MatrixMiner: A Red Pill to Architect Informal Product Descriptions in the Matrix

Sana Ben Nasr, Guillaume Bécan, Mathieu Acher, João Bosco Ferreira Filho, and Benoit Baudry
Inria - IRISA
University of Rennes 1, France

Nicolas Sannier
SNT Centre for Security, Reliability and Trust
University of Luxembourg

Jean-Marc Davril
University of Namur, FUNDP
Faculty of Computer Science
Namur, 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 condensed view of a product line. This paper introduces *MatrixMiner*: a tool for automatically synthesizing product comparison matrices (PCMs) from a set of product descriptions written in natural language. *MatrixMiner* is capable of identifying and organizing features and values in a PCM – despite the informality and absence of structure in the textual descriptions of products. Our empirical results of products mined from BestBuy show that the synthesized PCMs exhibit numerous quantitative, comparable information. Users can exploit *MatrixMiner* to visualize the matrix through a Web editor and review, refine, or complement the cell values thanks to the traceability with the original product descriptions and technical specifications.

Categories and Subject Descriptors

D.2.13 [**Software Engineering**]: Reusable Software; D.2.9 [**Software Engineering**]: Management—*Software configuration management*; D.2.1 [**Software Engineering**]: Requirements/Specifications—*Methodologies*

General Terms

Design, Management

Keywords

Software Product Lines, Variability Mining, Product Comparison Matrices

1. INTRODUCTION

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. The motivation for a customer is to choose the product that will exhibit adequate characteristics and support features of interest. In an organization, the identification of important features may help to transition to a product line; determine business advantage of some products as they hold specific features; or penetrate a new market.

Analyzing *manually* a set of related products is notoriously hard [7, 10]. There is a huge amount of scattered and informal data to collect, review, compare, and structure. 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, *MatrixMiner* provides automated techniques to synthesize *product comparison matrices (PCMs)*, i.e., tabular data that describe products along different features [4]. It enables the extraction and organization of information despite informality and absence of structure in the textual artifacts. Numerous tools have been implemented to mine variability [12, 14] and support domain analysis [2, 6–8, 10, 13], but none of them address the problem of structuring the information in a PCM. With the extraction of PCMs, organizations or individuals can obtain a synthetic, structured, and reusable model for the understanding of the differences 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.

MatrixMiner undertakes the underlying challenges behind the processing of informal and unstructured textual overviews. Our empirical results of thousands of products mined from BestBuy show that the synthesized PCMs exhibit numerous quantitative, comparable information. Users can then exploit *MatrixMiner* to visualize the matrix through a Web editor and review, refine, or complement the cell values thanks to a traceability with the original product overviews (texts) and technical specifications (roughly a list of features).

MatrixMiner targets domain analysts, software practitioners, customers, or organisations that want to build and maintain PCMs. Afterwards users can, from PCMs, (1) generate other domain models, such as feature models [7, 14]; (2) recommend features [10] (3) perform automatic reasoning (*e.g.* [9]); or (4) devise configurators or comparators;

ESEC/FSE'15, August 30 – September 4, 2015, Bergamo, Italy
© 2015 ACM. 978-1-4503-3675-8/15/08...
http://dx.doi.org/10.1145/2786805.2803180

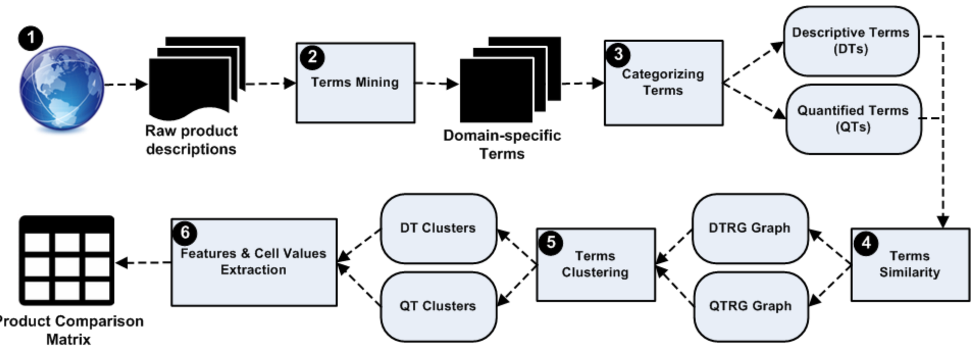


Figure 1: Extraction process; the resulting PCM can be visualized/edited in a Web environment (see Figure 2)

2. AUTOMATIC EXTRACTION

The extraction process of *MatrixMiner* is summarized in Figure 1 and consists of two primary phases. In the first phase, *domain specific terms* are extracted from a set of informal product descriptions (steps ❶ and ❷), while in the second phase the PCM is constructed (steps ❸ to ❻). 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 now outline the rest of the procedure.

