Universidade Federal do Amazonas

Instituto de Computação

Programa de Pós-Graduação em Informática

Learning to Recommend Similar Alternative Products in e-Commerce Catalogs

Urique Hoffmann de Souza Almeida

Manaus, Brasil

Junho, 2016

Urique Hoffmann de Souza Almeida

Learning to Recommend Similar Alternative Products in e-Commerce Catalogs

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática, Instituto de Computação - IComp, da Universidade Federal do Amazonas, como parte dos requisitos necessários à obtenção do título de Mestre em Informática

Universidade Federal do Amazonas

Instituto de Computação

Programa de Pós-Graduação em Informática

Orientador: Altigran Soares da Silva

Manaus, Brasil

Junho, 2016

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

A447l

Almeida, Urique Hoffmann de Souza
Learning to Recommend Similar Alternative Products in e-Commerce Catalogs / Urique Hoffmann de Souza Almeida. 2016
67 f.: il. color; 31 cm.

Orientador: Altigran Soares da Silva
Dissertação (Mestrado em Informática) - Universidade Federal do Amazonas.

1. Product Comparison Functions. 2. E-Commerce. 3. Sistemas de Recomendação. 4. Programação Genética. I. Silva, Altigran Soares da II. Universidade Federal do Amazonas III. Título

Urique Hoffmann de Souza Almeida

Learning to Recommend Similar Alternative Products in e-Commerce Catalogs

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática, Instituto de Computação - IComp, da Universidade Federal do Amazonas, como parte dos requisitos necessários à obtenção do título de Mestre em Informática

Trabalho Aprovado em. Manaus, Brasil, 15 de Junho de 2016:

Manaus, Brasil
Junho, 2016

I dedicate this work to the person who is responsible for everything I am, my mother Auriete de Araújo Souza.

Acknowledgements

First of all I thank GOD for giving me the opportunity to achieve this dream of having a master degree. Thank you my Lord, my King, my God. My God never fails.

I thank my mother Auriete de Araújo Souza for fighting with me for my dreams and spare no effort to help me. My queen, all I am and everything I have today I owe to this wonderful person she is, I can not express in words how grateful I am for everything you did, does and will do for me.

I thank my classmates for everything we passed through together.

I thank my professors Altigran Soares da Silva and Moisés Gomes de Carvalho for teaching me, for pushing me, for helping me, for having patience with me, for trusting in me. Without them, nothing of this would be possible.

Finally, I thank everyone who somehow helped me in this process.

Thank you so much!

'I have fought the good fight,
I have finished the race,
I have kept the faith'
(Holy Bible, 2 Timothy 4, 7)

Resumo

Nesse trabalho, descrevemos um novo método que projetamos, implementamos e testamos para a tarefa de encontrar produtos que são alternativas similares a um dado produto em um catálogo de um site de comércio eletrônico. Nesse trabalho, consideramos como alternativas similares produtos que, apesar de não serem idênticos a um produto de interesse, têm características que os tornam boas alternativas a esse produto. Nossa motivação para esse trabalho é poder recomendar produtos similares com base apenas nas suas características, sem a necessidade da utilização do histórico de compras dos usuários. Assim, nesse trabalho lidamos com o chamado problema de *cold start*, que é comumente encontrado em abordagens de recomendação, e que pode levar a perda de lucro em sites de comércio eletrônico. Nosso método, chamado *GPClerk*, utiliza Programação Genética (GP) para aprender funções que comparam dois produtos, e dizem se estes são similares ou não. Essas funções são chamadas nesse trabalho de *product comparison functions*. Para tornar nosso método viável em um cenário típico de comércio eletrônico, propomos também uma estratégia não supervisionada para gerar exemplos de treino a serem utilizados no processo de aprendizagem. Resultados de experimentos que executamos e descrevemos nessa dissertação indicam que nosso método é capaz de gerar funções adequadas, e que nossa estratégia para geração automática de dados de treino é efetiva para essa tarefa.

