸ are based on a natural language processing approach, named *contrastive analysis* [27], for mining *domain specific terms* from textual documents. In this context, a *term* is a conceptually independent expression (either single or multi-word). A multi-word is conceptually independent if it occurs in different context (*i.e.* it is normally accompanied with different words). For instance, "Multiformat Media Reader" is a term, while "Reader" is not a term, since in the textual product descriptions considered in our study it often appears coupled with the same word (*i.e.* "Media"). Combining single and compound words is essential to detect features and their values.

The purpose of the *contrastive analysis* method is to find out the terms which are *specific* for the domain of the document under study [27, 28]. Basically, contrastive analysis confronts the terms mined from domain-specific documents (here: informal product descriptions) and those retrieved from domain–generic documents (e.g., newspapers). If a term from the domain-specific document is not frequent in the domain-generic documents, it is a domain-specific term. Otherwise, it is a domain-generic term.

Information Extraction. Step ❹ aims at mining *numerical information* since they are capable to describe precisely the technical characteristics of a product. These information are domain relevant multi-word phrases which contain measures (*e.g.* "1920 x 1080 Resolution") including intervals (*e.g.* "Turbo Boost up to 3.1 GHz").

Inspired by the "termhood" concept used earlier, the extracted multi-words should be conceptually independent from the context in which they appear. For instance, suppose we have in the text this phrase "the processor has 3 MB cache and 2.0 GHz processor speed". Here, "2.0 GHz Processor Speed" is conceptually independent whereas "2.0 GHz Processor" is not. We use statistical filters inspired by the "termhood" metric applied in step ❷, to extract these numerical domain relevant multi-words from text.

Building the PCM. Once the top list for the terms and respectively for numerical information are identified for each product, we start the construction of the PCM. This process requires creating some intermediate structures. The key idea is to perform separately terms clustering from information clustering. A terms cluster gives the possible descriptor values (*e.g.* "Multiformat") while an information cluster provides the potential quantifier values (*e.g.* "1920 x 1080") for the retrieved feature. In step ❺ we compute similarity between terms and correspondingly between information to generate two weighted similarity relationship graphs: a Terms Relationship Graph (TRG) and an Information Relationship Graph (IRG). To identify coherent clusters, we first determine the similarity of each pair of elements by using syntactical heuristics. In step ❻ we apply clustering in each graph to identify terms clusters and information clusters. Finally, step ❼ extracts features and cell values to build the PCM. Elements which are not clustered will be considered as boolean features. We distinguish different types of features (see Figure 1): *boolean* which have Yes/No values, *quantified* when their values contain measures (*e.g.* "Resolution", "Hard Drive", *etc.*), *descriptive* if their values contain only noun and adjectival phrases (*e.g.* "Media Reader"). The resulting PCM can be visualized and refined afterwards. In the following sections, we elaborate these three main tasks. We address mining terms and information in Section 4 and the construction of the PCM in Section 5.

4. Terms & Information Extraction

In this section, we describe the first half of our approach which handle the terms and information extraction from textual descriptions. Several successful tools have been developed to automatically extract (simple or complex) terms [29, 27]. The reason why we develop our own terms extractor is that we propose later an extraction of numerical information inspired by the termhood concept. This section includes mining domain specific terms (steps ❶ to ❸) in Sections 4.1 and 4.2, and information extraction (step ❹) in Section 4.3.

4.1. Terms Mining

Terms mining consists in the first two steps of Figure 4. Firstly, raw feature descriptors are mined from each product overview via the BestBuy API. The product overview tend to include a general product description followed by a list of feature descriptors. Therefore, given n products of the same category, we have $D_1 ... D_n$ documents (products overviews). From each one of these documents we identify a ranked list of terms. In this section, we discuss the candidate extraction process, that makes use of: i) linguistic filters; ii) stoplist; iii) statistical filters (C-NC Value).

4.1.1. Linguistic filters.

The linguistic filters operate on the automatic Part–of–speech (POS) tagged and lemmatized text, making use of various types of linguistic feature. POS tagging is the assignment of a grammatical tag (e.g. noun, adjective, verb, preposition, determiner, etc.) to each word in the corpus. It is required by the linguistic filter which will only allow specific expressions for extraction. Table 1 contains lines of the corpus before the tagging and after the tagging. After POS tagging, we select all expressions (multi-words) which follow a set of specific POS patterns, that we esteem relevant in our context. Without any linguistic information, undesirable expressions such as of the, is a, etc., would also be mined.

Since most terms are made up of nouns and adjectives, [30], and sometimes prepositions, [31], we adopt linguistic filters that accept these kinds of expressions (see F1, F2 and F3). They extract terms like "operating system", "digital media reader", "wide array of streaming media", etc. The choice of linguistic filters has an influence on the precision and the recall of the output list, e.g a restrictive filter will have a positive influence on precision and a negative influence on recall [32]. We are not strict about the choice of a specific linguistic filter, because different applications need different filters.

Table 1: Sample of the corpus before and after POS tagging

BEFORE Tagging
Enjoy stunning images with this HP ENVY m7-k211dx laptop’s 17.3 high-definition touch screen and NVIDIA GeForce 840M graphics, with 2GB dedicated video memory, which display games, movies and streaming media in brilliant clarity and detail. 5th Gen Intel Core i7-5500U processor. Features a 4MB L3 cache and 2.4 GHz processor speed. 17.3 WLED-backlit high-definition touch-screen display.
AFTER Tagging
Enjoy/**VB** stunning/**JJ** images/**NNS** with/**IN** this/**DT** HP/**NNP** ENVY/**NNP** m7-k211dx/**JJ** laptop/**NN** ’s/**POS** 17.3/**CD** high-definition/**JJ** touch/**NN** screen/**NN** **and/CC** NVIDIA/**NNP** GeForce/**NNP** 840M/**CD** graphics/**NNS** **,/,** with/**IN** 2GB/**CD** dedicated/**JJ** video/**NN** memory/**NN** **,/,** which/**WDT** display/**VBP** games/**NNS** **,/,** movies/**NNS** and/**CC** streaming/**VBG** media/**NNS** in/**IN** brilliant/**JJ** clarity/**NN** and/**CC** detail/**NN** **./.** 5th/**JJ** Gen/**NNP** Intel/**NNP** Core/**NNP** i7-5500U/**JJ** processor/**NN** **./.** Features/**VBZ** a/**DT** 4MB/**NNP** L3/**NNP** cache/**NN** and/**CC** 2.4/**CD** GHz/**NNP** processor/**NN** speed/**NN** **./.** 17.3/**CD** WLED-backlit/**JJ** high-definition/**JJ** touch-screen/**JJ** display/**NN** **./.**

We will present our approach combined with each of these three filters:

F1: Noun^{+} Noun

F2: (Adj|Noun)$^{+}$ Noun

F3: (Noun Prep | Adj)* Noun^{+}

In our approach, we use a filter which also constrains the maximum number of words. This measure is to be considered as domain-dependent, being related to the linguistic peculiarities of the specialized language we are dealing with. In arts for example, terms tend to be shorter than in science and technology. The length also depends on the type of terms we accept. Terms that only consist of nouns for example, very rarely contain more than 5 or 6 words.

The process of finding this maximum length is as follows: we attempt to identify expressions of a specific length. If we do not obtain any expressions of this length, we decrease the number by 1 and make a new attempt. We carry on in this way until we find a length for which expressions exist. At this point, mining candidate expressions can take place.

4.1.2. Stoplist.

A stop-list is a list of words which are very common words. These words are not included in standard representations of documents because they are common to all the documents and cannot be good discriminators. Removing the stop words allows us to focus on the sole important words in the representations.

4.1.3. Statistical filters based on C-NC Value.

Terms are finally extracted and ranked by computing C-NC value [27]. This metric determines how much an expression is likely to be conceptually independent from its context. An expression is conceptually dependent if it requires additional words to be meaningful in its context while an expression is conceptually independent if it appears in different context. Some examples are: "touch screen", "high quality images", "plenty of storage capacity" or "Media Reader". In our study, "Media Reader" is considered as a whole since "Reader" often appears coupled with the same word "Media".

We eventually obtain for each D_i a ranked list of expressions together with their ranking according to the C-NC metric, and their frequency in the document. We choose from the list the k terms having the higher ranking. The value of k has been empirically selected: a higher value ensures more domain-specific terms but at the same time it could introduce noisy expressions. For further details, we provide in Appendix A an explanation of the computation of the C-NC value metric as well as an algorithm describing the steps to construct a list of candidate terms from a corpus [27, 33].

4.2. Contrastive Analysis

The contrastive ranking technique aims at refining extracted terms by filtering noise due to common words. We can re-rank terms according to their domain-specificity [27]. We consider terms extracted from both domain-specific document D_i and domain generic documents (the contrastive corpora) using the same method described in Section 4.1. Specifically, we have chosen as domain generic documents the Penn Treebank corpus which collects articles from the Wall Street Journal. Ferrari *et al.* [23] have employed a similar approach and corpus. The new rank $R_i(t)$ for a term t extracted from a document D_i is computed as follows:

$$R_i(t) = \arctan(\log(f_i(t)) \cdot (\frac{f_i(t)}{\frac{F_c(t)}{N_c}})$$

where $f_i(t)$ is the frequency of the term t extracted from D_i, $F_c(t)$ is the sum of the frequencies of t in the contrastive corpora, and N_c is the sum of the frequencies of all the terms extracted from D_i in the contrastive corpora. The rationale behind this ranking is as follows: If a term is less frequent in the domain-generic documents, it is considered as a domain-specific term and is consequently ranked higher. We obtain for each D_i a list of terms, together with their ranking according the function R, and their frequency in D_i. Finally, we empirically select the l terms having the higher ranking from each list. Our empirical selection is guided by the following observation: Higher values of l might introduce terms that are not domain-specific, while lower values could eliminate relevant terms.

4.3. Information Extraction

Besides domain-specific terms, we also consider *numerical information* defined as domain relevant multi-word phrases containing numerical values, since they are capable to describe precisely the technical characteristics of a product.

We use filters that extract multi-words including numbers (Integer, Double, pourcentage, degree, etc): "3.0 GHz Processor Speed", "Microsoft Windows 8.1 64-bit Operating System"; multiplication of numbers: "1920 x 1080 Resolution"; and intervals: "Turbo Boost up to 3.6GHz", "Memory expandable to 16GB". Our method is combined with each of these three filters:

F4: *Nb-Exp* (Adj|Noun)* Noun

F5: (Adj|Noun)* Noun *Nb-Exp*

F6: (Adj|Noun)* Noun *Nb-Exp* (Adj|Noun)* Noun

where, *Nb-Exp* is a measure following these patterns:

– Number (Integer, Double, pourcentage, degree, etc.): Nb, $Nb\%$, $Nb°$.

– Multiplication of numbers: $Nb \times Nb$.

– Interval: $Nb - Nb$, *up to* Nb, *down to* Nb, *expandable to* Nb, $\leq Nb$, $\geq Nb$, etc.

Inspired by the "termhood" concept used earlier, the extracted multi-words should be conceptually independent from the context in which they appears. For instance, "3.0 GHz Processor Speed" is conceptually independent whereas "3.0 GHz Processor" is not. Similarly, we identify a ranked list of domain relevant multi-words from each document D_i by applying first linguistic filters (F4, F5, F6) using POS tagging and second statistical filters inspired by C-NC Value.