Terms Mining. Step ❷ is based on a novel natural language processing approach, named *contrastive analysis* [5], for the extraction of *domain specific terms* from natural language documents. In this context, a term is a conceptually independent linguistic unit, which can be composed by a single word or by multiple words. A multi-word is conceptually independent if it occurs in different contexts (*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"). In particular, multi–words term extraction is carried out by identifying multi–word terms candidates in an automatically Part–Of–Speech (POS) tagged and lemmatized text, making use of different kinds of linguistic features. POS tagging is the assignment of a grammatical tag (e.g. noun, adjective, verb, etc.) to each word in the corpus. These candidates are then weighted with the C–NC value, currently considered as the state–of–the–art method for terminology extraction [5]. This metric establishes how much a multi-word is likely to be conceptually independent from the context in which it appears. The ranking of identified multi–words terms is then revised on the basis of a contrastive score calculated for the same terms.

Building the PCM. Once the top list of terms is identified for each product, we start the construction of the PCM: In step ❸ we divide the set of terms in two categories: quantified terms containing measures (*e.g.* "1920 x 1080 Resolution") including intervals (*e.g.* "Turbo Boost up to 3.1 GHz"); and descriptive terms containing noun phrases and adjectival phrases (*e.g.* "Multiformat Media Reader"). The key idea is to perform separately descriptive terms (DTs) clustering from quantified terms (QTs) clustering. A DTs cluster gives the possible descriptor values (*e.g.* "Multiformat") while a QTs cluster provides the potential quantifier values (*e.g.* "1920 x 1080") for the retrieved feature. In step ❹ we compute terms similarity to generate a weighted terms relationship graph for each category. To identify coherent clusters, we first determine the similarity of each pair of terms by using syntactical heuristic. In step ❺ we apply terms clustering in each graph to identify descriptive terms clusters and quantified terms clusters. The underlying idea is that a cluster of tight-related terms with different granularities can be generated by changing the clustering threshold value [6]. Finally, step ❻ extracts features and cell values to build the PCM. To extract the feature name from a cluster, we developed a process that involves selecting the most frequently occurring phrase from among all of the terms in the cluster. This approach is similar to the method presented in [11] for summarizing customer reviews. For example, "1920 x 1080 Resolution" and "1366 x 768 Resolution" represent QTs cluster that gives "Resolution" as a features name and two potential values: "1920 x 1080" and "1366 x 768". Terms which are not clustered will be considered as boolean features. Finally we distinguish different types of features (see Figure 2): *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"), and *empty* values. The resulting PCM can be visualized and refined afterwards (see next section).