Palavras-chave: Product Comparison Functions, E-Commerce, Sistemas de Recomendação, Programação Genética.

Abstract

In this work, we describe a novel method we designed, implemented and tested to finding products that are *similar alternatives* to a given product in the catalog of an e-commerce site. By similar alternatives, we mean products that, although are not identical to a product of interest, have features that make them suitable alternatives for customers that look for it. Our motivation is to enable the recommendation of alternative products based solely on the product's features, without relying on historical purchase data. By doing so, we address the so-called cold start problem, which is often found in product recommendation approaches, and that may lead to profit loss in e-commerce sites. Our method, we call *GPClerk*, uses Genetic Programming (GP) to learn functions for comparing two products and telling whether two products are similar alternatives or not. These functions are termed here as *product comparison functions*. To make our method feasible in typical e-commerce settings, we also propose an unsupervised strategy to generate training examples to be used in the learning process. Results of experiments we carried out and report here indicate that our method is capable of generating suitable product comparison functions and that our strategy for automatically generating training data is effective for this task.

Keywords: Product Comparison Functions, E-Commerce, Recommender Systems, Genetic Programming.

List of Figures

List of Tables

List of abbreviations and acronyms

GP	Genetic Programming
GPClerk	Name given for our proposed method
UVP	User-Viewed Product Pairs
HCP	Highly Co-related Products Pairs
PEFEvol	Name given for our proposed evolutionary algorithm
NPC	Naive Product Comparison
CAM	Category of Cameras
REF	Category of Refrigerators
LAP	Category of Laptops
SMP	Category of Smartphones
TV	Category of Tvs
R	Relevance Filter
S	Semantic Filter

List of symbols

$\in$	Set Membership
$\subseteq$	Subset
$\cup$	Set Union
$\cap$	Set Intersection
$<$	Strict Inequality
$>$	Proper Subgroup
$\leq$	Inequality
$\geq$	Subgroup
$\forall$	Universal Quantification
$\sum$	Summation
α	Alpha
τ	Tau
μ	Mu
σ	Sigma
β	Beta
ℓ	Ell
ϵ	Epsilon
$\mathbb{R}$	Real Numbers Set
$\mathcal{T}$	Set of All Training Pairs
$\mathcal{T}^+$	Set of Negative Training Pairs
$\mathcal{T}^-$	Set of Positive Training Pairs
$\mathcal{P}^+$	Set of Positive Pairs
$\mathcal{P}^-$	Set of Negative Pairs

$\mathcal{A}$	Attributes Set
$\mathcal{S}$	Product Comparison Function
$\mathcal{N}$	Set of New Individuals
$\mathcal{B}$	Set of Best Individuals
S	Set of All Similar Product Pairs
UV	Set of User-Viewed Product Pairs
P	Set of Products
P^2	Set of All Product Pairs
HC	Set of Positive Examples
NC	Set of Negative Examples
C	Product Category

Contents

1 Introduction

Recommender systems are used by most e-commerce sites to suggest products to users and provide additional information to help customers to decide which products to acquire. Products can be recommended based on several different types of information, such as top overall sellers on a site (LONG et al., 2012), customer's demographics, customer's past buying behaviour (HAMMAR; KARLSSON; NILSSON, 2013), product attributes (SCHAFER; KONSTAN; RIEDL, 2001; KAGIE; WEZEL; GROENEN, 2008b), e.g., technical specifications, general characteristics, brand, etc. Recommendations based on this last kind of information are called *content-based* or *knowledge-based* recommendations (BURKE, 2000).

A compelling form of content-based recommendation is to present the user that is looking for a particular product other products that are *similar* to it, with respect to its features. This form of recommendation is useful, for instance, when a costumer explicitly wants to find products with certain characteristics or when the seller wants to present alternatives to a product the customer is interested in, e.g., for the sake of comparison or to provide options to out-of-stock items. Although relatively new, this problem is already attracting attention both from the industry (KATUKURI et al., 2014) and from the academia (MCAULEY; PANDEY; LESKOVEC, 2015).