When combining the C-Value score with the context information (see Appendix A), the algorithm extracts the context words (obviously not numbers) of the top list of candidates and then calculates the N-Value on the entire list of candidate multi-words. A word is considered a context word if it appears with the extracted candidate multi-words.

When computing the weight of a context word w, $weight(w) = \frac{t(w)}{n}$, $t(w)$ is not only the number of candidate multi-words w appears with, but also the number of domain-specific terms containing w and n is the total number of considered candidate multi-words and domain-specific terms.

Hence, for each D_i, we have a ranked list of multi-words that can be considered domain relevant, together with their ranking according to the C-NC metric. The more a multi-word is likely to be a domain relevant, the higher the ranking. From the list we select the k multi-words that received the higher ranking. The value of k shall be empirically selected.

5. Building the PCM

Now that we have for each product a list of domain specific terms ranked according to the C-NC metric and their frequency in the corresponding product descriptions and also a list of numerical information ranked according to the C-NC Value, the whole challenge consists in building a sound and meaningful

PCM. This process requires to find out the final features and compute the cell value for each couple product-feature.

To extract features and cell values, a first natural strategy is to perform clustering based on the similarity of the elements (terms or information) to compute groups of elements. The intuitive idea is that clusters of syntactically similar elements can be exploited to identify the common concern, witch can be organized as variability concept, and its possible values, since elements in a cluster are likely to share a common feature but with different quantification (in the case of information clusters) or description (in the case of terms clusters). Cell values can be (see the PCM of Figure 1):

- **Boolean:** can take a value of True or False, to represent whether the feature is present or not.
- **Descriptors:** noun phrases and adjectival phrases given according to this pattern: *(Adj | Noun)+*: "Digital" and "Multiformat" are two descriptor values of "Media Reader"; and "Front-facing TrueVision high-definition" and "Built-in high-definition" represent two potential values of "Webcam".
- **Quantifiers:** measures that can be Integer, Double, Partial, etc; in compliance with *Nb-Exp ((Adj | Noun)* Noun)** pattern. For instance, "1366 x 768" as "Resolution", "12GB DDR3L SDRAM" as "Memory"; "up to 3.1 GHz" as "Turbo Boost"; and "Microsoft Windows 8.1 64-bit" as "Operating System".

5.1. Terms and Information Similarity

The goal here (step ❺ in Figure 4) is to determine a weighted similarity relationship graph among terms and respectively among numerical information. Two graphs were constructed: *Terms Relationship Graph* (TRG) and *Information Relationship Graph* (IRG) in which nodes represent respectively terms and information. Assume there are n terms, they and their relationships are modeled as an undirected graph $TRG = (V, E)$, in which:

$$V = \{T_i \,|\, T_i \text{ is an individual term}, 1 \leq i \leq n\},$$

$E = \{E_{ij} \,|\, E_{ij}$ is the relationship between terms T_i and $T_j, 1 \leq i, j \leq n\}$.

Similarly, assume there are m numerical information, IRG is an undirected graph and is defined as $IRG = (V', E')$, in which:

$V' = \{I_i \,|\, I_i$ is an individual information$, 1 \leq i \leq m\}$,

$E' = \{E'_{ij} \,|\, E'_{ij}$ is the relationship between information I_i and $I_j, 1 \leq i, j \leq m\}$.

The key point is to determine the weight of each edge to express the strength of the relationships between terms and respectively between numerical information. We define the weight function $W(e)$ for each edge, where $e \in E \cup E'$. To identify coherent clusters, we determined the similarity of each pair of elements through computing syntactical heuristics.

Syntactical heuristics use edit distance and other metrics based on words' morphology to determine the similarity of two elements. We used the so-called Levenshtein edit distance [34] that computes the minimal edit operations (renaming, deleting or inserting a symbol) required to transform the first string into the second one. For example, the Levenshtein distance between "kitten" and "sitting" is 3, since the following three edits change one into the other, and there is no way to do it with fewer than three edits (substitution of "s" for "k", substitution of "i" for "e", and insertion of "g" at the end).

In this work, we do not employ semantic similarity metrics. We favour a syntactical strategy since a substantial amount of features and values are made of specific technical terms and numerical values.

5.2. Terms and Information Clustering

After building the two relationship graphs, we apply terms clustering in TRG and information clustering in IRG to identify respectively terms clusters and information clusters (step ❻ in Figure 4). The underlying idea [35] is that a cluster of tight-related elements with different granularities can be generated by changing the clustering threshold value t. The algorithm for terms and information clustering, inspired by [35], is described in Algorithm 1.

```
Algorithm 1: Terms and Information clustering
Input  : undirected graph $G$ representing a terms/information
         relationship graph (TRG or IRG); $t$ as the threshold for
         clustering
Output: list of clusters $C$
for each edge $e$ in $G$ do
    if $W(e) \geq t$ then
        $e.validType = true$;
    else
        $e.validType = false$;
    end
end
$C$ = connectedComponentsByValidEdges($G$);
```

In this algorithm, the attribute *validType* of each edge indicates whether this edge is valid for computing the connected components. We use the function $connectedComponentsByValidEdges(G)$ to decompose the graph G into connected components. Here, G corresponds to terms relationship graph or information relationship graph. In the same connected component, vertices are reachable from each other through the edges whose *validType* attribute is true. This function returns a set of connected components, and each of them forms a cluster composed by a set of tight-related elements.

To identify clusters, a threshold value t is fixed. If an edge exists between two elements and its weight is greater than or equal to t, they will belong to the same cluster; otherwise, they will not. Thus, the edges whose weights are above or equal to the threshold value are set to be valid; otherwise, the edges are invalid. Then connected components are computed by the valid edges. Each connected component is a cluster of tight-related elements sharing the same concern which represents the feature. As we decrease the threshold value, more edges are set to be valid, and we get clusters with coarser granularities.

5.3. Extracting Features and Cell Values

Finally to construct the PCM, we need to extract the features and the cell values from terms clusters and information clusters (step ❼ in Figure 4). To retrieve the feature name from a cluster, we developed a process that involved selecting the most frequently occurring phrase from among all elements (terms or information) in the cluster. This approach is similar to the method presented in [36] for summarizing customer reviews. To identify the most frequently occurring phrase we reuse the POS tags identified earlier (see Section 4.1). The elements are then pruned to retain only *Noun*$^+$ for terms clusters and *(Adj|Noun)** *Noun* for information clusters, as the other expressions were found not to add useful information for describing a feature.

Frequent itemsets are then determined for each of the clusters. In this context, frequent itemsets represent sets of expressions which frequently co-occur together in the elements attributed to the same cluster. Formally, the support of an itemset I is the number of elements in the cluster that contain all the expressions in I. Given a pre-determined itemset support threshold, s, I is considered frequent if its support is equal or larger than s[1]. Different algorithms are proposed for mining frequent itemsets including the Apriori [37] and Eclat algorithms. We chose to adopt Apriori as it is shown to be memory-efficient and hence suitable for the size of our data set. To select a feature name, the frequent itemset of maximum size, FIS_{max} is selected. Finally, to extract cell values, we substitute FIS_{max} from each element within the cluster. For example, "Digital Media Reader" and "Multiformat Media Reader" form a terms cluster. "Media Reader" is the feature name, while "Digital" and "Multiformat" are two possible values. At the same time, "1920 x 1080 Resolution", "1366 x 768 Resolution" and "2560 x 1600 Resolution" represent information cluster that gives "Resolution" as a features name and three potential values: "1920 x 1080", "1366 x 768" and "2560 x 1600". Elements which are not clustered will be considered as boolean features. Each cluster adds one column in the PCM containing the feature and the corresponding cell value for each product in the family.

[1] We set this threshold to 1 since we want to find out itemsets that occur in all elements in the cluster.

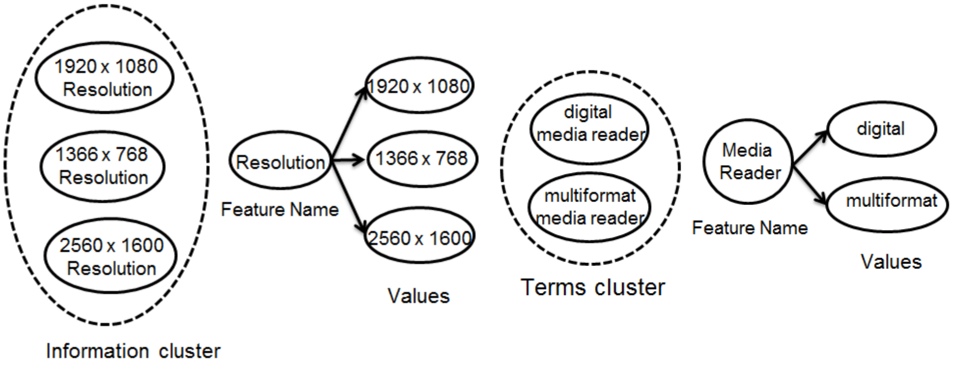


Figure 5: Extraction of Features and Cell Values

6. Tool Support

MatrixMiner offers an interactive mode where the user can import a set of product descriptions, synthesize a complete PCM, and exploit the result [24]. We also have pre-computed a series of PCMs coming from different categories of BestBuy (Printers, Cell phones, Digital SLR Cameras, Dishwashers, Laptops, Ranges, Refrigerators, TVs, Washing Machines). Our tool also provides the ability to visualize the resulting PCM *in the context* of the original textual product descriptions and also the technical specification typically to control or refine the synthesized information.

6.1. Implementation and Used Technologies

Stanford CoreNLP[2] provides a set of natural language analysis tools which can take raw text input and give the base forms of words, their parts of speech, etc. Stanford CoreNLP integrates many of NLP tools, including the Part-Of-Speech (POS) tagger that reads text in some language and assigns parts of speech to each word (and other token), such as noun, verb, adjective, etc. To tokenize and remove stop words from text we use Lucene[3] which is a high-performance, scalable Information Retrieval (IR) library for text indexing and

[2] http://nlp.stanford.edu

[3] https://lucene.apache.org

searching. *Smith-Waterman* and *Levenshtein* compute syntactical similarity based on words' morphology. They come from the Simmetrics[4] library. The specific source code of the extraction procedure is available online: https://github.com/sbennasr/matrix-miner-engine. Our Web environment reuses the editor of OpenCompare[5].