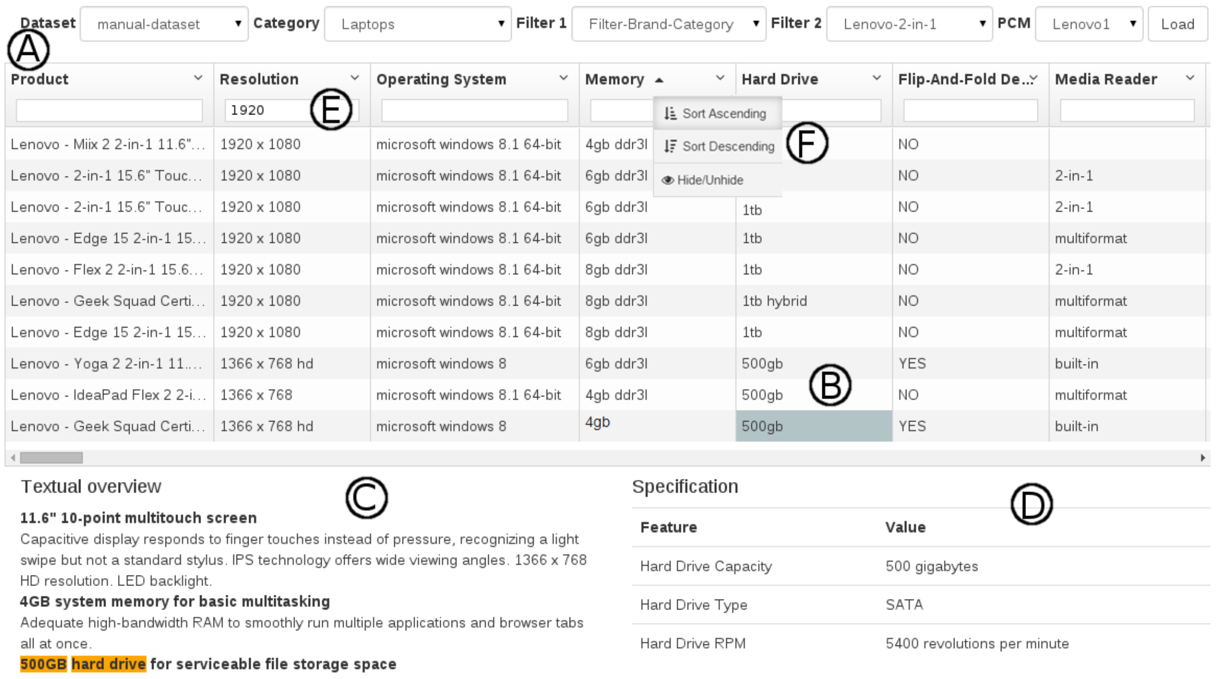
3. MATRIXMINER

MatrixMiner offers an interactive mode where the user can import a set of product descriptions, synthesize a complete PCM, and exploit the result. We have pre-computed a series of PCMs coming from different categories of BestBuy [15] (Printers, Cell phones, Digital SLR Cameras, Laptops, TVs, Washing Machines, Ranges, etc.). 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. *MatrixMiner*[1] is available online.

3.1 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 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),

[1]http://matrix-miner.variability.io
[2]http://nlp.stanford.edu



Dataset: manual-dataset | Category: Laptops | Filter 1: Filter-Brand-Category | Filter 2: Lenovo-2-in-1 | PCM: Lenovo1

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 2: The editor of *MatrixMiner* in action

such as noun, verb, adjective, etc. To tokenize and remove stop words from text we use Lucene[3] which is a widely used Information Retrieval (IR) library. *Levenshtein* computes syntactical similarity based on words' morphology. It comes from the the Simmetrics[4] library. The specific source code of the extraction procedure is available online[5]. Our Web environment reuses the editor of OpenCompare[6].

3.2 Empirical Results

We evaluate our proposal against numerous categories of products mined from BestBuy, which provides descriptions for hundreds of thousands of products, including: (1) *technical specifications*, which describe the technical characteristics of products through feature lists; (2) *products overviews*, texts describing features of products using natural language. We mined 2692 raw product overviews. Currently, we have implemented a mining procedure on top of BestBuy API [15] for retrieving numerous product pages along different categories. Only the extraction phase is specific to BestBuy. Another extraction engine is required like we did for Wikipedia [4] or any other sources.

Given a set of textual product overviews, we want to synthesize all the gathered information into a PCM. We empirically observed that, given a supervised scoping (selection of products), the synthesized PCMs exhibit numerous quantitative and comparable information. We reported that PCMs of 10 products comprise in average 12.5% of quantified features and 15.6% of descriptive features. We also noticed that a significant portion of features (49.7%) and cell values (26.2%) is recovered in the technical specifications, showing the usefulness of our approach. With our automated extraction from overviews, our qualitative review shows that there is also a potential to complement technical specifications of products: "Flip-and-Fold Design" is a boolean feature in the overview PCM of Laptops but it does not exist in the specification PCM. Our synthesized overviews PCMs can even refine specifications, *e.g.* "Media Reader" has three possible values ("digital", "multiformat" and "2-in-1", see Figure 2) in the overview PCM. However it is simply a boolean feature in the specification PCM. This tool provides a snapshot for the product descriptions. Though products may change at a fast pace, changes are not that dynamic as changes appear more in a matter of weeks or months.

3.3 Importing, Visualizing, and Editing

The empirical insights drive the design of the *MatrixMiner* environment dedicated to the visualisation and edition of PCMs. The results indeed suggest that automation has a great potential but also some limitations. Human intervention is beneficial to (1) refine/correct some values (2) reorganize 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 2, Ⓐ);
- 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 2, Ⓑ and Ⓒ);
- ways to visualize the PCM with a traceability with the technical specification (see Figure 2, Ⓓ). For each cell

[3]`https://lucene.apache.org`
[4]`http://sourceforge.net/projects/simmetrics`
[5]`https://github.com/sbennasr/matrix-miner-engine`
[6]`https://github.com/gbecan/OpenCompare`

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;

- 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 2, Ⓔ 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 (one cannot objectively know the preferred feature of a user), *etc.*

4. RELATED WORK

Numerous techniques for synthesising feature models have been proposed (*e.g.* [1, 3, 7, 14]). *MatrixMiner* can be used to fed them. The closest work is by Davril *et al.* [7] who presented an automated approach for constructing 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 [10]. Then the synthesis is performed from a PCM manually elaborated. *MatrixMiner* can be used to speed up and improve the elaboration of the matrix. An open problem for feature model synthesis is how to deal with numerical, unknown, or empty values contained in PCMs.