An interesting aspect of this kind of recommendation is that it enables suggesting products to the customers without relying on historical data. It means that the system can recommend products and provide buying options even if a customer is new to the system or if the product is new to the catalog. In fact, even for products that are not new, traditional recommendation techniques are suitable only for popular items, and recommendation error increases quickly with the decrease of popularity of the item. Recent research indicate that recommendation for infrequent items is especially important for e-commerce sites since they usually embrace relatively large marginal profit (YIN et al., 2012). This problem is referred to as cold start in the literature. It is prevalent in almost all recommender systems, and most existing approaches suffer from it (SCHEIN et al., 2002).

In typical e-commerce sites, to look for similar alternative products may require the user to browse manually through a large number of products. For instance, suppose a user is interested in a particular camera, say, "Nikon S3500". Currently, if this user wants to find alternative cameras that are similar to this model (i.e., having similar features), for the sake of comparing their prices, it is likely that she would have to browse through hundreds of other cameras in some site's catalog to find them. On the

Attribute	*Nikon S3500*	*Sony W830*
Brand	Nikon	Sony
Type of Camera	Compact	Digital Camera
Monitor/Display	2,7" LCD / TFT 230.000	2.7"-LCD TFT-Clear Photo LCD
Resolution	20,1	20,1
Internal Memory	25MB	27MB
Memory Cards	Yes	Yes
Compatible Memory Cards	SD, SDHC and SDXC	Stick Duo, Stick PRO Duo (High Speed)
Sensor	CCD 1/2, 3 inch.	Super HAD CCD
Optical Zoom	7x	8x
Digital Zoom	4x	32x
Lenses	Crystal NIKKOR 26-182mm fixed	-
Shutter Speed	1/2000 - 1 s 4 s	-
Focus range	[W]: Aprox. 50 cm/[T]: Aprox. 1 m . . .	-
Opening	f/3.4-6.4	-
Flash Modes	Automatic TTL Flash with pre-flash monitor	Auto/On/Off/ Slow Syncro / Flash Extended
Flash range	[T]:1,0 to 2,1m (3 feet 4 inch. to 7 feet 1 inch.) . . .	ISO Auto: Aprox. 0.3m to 2.8m
Battery Type	Rechargeable Li-ion Battery EN-EL19	Battery Charger Adapter, Power Cable
Video Features	Full HD: 1920px1080p/30 / HD: 1280px720p/30 . . .	-
Scene modes	Backlight,. . .,Sports, Sunset	Sensitivity/Twilight/. . ./Pets
File Formats	.avi,.jpg,.wav	JPEG
Built-in microphone	Yes	-
Tripod mount	Yes	-
Menu Languages	Chinese,Danish,. . ., Arabic	-
Color	Purple	Violet
Dimensions (HxWxD)	5,7x9,6x2cm	9,31x5,25x2,25cm
Weight	129g	120g

Table 1 – Attributes for two camera models with their values (some values were truncated to save space).

other hand, if this camera is not in stock, it would be interesting to provide the user with similar alternative cameras in stock, without having her to look over the whole catalog.

To find whether two products are similar or not, it is necessary to compare their *attributes*. This can be challenging, or even unfeasible, to be carried out manually by casual users on the Web. For instance, in a certain e-commerce site, to verify whether the "Nikon S3500" camera is similar to another camera, say the "Sony W830", a user has the option of comparing the 26 attributes provided for the first camera with the corresponding attributes of the second cameras. The lists of attributes available for each camera in a real site we used in our experiments are presented in Table 1. Notice that the second camera has only 18 attributes. Also, notice that many attributes are difficult to be compared, unless the user is an expert or knows the existent lingo related to the field.