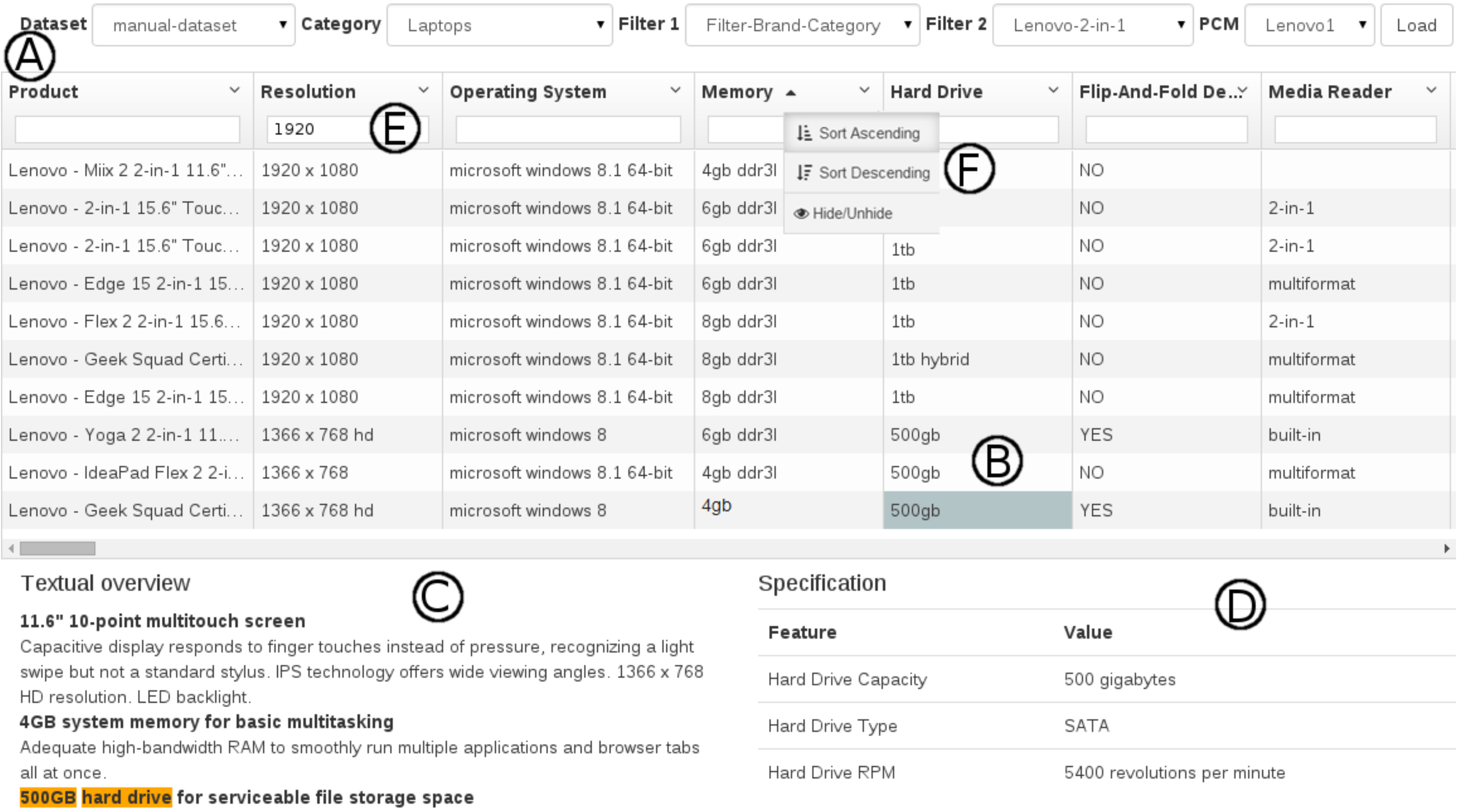
6.2. Importing, Visualizing, and Editing

The *MatrixMiner* environment is dedicated to the visualisation and edition of PCMs. Human intervention is beneficial to (1) refine/correct some values (2) re-organize the matrix for improving readability of the PCM.

As a result we developed an environment for supporting users in these activities. Our tool provides the capability for *tracing* products and features of the extracted PCM to the original product overviews and the technical specifications. Hence the PCM can be interactively controlled, complemented or refined by a user. Moreover users can restructure the matrix through the grouping or ordering of features. Overall, the features available are the following:

- select a set of comparable products. Users can rely on a number of filters (e.g. category, brand, sub categories, etc. See Figure 6, Ⓐ);
- ways to visualize the PCM with a traceability with original product descriptions. For each cell value, the corresponding product description is depicted with the highlight of the feature name and value in the text. For instance, "500GB Hard Drive" is highlighted in the text when a user clicks on "500GB" (see Figure 6, Ⓑ and Ⓒ);
- ways to visualize the PCM with a traceability with the technical specification (see Figure 6, Ⓓ). For each cell value, the corresponding specification is displayed including the feature name, the feature value and even other related features. Regarding our running example, "Hard Drive Capacity" and two related features ("Hard Drive Type" and "Hard Drive RPM") are depicted together with their corresponding values;

[4] http://sourceforge.net/projects/simmetrics
[5] https://github.com/gbecan/OpenCompare



Product	Resolution	Operating System	Memory	Hard Drive	Flip-And-Fold De...	Media Reader
	1920					
Lenovo - Miix 2 2-in-1 11.6"...	1920 x 1080	microsoft windows 8.1 64-bit	4gb ddr3l		NO	
Lenovo - 2-in-1 15.6" Touc...	1920 x 1080	microsoft windows 8.1 64-bit	6gb ddr3l		NO	2-in-1
Lenovo - 2-in-1 15.6" Touc...	1920 x 1080	microsoft windows 8.1 64-bit	6gb ddr3l	1tb	NO	2-in-1
Lenovo - Edge 15 2-in-1 15...	1920 x 1080	microsoft windows 8.1 64-bit	6gb ddr3l	1tb	NO	multiformat
Lenovo - Flex 2 2-in-1 15.6...	1920 x 1080	microsoft windows 8.1 64-bit	8gb ddr3l	1tb	NO	2-in-1
Lenovo - Geek Squad Certi...	1920 x 1080	microsoft windows 8.1 64-bit	8gb ddr3l	1tb hybrid	NO	multiformat
Lenovo - Edge 15 2-in-1 15...	1920 x 1080	microsoft windows 8.1 64-bit	8gb ddr3l	1tb	NO	multiformat
Lenovo - Yoga 2 2-in-1 11....	1366 x 768 hd	microsoft windows 8	6gb ddr3l	500gb	YES	built-in
Lenovo - IdeaPad Flex 2 2-i...	1366 x 768	microsoft windows 8.1 64-bit	4gb ddr3l	500gb	NO	multiformat
Lenovo - Geek Squad Certi...	1366 x 768 hd	microsoft windows 8	4gb	500gb	YES	built-in

Textual overview

11.6" 10-point multitouch screen
Capacitive display responds to finger touches instead of pressure, recognizing a light swipe but not a standard stylus. IPS technology offers wide viewing angles. 1366 x 768 HD resolution. LED backlight.
4GB system memory for basic multitasking
Adequate high-bandwidth RAM to smoothly run multiple applications and browser tabs all at once.
500GB hard drive for serviceable file storage space

Specification

Feature	Value
Hard Drive Capacity	500 gigabytes
Hard Drive Type	SATA
Hard Drive RPM	5400 revolutions per minute

Figure 6: The editor of *MatrixMiner* in action

- basic features of a PCM editor. Users can remove the insignificant features, complete missing values, refine incomplete values or revise suspect values if any – typically based on information contained in the textual description and the technical specification;
- advanced features of a PCM editor: means to filter and sort values (see Figure 6, Ⓔ and Ⓕ); ways to distinguish Yes, No and empty cells using different colors to improve the readability of the PCM; prioritise features by changing the columns order, *etc.*

7. Case Study and Evaluation Settings

So far, we have presented a procedure and automated techniques, integrated into the *MatrixMiner* environment, for synthesizing PCMs. For evaluating our approach, we considered a dataset coming from BestBuy, a multinational consumer electronics corporation. BestBuy provides descriptions for hundreds of thousands of products in different domains. We used the BestBuy dataset to synthesize PCMs out of informal product descriptions.

Our evaluation is made of two major studies.

Empirical Study (Section 8). We aim to measure some properties of the extracted PCMs. Is our extraction procedure able to synthesize comparable information and compact PCMs? Is there an overlap between synthesized PCMs and technical specifications?

User Study (Section 9). We aim to evaluate the quality of the information in the synthesized PCMs. How correct are features' names and values in the synthesized PCMs? Can synthesized PCMs refine technical specifications? Such a study necessitates a human assessment. We have involved users to review information of our synthesized PCMs using MatrixMiner traceabilities.

In the reminder of this section, we describe the dataset and evaluation settings we use for performing the two studies (see Section 8 and Section 9).

7.1. Data

We selected 9 products categories that cover a very large spectrum of domains (Printers, Cell phones, Digital SLR Cameras, Dishwashers, Laptops, Ranges, Refrigerators, TVs, Washing Machines) from Bestbuy. Currently, we have implemented a mining procedure on top of BestBuy API [25] for retrieving numerous product pages along different categories. We mined 2692 raw product overviews using Bestbuy API. The characteristics of the dataset are summarized in Table 2.

Table 2: Overview dataset

Products Category	#Products Overviews	#Words per Overview (Avg)
Laptops	425	350
Cell Phones	99	225
Cameras	141	279
Printers	183	277
TVs	253	283
Refrigerators	708	187
Ranges	538	275
Washing Machines	107	255
Dishwashers	238	263
Total	2692	897,020 (#Words in all Overviews)

Another important property of the dataset is that product descriptions across and within different categories do not share the same template. We came to this conclusion when manually grouping and looking at products descriptions within the same category. We have read hundreds of descriptions (169 clusters of comparable products have been obtained, see next section). We have observed that the text does not follow the same structures: There are not necessarily the same implicit sections (if any) or the same granularity of details. The absence of template challenges extractive techniques – precisely our approach does not assume any regular structure of product descriptions.

7.2. Threshold Settings

Our extraction procedure exhibits some parameters. As part of our experiments, we used the same exact parameters' values for all products categories (laptops, cell phones, cameras, printers, TVs, washing machines, etc.). Now, we describe how these values have been empirically set.

Among the automatically extracted terms, for each D_i we have selected the $k = 30$ items that received the higher ranking according to the C-NC Value. The value for k has been empirically chosen: we have seen that the majority of the domain-specific terms – to be re-ranked in the contrastive analysis phase – were actually included in the first 30 terms. We have seen that higher values of k were introducing noisy items, while lower values were excluding relevant domain-specific items.

The final term list is represented by the top list of 25 terms ranked according to the contrastive score: such a list includes domain–specific terms only, without noisy common words. It should be noted that the two thresholds for top lists cutting as well as the maximum term length can be customized for domain–specific purposes through the configuration file. As it was discussed in Section 4.1.1, the length of multi–word terms is dramatically influenced by the linguistic peculiarities of the domain document collection. We empirically tested that for the electronics domain, multi–word terms longer than 7 tokens introduce noise in the acquired term list.

Now regarding automatically retrieved numerical information, for each D_i we have selected the k = 15 items that received the higher ranking according to the C-NC Value. To calculate clusters of similar terms (resp. information), the threshold of similarity t has been set empirically after experiments at 0.6 (resp. 0.4). We have seen that the majority of well-formed clusters actually occur when the similarity thresholds are set at these values.

8. Empirical Study

In this section, we address three research questions:

- **RQ1.1: What are the properties of the extracted PCMs (being from overviews or technical specifications)?** We measure the amount of comparable information, the size of matrices, and the incompleteness of the information. Does our extraction procedure synthesize

PCMs of good quality? Do synthesized PCMs differ from technical specification PCMs?

- **RQ1.2: What is the impact of selected products on the extracted PCMs (being from overviews or technical specifications)?** A naive selection of products may lead to a non-compact and non-exploitable PCM since considered products have little in common and thus hard to compare.

- **RQ1.3: Is there an overlap between synthesized PCMs and technical specifications?** We automatically compute common features' names and values in both sides for investigating the complementarity of the two sources of information.

8.1. Dataset

We created two main datasets: overviews dataset and specifications dataset. Each of them comprises two sub-datasets (random and supervised) which contain respectively a random and supervised selection of groups of 10 products belonging to the same category (e.g. laptops).