Nadi *et al.* [12] developed a comprehensive infrastructure to extract configuration constraints automatically from C code. Alves *et al.* [2], Niu *et al.* [13], Weston *et al.* [16], and Chen *et al.* [6] applied information retrieval techniques to abstract requirements from existing specifications, typically expressed in natural language. Weston *et al.* [16] provided a tool framework *ArborCraft* that automatically processes natural-language requirements documents into a candidate feature model, which can be refined by the requirements engineer. Our goal is to organize the variability information in a comparable way (through a PCM), hence raising specific challenges. Closest to our work is Ferrari *et al.* [8] who applied natural language processing techniques to mine commonalities and variabilities from brochures. In our context a notable difference is that (1) the textual corpus is smaller, *i.e.*, overviews are short texts while brochures per vendors typically exhibit numerous documents, pages and words; (2) we aim to structure features and extract corresponding values in a comparison matrix; (3) to keep the traceability of the synthesized PCM with the original product descriptions and technical specification for further refinement or maintenance by users.

5. CONCLUSION

We presented *MatrixMiner*, a Web environment with an interactive support for automatically synthesizing *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. Our empirical evaluation on BestBuy products showed that the synthesized PCMs exhibit numerous quantitative, comparable information; yet users may need to complement or even refine cell values. For this reason, *MatrixMiner* also provides the ability to *tracing* products and features of a PCM 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.

6. REFERENCES

[1] M. Acher, A. Cleve, G. Perrouin, P. Heymans, C. Vanbeneden, P. Collet, and P. Lahire. On extracting feature models from product descriptions. In *VaMoS'12*. ACM, 2012.

[2] V. Alves, C. Schwanninger, L. Barbosa, A. Rashid, P. Sawyer, P. Rayson, C. Pohl, and A. Rummler. An exploratory study of information retrieval techniques in domain analysis. In *SPLC*, 2008.

[3] G. Bécan, M. Acher, B. Baudry, and S. Nasr. Breathing ontological knowledge into feature model synthesis: an empirical study. *ESE*, 2015.

[4] G. Bécan, N. Sannier, M. Acher, O. Barais, A. Blouin, and B. Baudry. Automating the formalization of product comparison matrices. In *ASE*. ACM, 2014.

[5] F. Bonin, F. Dell'Orletta, G. Venturi, and S. Montemagni. A contrastive approach to multi-word term extraction from domain corpora. In *LREC*, 2010.

[6] K. Chen, W. Zhang, H. Zhao, and H. Mei. An approach to constructing feature models based on requirements clustering. In *RE*, 2005.

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

[8] A. Ferrari, G. O. Spagnolo, and F. dell'Orletta. Mining commonalities and variabilities from natural language documents. In *SPLC*, 2013.

[9] J. Guo, E. Zulkoski, R. Olaechea, D. Rayside, K. Czarnecki, S. Apel, and J. M. Atlee. Scaling exact multi-objective combinatorial optimization by parallelization. In *ASE*, 2014.

[10] N. Hariri, C. Castro-Herrera, M. Mirakhorli, J. Cleland-Huang, and B. Mobasher. Supporting domain analysis through mining and recommending features from online product listings. *IEEE Transactions on Software Engineering*, 99:1, 2013.

[11] M. Hu and B. Liu. Mining and summarizing customer reviews. In *KDD*, 2004.

[12] S. Nadi, T. Berger, C. Kästner, and K. Czarnecki. Mining configuration constraints: Static analyses and empirical results. In *ICSE*, 2014.

[13] N. Niu and S. M. Easterbrook. Concept analysis for product line requirements. In *AOSD*, 2009.

[14] S. She, U. Ryssel, N. Andersen, A. Wasowski, and K. Czarnecki. Efficient synthesis of feature models. *Information & Software Technology*, 56(9), 2014.

[15] `http://www.bestbuy.com`. Bestbuy, 2014.

[16] N. Weston, R. Chitchyan, and A. Rashid. A framework for constructing semantically composable feature models from natural language requirements. In *SPLC*, 2009.