In general, the same situation occurs in many other categories, that is, finding, in a catalog, products that are similar to a given product may require comparing tens of attributes from thousands of products pairs. For each pair of products of a given category, comparing its attributes may be hard, since some of the attributes may have very specific semantics. Moreover, the casual user may not be aware of the relevance of some attributes when comparing products on-line.

To deal with this problem, in this work we describe a novel method we designed, implemented and tested to finding products that are similar alternatives to a given product. We assume that we are given a set of products from a same category of a same on-line store, along with their attributes. Our method uses Genetic Programming (GP) (KOZA, 1992) to learn functions for comparing two products based on their attributes, so to tell whether the two products are similar or not. These functions are called here *product comparison functions*. Notice that determining if a product can be a similar alternative for another product is intrinsically dependent on the users' judgement. We thus claim that machine learning techniques such as GP are suitable for capturing such a notion from user provided examples. Intuitively, we picture our method as playing the role of a savvy store clerk, that is able to recommend alternative products to a costumer. Thus, we call our method *GPClerk*.

GP is a well-known optimization and machine learning technique that has been successfully applied for several problems that involve learning functions for comparing multi-attribute objects. In particular, recent GP-based methods have been proposed for the problem of record deduplication (CARVALHO et al., 2006; CARVALHO et al., 2012; ISELE; BIZER, 2012), which is related, although different, from the problem we address here. This motivated us to choose GP among many other possible machine learning techniques.

In our case, GPClerk relies on a set of attribute-specific similarity functions to compare the values of attributes in two given products. These similarity functions are automatically selected according to the attributes that are common to the two products. A suitable product comparison function must be able of properly combine and weight the scores these functions obtain, so to produce a single score that measures how similar the two products are. We then propose a GP-based algorithm to evolute such functions from sets of training pairs of products that can be considered as similar alternatives and pairs of products that cannot be considered as so.

A critical issue in any learning method is how to obtain the training examples. This is particularly problematic in our case, since asking users to find examples of pairs of similar alternative products to provide as examples might be unfeasible, due to the potentially high number of pairs and to the possible degree of expertise required to compare values of some more technical attributes (e.g., the shutter speed of a camera). To overcome this, we propose an unsupervised strategy to find training examples by mining the logs of the web site to find pairs of products of a same category that are frequently viewed together in same user section on the site. This information is easy to be obtained and it is, in fact, used for other types of product recommendation methods, such as the well-known collaborative filtering methods.

To evaluate our GPClerk, we carried out several experiments using real datasets on 5 different popular product categories, involving more than a thousand products, and nearly 200000 potential product pairs. The results indicate our method is able to correctly predict whether two given products are similar alternative or not based on the values of their attributes. Indeed, the product comparison functions it generates achieved, on average, above 0.8 in terms of F-measure. The results also indicated that the strategy we proposed to find training examples is an effective alternative for obtaining such examples in an unsupervised way.

The remainder of this thesis is organized as follows. In Chapter 2 we review related work. In Chapter 3 we present an overview of our method. In Chapter 4 we present our first contribution in this work, an attribute taxonomy for four different classes of common attributes present in products of e-commerce Web sites and their respective attribute-specific similarity functions. In Chapter 5 we describe our second contribution in this work, the evolutionary algorithm we designed to generate suitable product comparison functions. In Chapter 6 we detail our third contribution, a strategy to obtain training examples for this algorithm in an unsupervised way. In Chapter 7, we report experiments we carried out to evaluate our method and its results. In Chapter 8 we present our conclusions and directions for future work.

2 Related Work

In this chapter we overview recent works we considered as related to ours in two aspects. First, we discuss methods presented in the literature that face the problem of finding similar products. Next, we cover methods that apply GP to the problem of record deduplication. These methods were influential to us in choosing GP as the basis of our method.