8.1.1. Overviews Dataset (D1)

SD1.1: Overviews Dataset (random). We randomly select a set of products (also called clusters hereafter) in a given category (e.g. laptops) and we gather the corresponding products overviews. To reduce fluctuations caused by random generation [38], we run 40 iterations for each category. Results are reported as the mean value over 40 iterations.

SD1.2: Overviews Dataset (supervised clustering). A domain expert manually selected 169 clusters of comparable products against product overviews. To this end, he relies on a number of filters proposed by Bestbuy (brand, sub categories, *etc.*). The key idea is to *scope* the set of products so that they become comparable.

8.1.2. Specifications Dataset (D2)

SD2.1: Specifications Dataset (random). We keep the *same* set of products as SD1.1 (that is based on a random strategy). This time we consider technical specifications.

SD2.2: Specifications Dataset (supervised). We keep the *same* set of products as SD1.2. (that is based on a supervised clustering). We consider technical specifications.

8.2. RQ1.1. What are the properties of the extracted PCMs (being from overviews or technical specifications)?

Motivation of the study. We investigate whether our approach is able to synthesize comparable information and compact PCMs out of informal overviews. As an extreme case we do not want very large and sparse PCMs with lots of empty cells. We also compare synthesized PCMs with technical specifications. For example: how incomplete are technical specifications in comparison to our synthesized PCMs? Overall, this study gives preliminary insights about (1) the quality of our extraction procedure; (2) the strengths and weaknesses of our synthesized PCMs.

Experimental Setup. To answer our research question, we compute the following metrics over these two datasets: SD1.2 and SD2.2.

- **PCM size:** the smaller is the size of the PCM, the more exploitable is the matrix.
- **% Boolean features:** the fewer boolean features there are, the more readable is the PCM.
- **% Descriptive and quantified features**: the more quantified and descriptive features there are, the more usable and exploitable is the PCM.
- **% Empty cells (N/A):** the fewer empty cells there are, the more compact and homogeneous is the PCM.
- **% Empty cells per features category:** in particular, we measured the percentage of boolean empty cells, the percentage of quantified empty cells and the percentage of descriptive empty cells.

- **Number of empty cells per features category (Avg):** specifically, we measured the average of empty cells per boolean feature, the average of empty cells per quantified feature and the average of empty cells per descriptive feature.

The *experimental results* show that the synthesized PCMs exhibit numerous quantitative and comparable information. Indeed, the resulting overview PCMs contain in average 107.9 of features including 12.5% of quantified features and 15.6% of descriptive features. Only 13% of cell values are empty which demonstrate that our approach is able to generate *compact* PCMs.
When applying a supervised scoping, we notice that specification PCMs have 35.8% less features in average than overview PCMs. The nature of product overviews (and the verbosity of natural languages) partly explains the phenomenon. Interestingly, overview PCMs reduce the percentage of empty cells by 27.8 percentage points.

8.3. RQ1.2. What is the impact of selected products on the extracted PCMs (being from overviews or technical specifications)?

Motivation of the study. In the previous research question, the 10 products subject to comparison have been carefully selected. There may be less favourable cases for which the 10 products have been randomly chosen and are thus harder to compare, despite being in the same category. A first assumption is that, for such cases, our extraction procedure can have more difficulties to synthesize PCMs of good quality. A second assumption is that technical specifications face similar issues (e.g., the number of empty cells increases) when a naive selection of input products is performed. The question can be formulated as follows: does the set of considered products influence the properties of the PCMs (e.g., number of empty cells)? Overall, this study gives further insights about our extraction procedure and the kinds of resulting PCMs we can extract.

Experimental Setup. To answer our research question, we compare random and supervised techniques for products selection according to the metrics that we had used previously in RQ1.1. Thus, we need to compute as well these

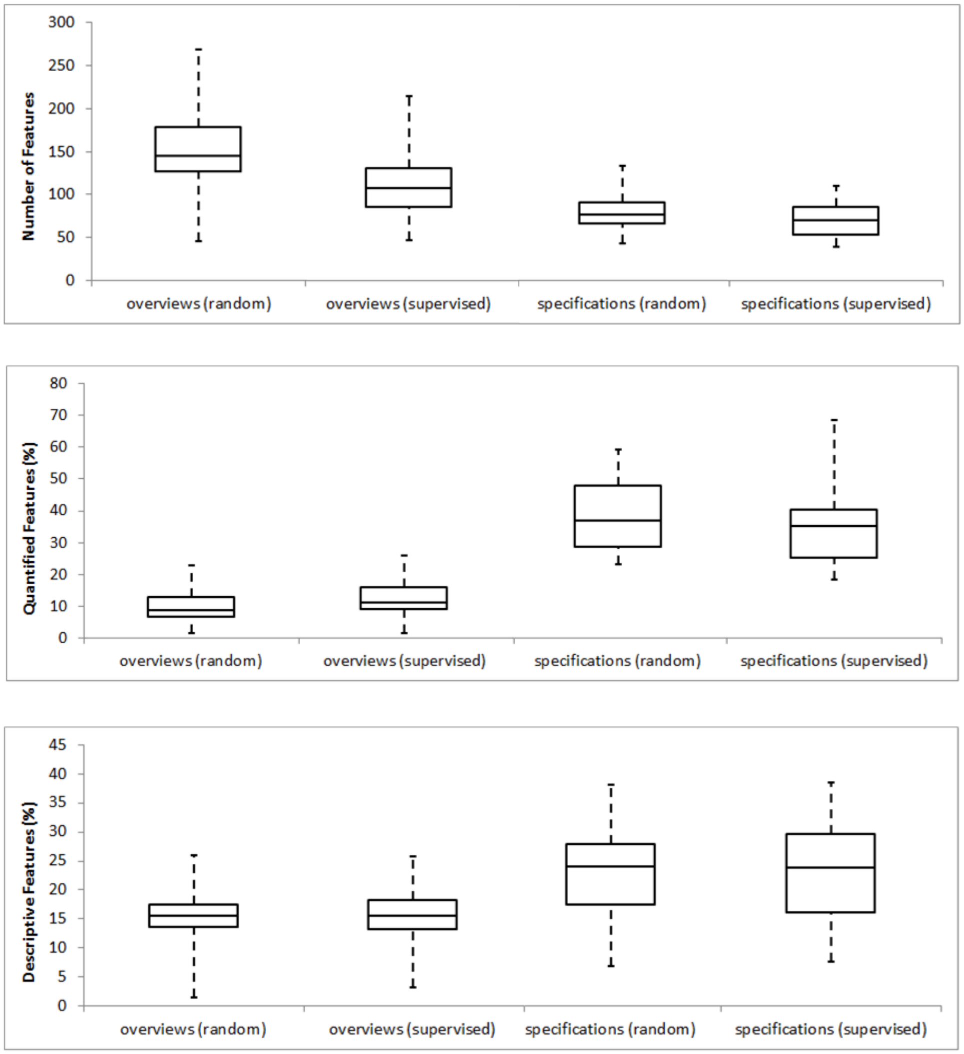


Figure 7: Features: Random vs Supervised Scoping

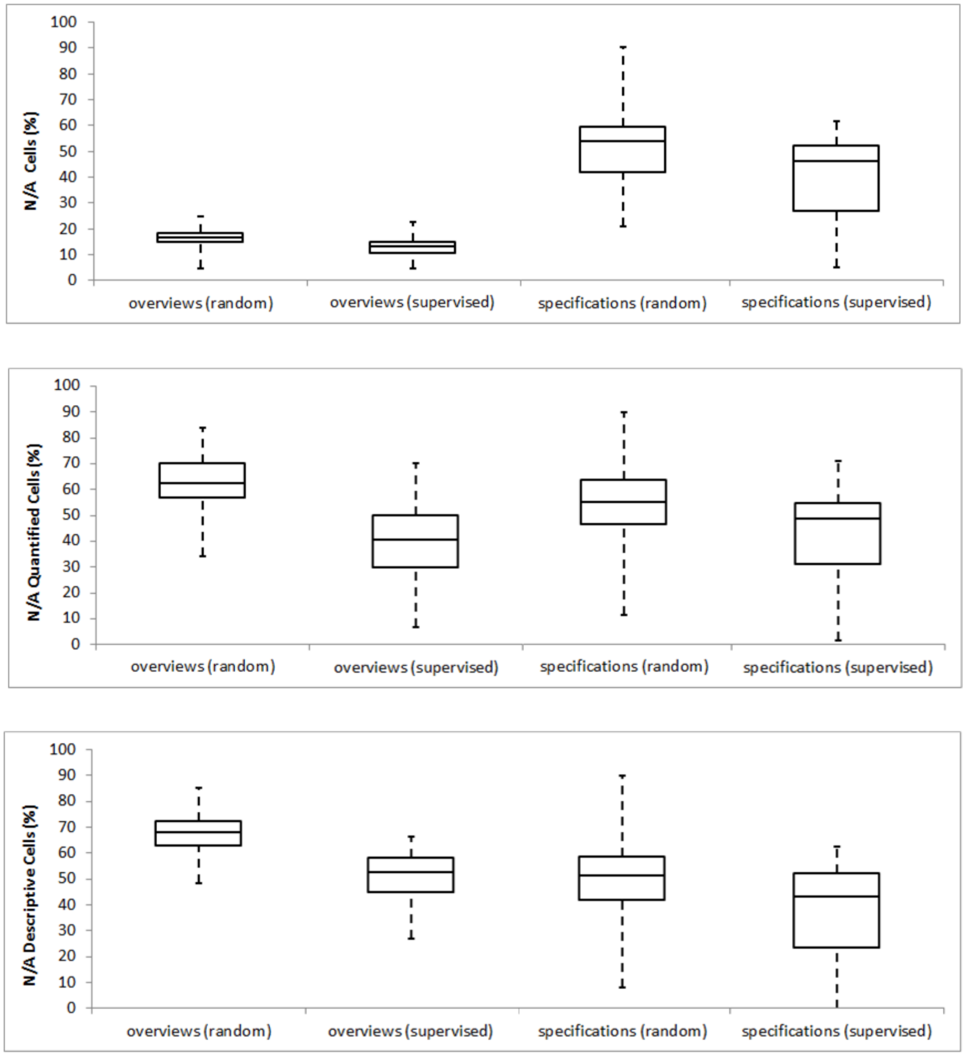


Figure 8: Cell Values: Random vs Supervised Scoping

metrics over random datasets: SD1.1 and SD2.1. Figures 7 and 8 describe the properties of the synthesized PCMs when applying random and supervised scoping.

The *experimental results* are as follows.

Complexity of PCMs. We compare the properties of overview PCMs generated using a random scoping and those engendered from a supervised scoping. We first notice that a supervised manner reduces significantly the complexity of the derived PCMs with 30.4% less cells and as much less features than a random selection of products. These results also show that our extraction is capable of exploiting the fact that products are closer and more subject to comparison.

Similarly, when we compare specification PCMs obtained respectively from a naive and supervised selection of products, we observe that a supervised scoping gives better results. Indeed, supervised PCMs contain 13.1% less cells and likely less features than random PCMs.

Homogeneity of PCMs. Following a naive scoping, we extracted overview PCMs with 16.4% of empty cells in average, whereas a manual clustering of products leads to a lower percentage of empty cells (13% in average). In particular, we observe that supervised matrices decrease by 22.7 (resp. 15.9) percentage points the percentage of quantified (resp. descriptive) empty cells. For both naive and manual selection, we obtained no boolean empty cells. Considering a supervised manner, our approach increases by around 3 percentage points the percentage of quantified features, with 2.2 less empty cells per feature in average. Supervised matrices have almost the same percentage of descriptive features as random matrices (15.6% in average) but with 1.6 less empty cells per feature in average. Similarly, supervised scoping enhances the homogeneity of the specification PCMs. The percentage of empty cells declines by 11.1 percentage points. Specifically, supervised PCMs reduce the percentage of quantified (resp. descriptive) empty cells by 11.4 (resp. 10.1) percentage points. In the same time, the supervised selection increases by

2 percentage points the percentage of quantified features and around one percentage point the percentage of descriptive features.

Key findings for RQ1.1 and RQ1.2.

- Our approach is capable of extracting numerous quantitative and comparable information (12.5% of quantified features and 15.6% of descriptive features).
- A supervised scoping of the input products reduces the complexity (in average 107.9 of features and 1079.7 of cells) and increases the homogeneity and the compactness of the synthesized PCMs (only 13% of empty cells).
- An open issued made apparent with RQ1.1 and RQ1.2 is that the size of PCMs can be important while PCMs, being from overviews or technical specifications, can be incomplete. It motivates the next research question RQ1.3. on the complementarity of both PCMs.

8.4. RQ1.3. Is there an overlap between synthesized PCMs and technical specifications?

Motivation of the study. The purpose of RQ1.3 is to analyze the relationship and possible overlaps between generated PCMs (coming from the textual overviews) and specification PCMs. In case generated PCMs can be made more complete with the technical specifications (or vice-versa), it can (1) increase the quality of PCMs (e.g., empty cells are replaced by actual values) (2) reduce the user effort in case an information is missing or unclear (he or she can refer to the other source of information).

Experimental Setup. To address RQ1.3, we compared the features and the cell values for the same set of products in both overview and specification PCMs using the following metrics:

- % Correct features in the overview matrices comparing to the specification matrices (`Features Over in Spec`): we consider that a feature in an overview PCM is correct, if it is similar to another feature in the specification PCM.

- % Correct features in the specification matrices comparing to the overview matrices (`Features Spec in Over`): we follow the same principle described above.
- % Correct cell values in the overview matrices comparing to the specification matrices (`Cells Over in Spec`): for a given product and two similar features in the overview PCM and the specification PCM, we consider that the cell value in the overview PCM is correct if it is similar to the cell value in the the specification PCM.
- % Correct cell values in the specification matrices comparing to the overview matrices (`Cells Spec in Over`): we apply the same principle as `Cells Over in Spec`.

Two features are similar if at least one of them occurs in the other. Now, for two similar features and a given product, two cell values are similar if at least one of them contains the other. Figure 9 illustrates the overlap between overview PCMs and specification PCMs.

The *experimental results* are as follows.

Features Overlap. Overview matrices cover approximately half of the features in the specification matrices (49.7% in average, 51.0% for median). However, these latters cover only 20.4% of features in the overview matrices (20.6% for median).

Cells Overlap. Overview matrices cover 26.2% of cell values in the specification PCMs (in average, 26.3% for median), while these latters cover only 8.6% of cell values in the overview PCMs (8.5% for median).

The results provide evidence that, with our automated extraction from overviews, there is also a potential to complement technical specifications of products. Another interesting point is that the user can rely on the overlapping features between specifications and overviews to prioritize features and then keep the most relevant ones, in order to reduce the complexity of the overview PCM.

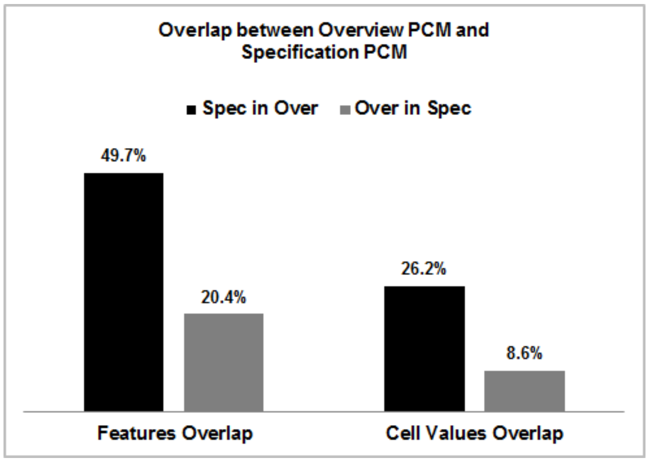


Figure 9: Complementarity of Overview PCMs and Specification PCMs (RQ1.3)

Key findings for RQ1.3.

- A significant portion of features (49.7%) and cell values (26.2%) is recovered in the technical specifications.
- The proportion of overlap of overview PCMs regarding specification PCMs is significantly greater than the overlap of the latter regarding overview matrices. This is explained by the fact that the natural language is richer, more refined and more descriptive compared to a list of technical specifications.
- Overall, users can benefit from an interesting overlap. They can reduce the complexity of the PCMs by only focusing on overlapping features' names and values. They can also complete missing cell values or even refine some information of PCMs. It motivates the next "user study".

9. User Study

Our previous study does not evaluate the quality of the information in the synthesized PCMs. For example, we ignore how correct are features' names and values in the synthesized PCMs coming from informal and textual overviews (see RQ2.1 below). We also want to further understand the relationship between technical specifications and our synthesized PCMs (see RQ2.2 below). Such investigation necessitates a more qualitative judgment and an human assessment; we have involved some participants to review information of our synthesized PCMs using MatrixMiner traceabilities.

9.1. Experiment Settings

Dataset: We considered the same set of supervised overview PCMs used earlier in the empirical study: the dataset SD1.2 (169 PCMs in the total). These PCMs cover a very large spectrum of domains (Printers, Cell phones, Digital SLR Cameras, Dishwashers, Laptops, Ranges, Refrigerators, TVs, Washing Machines, *etc.*). These PCMs are made from various sizes, going from 47 to 214 columns (features), and 10 rows (products).

Participants: The PCMs were evaluated separately by 20 persons, each using their own computers. Participants were computer science researchers and engineers at Inria (France). They have strong background in software engineering. They were not aware of our work.

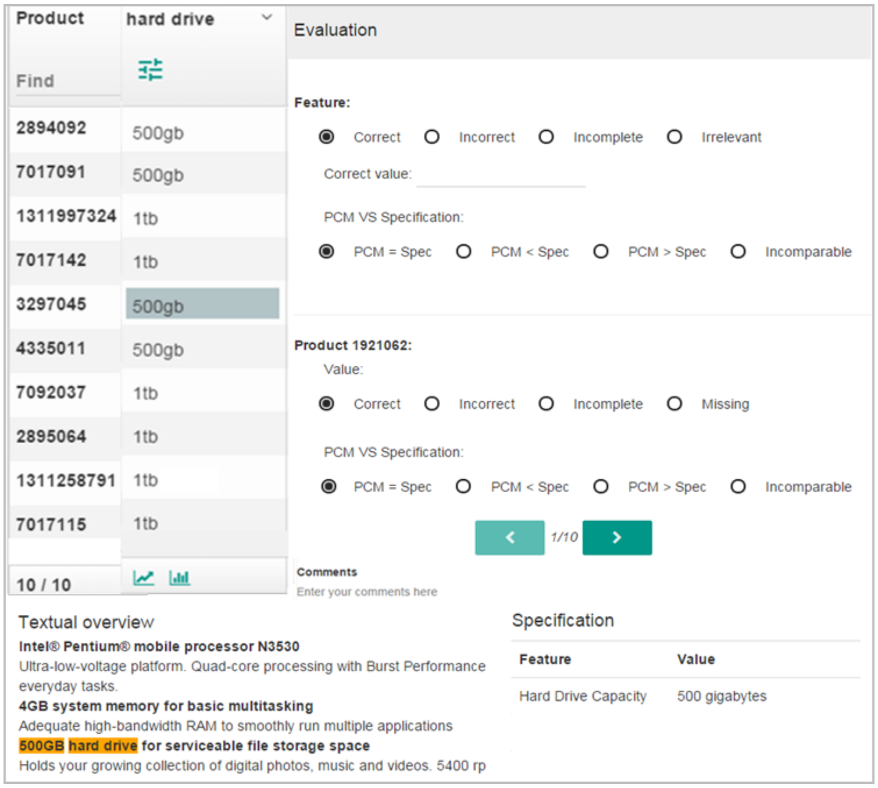
Evaluation Sessions: We organized one evaluation session in which we explain the goal of the experiment to the evaluators. We provided a tutorial describing the tool they would have to use, as well as the concepts they were about to evaluate and related illustrative examples. We displayed randomly one column at a time (from any PCM) and the evaluator has to attribute scores for the feature and cell values. The evaluation session took one hour in total.

The evaluators have to *validate* features and cell values in the PCM against the information contained in the original text. To this end, the tool provides ways to visualize the PCM with a traceability with original product descriptions. For each cell value, the corresponding product overview is depicted with the highlight of the feature name and the value in the text.

For each displayed column, the checking process consists of:

1. Looking at the feature, the evaluators have to state whether the feature is correct, incorrect, incomplete or irrelevant.
2. Looking at each cell value, the evaluators have to state whether the expected cell value is correct, incorrect, incomplete or missing.

The evaluators can propose a correction of incorrect, incomplete or missing information.



Product	hard drive
Find	
2894092	500gb
7017091	500gb
1311997324	1tb
7017142	1tb
3297045	500gb
4335011	500gb
7092037	1tb
2895064	1tb
1311258791	1tb
7017115	1tb
10 / 10	

Evaluation

Feature:

● Correct ○ Incorrect ○ Incomplete ○ Irrelevant

Correct value:

PCM VS Specification:

● PCM = Spec ○ PCM < Spec ○ PCM > Spec ○ Incomparable

Product 1921062:

Value:

● Correct ○ Incorrect ○ Incomplete ○ Missing

PCM VS Specification:

● PCM = Spec ○ PCM < Spec ○ PCM > Spec ○ Incomparable

< 1/10 >

Comments

Enter your comments here

Textual overview

Intel® Pentium® mobile processor N3530
Ultra-low-voltage platform. Quad-core processing with Burst Performance everyday tasks.
4GB system memory for basic multitasking
Adequate high-bandwidth RAM to smoothly run multiple applications
500GB hard drive for serviceable file storage space
Holds your growing collection of digital photos, music and videos. 5400 rp

Specification

Feature	Value
Hard Drive Capacity	500 gigabytes

Figure 10: Overview of the environment during PCMs evaluation (by column)

It should be noted that we did not ask to participants to determine whether some features have been missed by our extraction. There may be features in the textual descriptions but not present in the extracted PCMs. Identifying such missing features would require a complete review of the text from a domain expert and is labor-intensive.

To keep the amount of manual effort reasonable, we therefore only accounted for end-user validation of the extracted features.

Furthermore the evaluators had to specify for each column whether the PCM contains more/less refined information (features and cell values) than in the specification:

- PCM = Spec: the PCM and the specification contain the same information.
- PCM > Spec: the PCM contains more information comparing to the specification.
- PCM < Spec: the PCM contains less information comparing to the specification.
- incomparable: the information in the PCM and the specification do not match.

The tool offers ways to visualize the PCM with a traceability with the specification. For each cell value, the corresponding specification is depicted including the feature name and the cell value. The evaluators can add a comment at the end of the evaluation of each column.