2.1 Finding Similar Products

Over the years, there has been a lot of attention in the literature to the problem of finding the *same* product occurring in various e-commerce sites, for instance, for the sake of price comparison (BILENKO; BASU; SAHAMI, 2005; GOPALAKRISHNAN et al., 2012; NGUYEN et al., 2011; KÖPCKE et al., 2012). In most case, several titles of product offers are collected from distinct merchants, and the goal is to cluster those offers that correspond to the same product. Using different strategies, methods such as those described in (GOPALAKRISHNAN et al., 2012), (NGUYEN et al., 2011), and (KÖPCKE et al., 2012), try to spot in the title, strings that can be used as key identifiers for the product, such as product codes, model part numbers, or universal identifier such as UPCs. These identifiers then are used for clustering. A different approach is taken in (BILENKO; BASU; SAHAMI, 2005), where a similarity function is trained by inducing a linear combination of basis similarity functions that compare attributes of different products. Thus, the authors consider, as we do, that the product catalog, with attributes and their values, is available.

Although important and challenging, this problem is different from the one we tackle here, since we do not want to spot the same product, but different products that are similar to a given product. The problem we addressed here is relatively new and, to best of our knowledge, has been studied in only a few previous work in the literature (KAGIE; WEZEL; GROENEN, 2008b; KAGIE; WEZEL; GROENEN, 2008a; KATUKURI et al., 2014; KöNIK; MUKHERJEE; KATUKURI, 2015; MCAULEY; PANDEY; LESKOVEC, 2015).

In (KAGIE; WEZEL; GROENEN, 2008b), Kagie et. al. proposed a content-based graphical shopping interface based on product attributes to recommend similar products. To use this interface, the user must first define an *ideal* product by providing desired values to its attributes. The interface then shows products considered as similar to this ideal product in a 2D Map. By interacting with this map, the user chooses, from the products plotted, the most similar to the ideal. The interface then recalculates the similarity between the ideal product to all other products in the dataset. This process continues until the interface shows a product the user considers as the most similar. In this work the authors consider only two of attribute classes: categorical and numeric. In (KAGIE; WEZEL; GROENEN, 2008a), the same authors improve their similarity calculation by proposing a Poisson regression model to determine attribute weights. Our method differs from this in two main aspects. First, in GPCleark, the user does not need to specify an ideal product. Second, we do not require the user to actively interact with the system to find similar products.

In (KATUKURI et al., 2014), the authors present a similarity-based recommendation system designed to the e-Bay web site. This system works in two steps. First, an offline algorithm groups frequent user queries based on common features of the items that return from those queries. This results in a dictionary of clusters definitions. The goal of this algorithm is to capture item similarity dimensions that the users consider as important. Second, a runtime component retrieves the best matching clusters for a given a seed item. From these clusters, this component construct a small set of items that are similar to the seed item in dimensions considered as important. This approach makes use of features extracted from items description, mainly the title. Also, it may require the user to enter additional attribute-value pairs to enrich the description of the items. The paper does not present experimental results with direct measures of quality. Instead, it reports metrics showing improvements in user engagement and revenue in comparison to a previous system they used. In (KöNIK; MUKHERJEE; KATUKURI, 2015), the same authors propose a new algorithm that improves the system by personalizing the recommendations according to user intentions.

In (MCAULEY; PANDEY; LESKOVEC, 2015), the authors describe a system named *Sceptre*, whose main goal is to find, for a given product, other related products, specifically *complementary* and *substitutable* products. Intuitively, the concept of *substitutable* product, resembles the notion of similar alternative product used in our work. To accomplish this, Sceptre first builds a graph of related products based on log information, that is, products bought or views by the same user, products bought together, etc. Then, the system iteratively mines features from the text of reviews, using topic modelling, to build a vector representing each product, and applies supervised link prediction to predict links between products that denote the semantics of some relationship, that is, complementarity or substitutability. During the process, the product representations are continually refined to find best features, so to improve prediction, until convergence is reached. Finally, the resulting graph is used to recommend a list of the most related, i.e., complementary or substitutable, products to a given product.