Evaluation Scenario: Participants performed the evaluation as follows. We displayed one column at a time (see Figure 10 and the column hard drive). The evaluators have to validate the feature and cell values. They can either refer to the original text (Figure 10 shows how we highlight both feature names and values) or to the technical specification. Once the evaluation of one column is finished, the evaluator submits his/her evaluation and starts again a new evaluation for a new column.

Evaluation Outputs: We obtained 118 evaluated features and 1203 evaluated cell values during an evaluation session of one hour. Overall, 50% of evaluated features belong to ranges, 24.57% come from laptops, 16.10% are related to printers, and 9.32% correspond to features of refrigerators, TV and washing machines. On the other hand, 45.95% of evaluated cell values are about ranges, 22.61% are contained in laptops PCMs, 16.90% of values belong to printers and 14.52% are related to refrigerators, TV and washing machines.

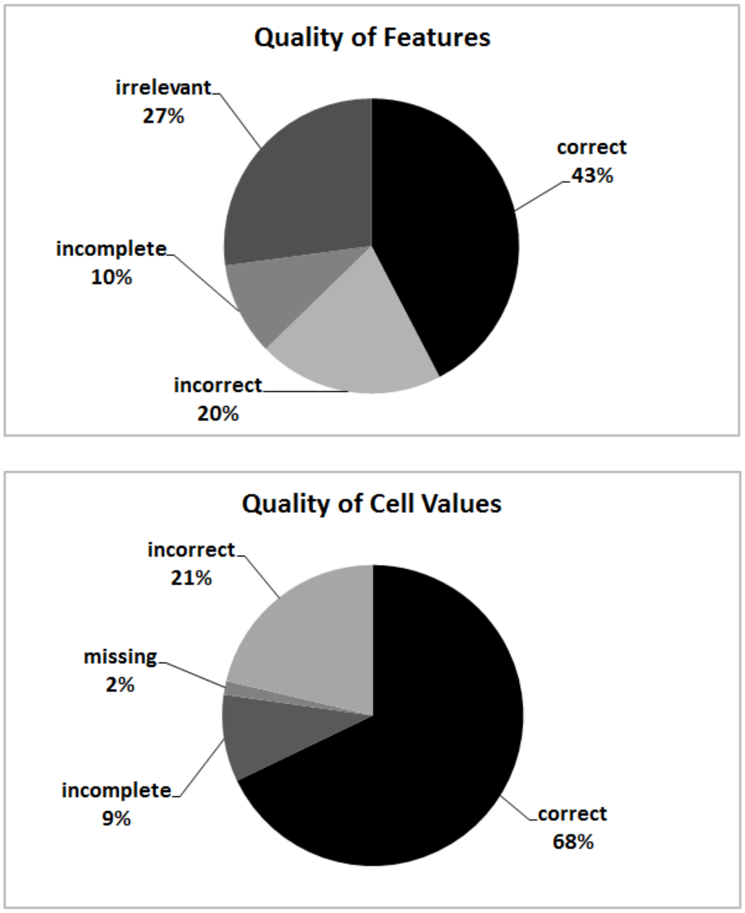


Figure 11: Quality of Features and Cell Values

9.2. RQ2.1. How do users perceive the information of synthesized PCMs when confronted to the original, textual overviews?

Motivation of the study. In Figure 10, the feature hard drive and its 10 cell values make sense. However our extraction procedure can sometimes introduce errors in the PCMs: features' names or cell values may be incorrect or irrelevant. In this study, we aim to qualitatively confront the information in the synthesized

PCM with the original text. As there is no automated oracle, we rely on humans judgements to assess it. Thanks to users we can, for instance, compute the percentage of correct features and cell values.

This study aims to provide some insights on the quality of our extraction procedure. Another motivation of this study is to investigate how MatrixMiner traceability mechanisms can help users in reviewing and controlling the information – we only highlight some elements in the texts and avoid a reading of the entire text.

Experimental results are reported in Figure 11 and show that our automatic approach retrieves 43% of correct features and 68% of correct cell values in one step and without any user intervention, showing the usefulness of our approach. We also note that 10% of features and 9% of cell values are incomplete which means that are correct but are not enough precise. This means that we are very close to the right values. Using the traceability with the original text, users can easily retrieve the full information and complete the PCM.

Only 20% of features and 21% of cell values are incorrect with 2% of these latters are missing. In the same time, we observe that 27% of features extracted automatically are irrelevant (one cannot objectively know the preferred features for a user). Again, the results provide evidence that the role of the user remains crucial. Indeed, the user is able to correct or complete the information in the PCM thanks to the traceability with the original product descriptions and the specifications. Also, he/she can remove the features which he/she consider irrelevant.

9.3. RQ2.2. How do users perceive the overlap between synthesized PCMs and technical specifications?

Motivation of the study. Thanks to MatrixMiner and basic matching techniques, we can relate information in the synthesized PCMs with technical specifications. Previous sections (see RQ1.1, RQ1.2 and RQ1.3) suggest some potential, but we ignore the exact relationship: Is it a refinement? Is it a new information? Here, we gather insights on the overlap between the synthesized PCM (overviews) and the technical specifications. We investigate how our synthesized

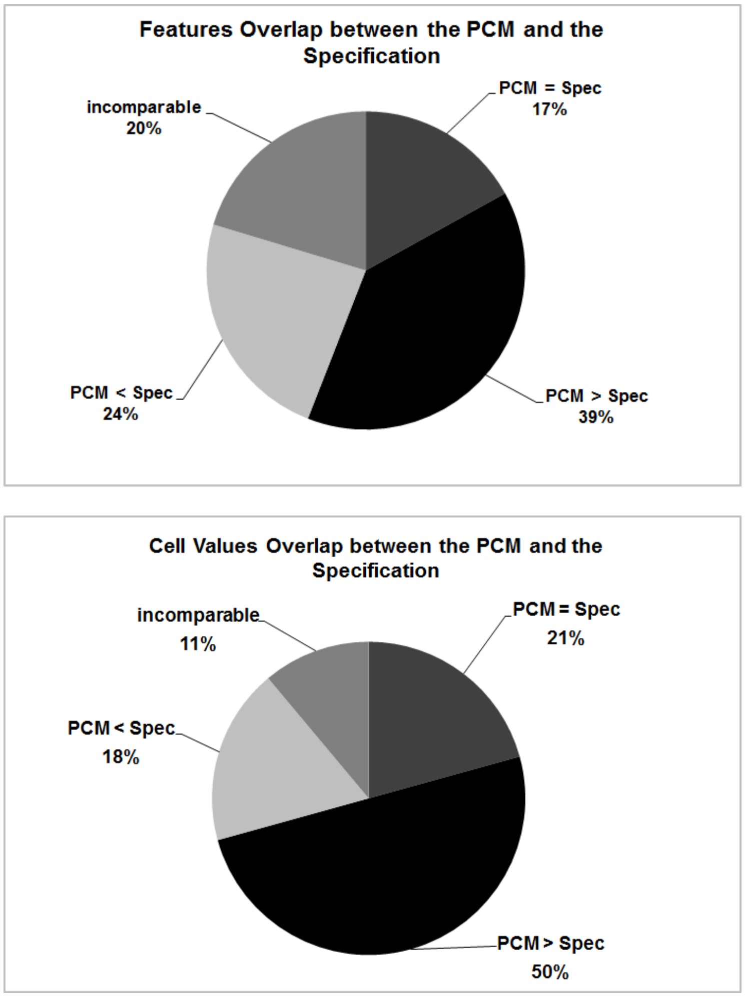


Figure 12: Overlap between PCMs and the Specifications

PCMs can complement or refine technical specifications (and vice-versa). It can drastically improve or expand the information in PCMs.

Experimental Results. We compared the features and the cell values for the same set of products in both synthesized PCMs and the specifications. Figure 12 shows that regarding 56% (resp. 71%) of the total features (resp. cell values), we have as much or more information in the PCMs than in the specifications.

In particular, the PCMs outperform the specifications with 39% more refined features, while these latters contain only 24% more refined features than the PCMs. We reported that 17% of features are equal in both PCMs and the specifications. Concerning cell values, PCMs are more accurate than the specifications in 50% of cases and equal to the specifications in 21% of cases. Only 18% of cell values are more detailed in the specifications.

Furthermore, we report that 20% of features and 11% of cell values are incomparable which means that the information are different in the PCMs and the specifications. These results are of great interest for the user since he/she can get a complete view when merging these incomparable information, and thus can maintain or refine the information in the resulting PCMs. This shows the importance of exploiting the text.

Key findings for RQ2.1 and RQ2.2

- Our automatic approach retrieves 43% of correct features and 68% of correct cell values. Users can rely on MatrixMiner's traceability to control, edit or remove some features' names and values without having to review the entire textual descriptions.

- Results show that we have as much or more information in the synthesized PCMs than in the technical specifications for a significant portion of features (56%) and cell values (71%). Again, users can rely on MatrixMiner to refine or expand the information in both sources.

9.4. Discussion

As shown in the evaluation, the role of users may remain crucial to complete the information in the PCM. In this context, the tool provides the capability for tracing the information contained in the extracted PCM with:

1. The original product overviews: MatrixMiner allows users to only focus on the related part of the text, and not the whole text, to correct/refine the information in the PCM.

2. The technical specifications: the user can merge the information coming from both sources to get a complete view.

In this way, the PCM can be easily controlled, complemented or refined by a user. Our tool is based on a syntactical matching to maintain the traceability. This matching is beneficial since it provides some useful information:

- 43% of correct features and 68% of correct cell values, by referring to the text.

- for 56% of features and 71% of values, we have as much or more information in the PCMs, by referring to the specifications.

More sophisticated, semantic-based mechanisms can also be considered to map information in the synthesized PCMs with technical specifications. The user effort can sometimes be obsolete when the information is just not there, neither in the text nor in the specifications.

10. Threats to Validity

An *external threat* to validity has to do with the context of our case study, which is limited to the BestBuy dataset. We considered numerous categories and products to diversify the textual corpus. The product descriptions used in our experiments do not generally have the same template. In the empirical study, a domain expert read and analysed manually 1215 product overviews to cluster comparable products. We note that even within the same category, products may describe the same feature in different ways, or chose to describe different features. These factors have concrete implications in the PCM definition. For instance 13% of the cells are empty, which shows the diversity of feature descriptions in the overview.

As a consequence, a template-based approach is less likely to be applied in an effective manner, whereas a NLP approach is more agnostic in terms of product description. Nevertheless, we plan to apply our procedure on other websites than BestBuy in order to investigate the generalizability of our approach. However, we are confident that our approach is independent from Bestbuy and can be technically configured for other Websites.

There are *internal threats* to validity. A first internal threat comes from the manual optimization of the clustering thresholds (regarding terms and information) for the evaluation of the heuristic. Another set of thresholds could generate less favorable results. Similarly, a manual optimization of top lists thresholds according to C-NC Value or domain-specificity metrics, might affect the quality of the domain specific terms. We did not optimize parameters tresholds for a given (sub)domain: we used the same parameters' values for the different product categories (laptops, cell phones, cameras, printers, TVs, washing machines, etc.).

Second, the computation of overlapping parts between the specifications and the overviews is based on an equivalence between features names and cell values (see RQ1.3). We chose a simple measurement based on occurrence of names to reduce the false positives. A more sophisticated measure (*e.g.* based on synonyms) could identify more overlapping information with the risk of providing false positives.

Besides we chose to consider only 10 products. The main rationale is that we wanted to obtain compact PCMs with numerous comparable information. Another number of products might give other results. A big number of products could increase the complexity of the PCM and a very small number of products could lead to few comparable information. It is an open problem to determine for which number the approach is applicable and useful.

The final threat we discussed here is related to the evaluation process in the user study. It is not always evident for the evaluators to decide whether the synthesized PCM has more or less information than in the specification (see RQ2.2). In some cases, the evaluator could find more and different refined information regarding a same feature in the two sides.

11. Related Work

Terminology Extraction. Term extraction systems make use of various degrees of linguistic filtering. Statistical measures can be employed such as Term Frequency/Inverse Document Frequency (TF/IDF) [39], the C-NC Value method [33], or lexical association measures (e.g., log likelihood [40]). Extensive semantic resources can be considered as well [41, 42].

Another interesting line of research is based on the comparison of the distribution of terms across corpora. Under this approach, identification of relevant term candidates is carried out through inter-domain contrastive analysis [43, 44, 32]. Our process relies on methods for term recognition and information extraction. We adapt such techniques for specifically extracting and organizing variability information into a PCM.

Mining Features. The majority of existing approaches are about mining features from textual requirements and legacy documentation [45, 35, 19, 46, 47, 48]. Frakes *et al.* [45] implemented the DARE tool which extracts features based on term frequency, while these works [35, 19, 46, 47, 48] rely on clustering technology to determine features. Chen *et al.* [35] evaluate manually the similarity among requirements. Alves *et al.* [19] employ automated techniques such as the Vector Similarity Metric (VSM) and Latent Semantic Analysis (LSA).
Niu *et al.* [46, 47] identify *Functional Requirements Profiles* in functional requirements using *Lexical Affinities (LA)*. Weston *et al.* [48] determine cross-cutting concerns, called *Early Aspects*, to obtain features. They adopt LSA aided with syntactic and semantic analysis.

On the other hand, other approaches [49, 50, 51] perform features extraction from public product descriptions, as in our case. Dumitru *et al.* [50] utilize text mining and an incremental diffusive clustering algorithm to discover features. The proposed approach is also able to generate feature recommendations. Acher *et al.* [51] look for variability patterns within structured product descriptions, expressed in tables. Ferrari *et al.* [23] apply natural language processing techniques to mine commonalities and variabilities from brochures. We rely on similar techniques for extracting variability information, but we also have to develop new ones to take the specificities of inputs (rather short descriptions of products) and outputs (PCMs) into account. In [36], the authors propose a set of techniques for mining and summarizing product reviews based on data mining and natural language processing methods. The objective is to provide a feature-based summary of a large number of customer reviews of a product sold online.

Synthesis of Feature Models. Numerous techniques for synthesizing feature models have been proposed (e.g., [10, 7, 11, 52]). Our extraction work is complementary as we identify and structure features into a product matrix. Boolean feature models can be synthesized from a set of products (or configurations) if a hierarchy is specified, inferred or arbitrarily chosen [53, 54, 7, 11, 52]. Chen *et al.* [35], Alves *et al.* [55], Niu *et al.* [56], and Weston *et al.* [48] applied information retrieval (IR) techniques to abstract requirements from existing specifications, typically expressed in natural language.

Few approaches have been proposed to extract variability from informal product descriptions [50, 7]. Dumitru *et al.* [50] implemented a recommender system that models and recommends product features for a given domain. Davril *et al.* [7] provided an automated approach for building feature models from publicly available product descriptions found in online product repositories such as SoftPedia. They based the feature extraction technique on their previously data mining procedure [6]; then the synthesis of an FM from a product-by-feature matrix is performed. Our extraction procedure could be used to replace the manual elaboration of such a matrix.

Bakar *et al.* [13] performed a systematic literature review of approaches in feature extractions from natural language requirements for reuse in software product line engineering. Our extraction process follows similar steps as the considered studies. A key difference is the last step: instead of forming a feature model, we aim to build a PCM (or product-by-feature matrix). There are two reasons. First, a PCM is *per se* a widely used abstraction for understanding and comparing products within a domain [53, 7, 8]. Second, the formation of feature models from a PCM is possible with synthesis techniques [10, 7, 6, 11, 12, 52]. However, as shown in our empirical study, extracted PCMs contain numerous numerical, string or unknown values. It challenges the development of novel feature model synthesis techniques capable of handling such values. Another open question is how humans cognitively perceive feature models comparatively to PCMs, especially when the number of features tends to be quite important (e.g., hundreds). Besides, Bakar *et al.* [13] reported the lack of publicly available tools. In response we provided MatrixMiner, a Web tool `http://matrix-miner.variability.io` and an open source project `https://github.com/OpenCompare/matrix-miner`.

12. Conclusion

Numerous organizations or individuals rely upon publicly available, mostly informal *product descriptions* for analysing a domain and a set of related products. As a manual analysis is labour-intensive and time-consuming, we developed an approach to automatically extract *product comparison matrices (PCMs)* from a set of informal product descriptions written in natural languages. Instead of reading and confronting the information of products case-by-case, we aimed to deliver a compact, synthetic, and structured view of a product line – a PCM.

We developed an automated process, based on term recognition, contrastive analysis, information extraction, clustering, and similarities, capable of extracting and structuring features and values into a PCM. Our empirical results show that the synthesized PCMs are compact and exhibit numerous quantitative, comparable information: 12.5% of quantified features, 15.6% of descriptive features, with only 13.0% of empty cells. A supervised selection of comparable products (scoping) is necessary though. The user study shows that our automatic approach retrieves a significant portion of correct information: 43% of correct features and 68% of correct cell values in the generated PCMs. We also show that the extracted PCMs supplement the technical specification: for 56% of features and 71% of cell values, we have as much or more information in the synthesized PCMs than in the specifications. We provide empirical evidence that there is a potential to complement or even refine technical information of products thanks to our extraction.

The evaluation insights drive the design of the *MatrixMiner*, a web environment with an interactive support for synthesizing, visualising and editing PCMs. *MatrixMiner* also provides the ability to *trace* products and features of synthesized PCMs to the original product descriptions (textual overviews) as well as technical specifications. Likewise users can understand, control and refine the information of the synthesized PCMs within the context of product descriptions and specifications.

The presented work has the potential to crawl scattered and informal product descriptions that abound on the web. Other inputs such as online reviews of products can be considered as well. The identification of relationships between features (e.g., conflict) is also an interesting perspective. We are integrating the tool-supported approach as part of OpenCompare an initiative for the collaborative edition, the sharing, the standardisation, and the open exploitation of PCMs. The goal is to provide an integrated set of tools (e.g., APIs, visualizers, configurators, recommenders, editors) for democratizing their creation, import, maintenance, and exploitation. *MatrixMiner* is available online:

http://matrix-miner.variability.io

Acknowledgements

Financial support for this work was provided by Luxembourg's National Research Fund (FNR) under grant number FNR/P10/03.

[1] K. Pohl, G. Böckle, F. J. van der Linden, Software Product Line Engineering: Foundations, Principles and Techniques, Springer-Verlag, 2005.

[2] S. Apel, D. Batory, C. Kästner, G. Saake, Feature-Oriented Software Product Lines: Concepts and Implementation, Springer-Verlag, 2013.

[3] I. Reinhartz-Berger, Can domain modeling be automated? levels of automation in domain modeling, in: SPLC '14, 2014, p. 359.

[4] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, A. S. Peterson, Feature-oriented domain analysis (foda) feasibility study, Tech. rep., DTIC Document (1990).

[5] S. Iyengar, The Art of Choosing, Twelve, 2010.

[6] N. Hariri, C. Castro-Herrera, M. Mirakhorli, J. Cleland-Huang, B. Mobasher, Supporting domain analysis through mining and recommending features from online product listings, IEEE Transactions on Software Engineering 99 (PrePrints) (2013) 1. doi:http://doi.ieeecomputersociety.org/10.1109/TSE.2013.39.

[7] J.-M. Davril, E. Delfosse, N. Hariri, M. Acher, J. Cleland-Huang, P. Heymans, Feature model extraction from large collections of informal product descriptions, in: ESEC/FSE'13, 2013.

[8] G. Bécan, N. Sannier, M. Acher, O. Barais, A. Blouin, B. Baudry, Automating the formalization of product comparison matrices, in: Proceedings of the 29th ACM/IEEE international conference on Automated software engineering, ACM, 2014, pp. 433–444.

[9] N. Sannier, G. Bécan, M. Acher, S. B. Nasr, B. Baudry, Comparing or configuring products: are we getting the right ones?, in: The Eighth International Workshop on Variability Modelling of Software-intensive Systems, VaMoS '14, 2014, p. 9.

[10] N. Andersen, K. Czarnecki, S. She, A. Wasowski, Efficient synthesis of feature models, in: Proceedings of SPLC'12, ACM Press, 2012, pp. 97–106. doi:10.1145/2362536.2362553.

[11] R. E. Lopez-Herrejon, L. Linsbauer, J. A. Galindo, J. A. Parejo, D. Benavides, S. Segura, A. Egyed, An assessment of search-based techniques for reverse engineering feature models, Journal of Systems and Software.

[12] G. Bécan, R. Behjati, A. Gotlieb, M. Acher, Synthesis of attributed feature models from product descriptions, in: 19th International Software Product Line Conference (SPLC'15), Nashville, TN, USA, 2015, (research track, long paper).

[13] N. H. Bakar, Z. M. Kasirun, N. Salleh, Feature extraction approaches from natural language requirements for reuse in software product lines: A systematic literature review, Journal of Systems and Software 106 (2015) 132–149.

[14] R. Olaechea, D. Rayside, J. Guo, K. Czarnecki, Comparison of exact and approximate multi-objective optimization for software product lines, in: SPLC'14, 2014, pp. 92–101.

[15] C. Kästner, A. Dreiling, K. Ostermann, Variability mining: Consistent semiautomatic detection of product-line features, IEEE Transactions on Software Engineering.

[16] S. Nadi, T. Berger, C. Kästner, K. Czarnecki, Mining configuration constraints: Static analyses and empirical results, in: Proceedings of the 36th International Conference on Software Engineering (ICSE), 2014.

[17] K. Chen, W. Zhang, H. Zhao, H. Mei, An approach to constructing feature models based on requirements clustering, in: Proceedings of RE'05, 2005, pp. 31–40. doi:10.1109/RE.2005.9.
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1531025

[18] N. Niu, S. M. Easterbrook, Concept analysis for product line requirements, in: K. J. Sullivan, A. Moreira, C. Schwanninger, J. Gray (Eds.), AOSD, ACM, 2009, pp. 137–148.

[19] V. Alves, C. Schwanninger, L. Barbosa, A. Rashid, P. Sawyer, P. Rayson, C. Pohl, A. Rummler, An exploratory study of information retrieval techniques in domain analysis, in: SPLC'08, 2008, pp. 67–76.

[20] E. Bagheri, F. Ensan, D. Gasevic, Decision support for the software product line domain engineering lifecycle, Automated Software Engineering 19 (3) (2012) 335–377.

[21] N. Itzik, I. Reinhartz-Berger, SOVA - A tool for semantic and ontological variability analysis, in: Joint Proceedings of the CAiSE 2014 Forum and CAiSE 2014 Doctoral Consortium, 2014, pp. 177–184.
URL http://ceur-ws.org/Vol-1164/PaperDemo06.pdf

[22] I. Reinhartz-Berger, A. Sturm, Y. Wand, Comparing functionality of software systems: An ontological approach, Data Knowl. Eng. 87 (2013) 320–338. doi:10.1016/j.datak.2012.09.005.
URL http://dx.doi.org/10.1016/j.datak.2012.09.005

[23] A. Ferrari, G. O. Spagnolo, F. dell'Orletta, Mining commonalities and variabilities from natural language documents, in: SPLC, 2013, pp. 116–120.

[24] S. Ben Nasr, G. Bécan, M. Acher, J. a. B. Ferreira Filho, B. Baudry, N. Sannier, J.-M. Davril, Matrixminer: A red pill to architect informal product descriptions in the matrix, in: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2015, ACM, 2015, pp. 982–985.

[25] http://www.bestbuy.com, Bestbuy (2014).

[26] N. Sannier, M. Acher, B. Baudry, From comparison matrix to variability model: The wikipedia case study, in: ASE'13, IEEE, 2013, pp. 580–585.

[27] F. Bonin, F. Dell'Orletta, G. Venturi, S. Montemagni, A contrastive approach to multi-word term extraction from domain corpora, in: Proceedings of the "7th International Conference on Language Resources and Evaluation", Malta, 2010, pp. 19–21.

[28] F. Dell'Orletta, Ensemble system for part-of-speech tagging, Proceedings of EVALITA 9.

[29] P. Drouin, Term extraction using non-technical corpora as a point of leverage, Terminology 9 (1) (2003) 99–115.

[30] J. C. Sager, A practical course in terminology processing, John Benjamins Publishing, 1990.

[31] J. S. Justeson, S. M. Katz, Technical terminology: some linguistic properties and an algorithm for identification in text, Natural language engineering 1 (01) (1995) 9–27.

[32] R. Basili, A. Moschitti, M. T. Pazienza, F. M. Zanzotto, A contrastive approach to term extraction, in: Terminologie et intelligence artificielle. Rencontres, 2001, pp. 119–128.

[33] K. T. Frantzi, S. Ananiadou, The c-value/nc-value domain-independent method for multi-word term extraction, Journal of natural language processing 6 (3) (1999) 145–179.

[34] R. A. Wagner, M. J. Fischer, The string-to-string correction problem, Journal of the ACM (JACM) 21 (1) (1974) 168–173.

[35] K. Chen, W. Zhang, H. Zhao, H. Mei, An approach to constructing feature models based on requirements clustering, in: Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on, IEEE, 2005, pp. 31–40.

[36] M. Hu, B. Liu, Mining and summarizing customer reviews, in: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2004, pp. 168–177.

[37] R. Agrawal, R. Srikant, et al., Fast algorithms for mining association rules, in: Proc. 20th int. conf. very large data bases, VLDB, Vol. 1215, 1994, pp. 487–499.

[38] A. Arcuri, L. Briand, A practical guide for using statistical tests to assess randomized algorithms in software engineering, in: Software Engineering (ICSE), 2011 33rd International Conference on, IEEE, 2011, pp. 1–10.

[39] G. Salton, C. Buckley, Term-weighting approaches in automatic text retrieval, Information processing & management 24 (5) (1988) 513–523.

[40] T. Dunning, Accurate methods for the statistics of surprise and coincidence, Computational linguistics 19 (1) (1993) 61–74.

[41] D. Maynard, S. Ananiadou, Term extraction using a similarity-based approach, Recent advances in computational terminology (1999) 261–278.

[42] R. Basili, M. T. Pazienza, F. M. Zanzotto, Modelling syntactic context in automatic term extraction, in: In Proc. of Recent Advances in Natural Language Processing (RANLP'01), Tzigov Chark, Citeseer, 2001.

[43] A. Peñas, F. Verdejo, J. Gonzalo, et al., Corpus-based terminology extraction applied to information access, in: Proceedings of Corpus Linguistics, Vol. 2001, 2001.

[44] T. M. Chung, P. Nation, Identifying technical vocabulary, System 32 (2) (2004) 251–263.

[45] C. Fox, A domain analysis and reuse environment.

[46] N. Niu, S. Easterbrook, Extracting and modeling product line functional requirements, in: International Requirements Engineering, 2008. RE'08. 16th IEEE, IEEE, 2008, pp. 155–164.

[47] N. Niu, S. Easterbrook, On-demand cluster analysis for product line functional requirements, in: Software Product Line Conference, 2008. SPLC'08. 12th International, IEEE, 2008, pp. 87–96.

[48] N. Weston, R. Chitchyan, A. Rashid, A framework for constructing semantically composable feature models from natural language requirements, in: Proceedings of the 13th International Software Product Line Conference, Carnegie Mellon University, 2009, pp. 211–220.

[49] I. John, Capturing product line information from legacy user documentation, in: Software Product Lines, Springer, 2006, pp. 127–159.

[50] H. Dumitru, M. Gibiec, N. Hariri, J. Cleland-Huang, B. Mobasher, C. Castro-Herrera, M. Mirakhorli, On-demand feature recommendations derived from mining public product descriptions, in: Software Engineering (ICSE), 2011 33rd International Conference on, IEEE, 2011, pp. 181–190.

[51] M. Acher, A. Cleve, G. Perrouin, P. Heymans, C. Vanbeneden, P. Collet, P. Lahire, On extracting feature models from product descriptions, in: Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems, ACM, 2012, pp. 45–54.

[52] G. Bécan, M. Acher, B. Baudry, S. Nasr, Breathing ontological knowledge into feature model synthesis: an empirical study, Empirical Software Engineering (2015) 1–48.
URL http://dx.doi.org/10.1007/s10664-014-9357-1

[53] M. Acher, A. Cleve, G. Perrouin, P. Heymans, C. Vanbeneden, P. Collet, P. Lahire, On extracting feature models from product descriptions, in: VaMoS'12, ACM, 2012, pp. 45–54.

[54] L. Yi, W. Zhang, H. Zhao, Z. Jin, H. Mei, Mining binary constraints in the construction of feature models, in: RE'12, IEEE, 2012, pp. 141–150.

[55] V. Alves, C. Schwanninger, L. Barbosa, A. Rashid, P. Sawyer, P. Rayson, C. Pohl, A. Rummler, An exploratory study of information retrieval techniques in domain analysis, in: Software Product Line Conference, 2008. SPLC'08. 12th International, IEEE, 2008, pp. 67–76.

[56] N. Niu, S. Easterbrook, Concept analysis for product line requirements, in: Proceedings of the 8th ACM international conference on Aspect-oriented software development, ACM, 2009, pp. 137–148.

Appendix A. Detailed Description of the C-NC Value Method

In this appendix, we present a detailed description of the C-NC Value method [27]. In particular, we provide an explanation of the computation of the metric and an algorithm describing the steps taken in the C-value method to construct a list of candidate terms from a corpus.

C Value. The C-Value [27] computes the frequency of a term and its subterms. There are two cases. First case: If a candidate term is a string of maximum length or is not found as nested, the C-Value is the result of its total frequency and its length. Second case: If it appears as part of longer candidate terms, then the C-Value will also consider its frequency as a nested term, as well as the number of these longer candidate terms. Given the candidate term t, the C-Value of t is [27]:

$$C - value(t) = \begin{cases} \log_2 |t| \cdot f(t) & \\ & \text{if } t \text{ is not nested,} \\ \log_2 |t| \cdot (f(t) - \frac{1}{P(T_t)} * \sum_{b \in T_t} f(b)) & \\ & \text{otherwise.} \end{cases}$$

where $|t|$ denotes its length, $f(t)$ is the frequency of t in the corpus, T_t is the set of terms that contains t, $P(T_t)$ is the number of candidate terms in T_t, and $\sum_{b \in T_t} f(b)$ is the sum of frequencies of all terms in T_t.

NC Value. The NC-Value measure [33] incorporates context information into the C-Value method for the extraction of terms. A context word is defined as a word appearing with the extracted candidate terms. We first identify the context words of the top list of candidates, and then compute the N-Value on the entire list of candidate terms. The higher the number of candidate terms in which a word occurs, the higher the likelihood that the word is related to terms. Hence it will exist with other terms in the same corpus. Formally, we reused the definitions of [33]. Given w as a context word, its weight will be: $weight(w) = \frac{t(w)}{n}$ where $t(w)$ is the number of candidate terms w appears with, and n is the total number of considered candidate terms; hence, the N-Value of the term t will be $\sum_{w \in C_t} f_t(w) * weight(w)$, where $f_t(w)$ is the frequency of w as a context word of t, and C_t is the set of distinct context words of the term t. The general score, NC-Value, is:

$$NCValue = \alpha * CValue(t) + \beta * NValue(t)$$

where, in our model, α and β are empirically set ($\alpha = 0.8$ and $\beta = 0.2$).

In the following, we describe the steps taken in the C-value method, proposed in [33], to construct a list of candidate terms from a corpus (see Algorithm 2).

Algorithm 2: C-Value method

```
tag the corpus;
extract strings using linguistic filter;
remove tags from strings;
remove strings below frequency threshold;
filter rest of strings through stop-list;
for all strings $a$ of maximum length do
    calculate $C - value(a) = \log_2|a| \cdot f(a)$;
    if $C - value(a) \geq Threshold$ then
        add $a$ to output list;
        for all substrings $b$ do
            revise $t(b)$;
            revise $c(b)$;
        end
    end
end
for all smaller strings $a$ in descending order do
    if $a$ appears for the first time then
        $C - value(a) = \log_2|a| \cdot f(a)$;
    else
        $C - value(a) = \log_2|a| \cdot (f(a) - \frac{1}{c(a)} t(a))$;
    end
    if $C - value(a) \geq Threshold$ then
        add $a$ to output list;
        for all substrings $b$ do
            revise $t(b)$;
            revise $c(b)$;
        end
    end
end
```

First, we tag the corpus and extract those expressions that satisfy the linguistic filter and frequency threshold. Then, we evaluate the C-value for each of the candidate expressions. The process begins with the longest expressions and finishes with the bigrams. The C-value for the longest terms will be assigned by the top branch of the first formula. We set a C-value threshold and only those expressions with C-value equal to or greater than this threshold are considered as candidate terms. To compute the C-value for a shorter expression, we also require its frequency as part of longer candidate terms and the number of these longer candidate terms. In the following, we explain how to get these two parameters:

For every extracted candidate term a, we create for each of its substrings b, a triple $(f(b); t(b); c(b))$, where $f(b)$ is the frequency of b, $t(b)$ is the frequency of b as a nested expression of candidate terms, $c(b)$ is the number of these longer candidate terms. Initially, $c(b) = 1$ and $t(b)$ equals the frequency of a. Every time b occurs after that, $t(b)$ and $c(b)$ are updated, while $f(b)$ remains the same. Indeed, each time b appears within a longer extracted candidate term a, $c(b)$ is incremented by 1 and $t(b)$ is increased by the frequency of a, $f(a)$. If $n(a)$ is the number of times a has been found as nested, then $t(b)$ will be increased by $f(a) - n(a)$.

Now, to evaluate the C-value for an expression a which is shorter by one word, we either already have for it a triple $(f(a); t(a); c(a))$ or we do not. If we do not, the C-value is given by the top branch of the formula. If we do, we consider the bottom branch of the formula. In that case, $P(T_a) = c(a)$ and $\sum_{b \in T_a} = t(a)$. After computing the C-value for expressions of length l, we evaluate the C-value for expressions of length $l - 1$; so that it is easy to know whether the expression to be processed has occurred as nested in longer candidate terms. Finally, we obtain a list of candidate terms ranked by their C-value.