Differently from the systems described in (KATUKURI et al., 2014) and (MCAULEY; PANDEY; LESKOVEC, 2015), which rely on features extracted from textual sources, GPClerk relies on data available in a product catalog, which contains the specifications of the products. For instance, in the case of (KATUKURI et al., 2014), the system is designed to the e-Bay scenario, in which only item descriptions are available. This is reason why in (KATUKURI et al., 2014), the set of feature extracted from the description sometimes needs to be expanded by the user.

Also, both in (KATUKURI et al., 2014) and (MCAULEY; PANDEY; LESKOVEC, 2015), the recommendation is made from structures build offline, i.e., as set of products clusters in the case of (KATUKURI et al., 2014), and graph representing predict product relationships in (MCAULEY; PANDEY; LESKOVEC, 2015). This means that these structures must be re-generated from times to times to include new added items. In GPClerk, product comparison functions relies on data available in a product catalog, and this is the only required information for making recommendations for new added products. It means that, once we have the product comparison functions, having only the information from the attributes of the new added products, GPClerk can naturally make recommendations for them.

2.2 GP Methods for Record Deduplication

Our solution to the problem of finding similar alternative products is inspired by previous works that successfully applied GP to the problem of finding record duplicates based on attributes values (CARVALHO et al., 2012; ISELE; BIZER, 2012). Thus, in the following we review recent representative methods based on such an approach.

In (CARVALHO et al., 2012), the authors propose a GP-based approach for record deduplication. Similarly to GPClerk, their method takes as input training examples and generates record deduplication functions, using an evolutionary algorithm. This methods is also similar to GPClerk in the sense that the record deduplication functions combine and weights several attribute similarity functions. However, differently from our method, only string and textual similarity functions are considered. In GPClerk, a set of similarity functions suitable for comparing typical attributes used for describing e-commerce products is adopted. Indeed, as described in Chapter 4, GPClerk deals with many forms of textual and numerical attributes. Another interesting difference on GPClerk is that it takes advantage of the e-commerce setting to automatically find examples to be provided to the evolutionary algorithm. Thus, GPClerk runs in unsupervised way.

In (ISELE; BIZER, 2012), the authors describe *GenLink*, a supervised learning algorithm which employs GP in order to generate rules that specify the conditions that two entities must fulfill in order to be considered the same real-world object. As it happens in (CARVALHO et al., 2012), GenLink takes as input examples of pairs that identify the same real world object. However, instead of a function, GenLink generates rules that combine different kind of operators, i.e., value operators and similarity operators, to find the same entity. Similarly to (CARVALHO et al., 2012), Genlink is also supervised.

3 Overview

In this chapter we present an overview of our approach for dealing with the problem of finding similar alternative products in a catalog.

3.1 Problem Statement

Consider a set P of products of some category C in the catalog of an e-commerce Web site. Assume that each product P has a value for each attribute of a set $\mathcal{A}(C)=\{A_1,\ldots,A_n\}$, that is, the set of attributes common to products of this category. The value of some of these attributes may be *null* for some products. For instance, Table 1 shows an example of two products and the values they have for each attribute of the Camera category.

Let $P^2=P\times P$, be the set of all possible pairs of products in C. We assume that there is a subset $S\subseteq P^2$ whose elements are pairs of *similar alternative* products.

Definition 1 *We say that two products x and y of some category C are* ***similar alternatives*** *if the products are evaluated by costumers as alternatives for buying, given their features.*

Intuitively, a consumer who wants to buy x could buy y as well instead, if, for instance, y is cheaper, or x is out-of-stock, etc.

In Figure 1 we illustrate pairs of similar alternative products from two distinct categories. We took these pairs from a set of real similar alternative product pairs identified by users in our experiments. Notice that, for some attributes, which are highlighted in the figure, the values are not the same in the two similar alternative products. This is the case of **Brand** in the Notebook category. In the case of numerical attributes, values may be very divergent, as it is the case of **Memory** in the Smartphones categories. This suggests that not all attributes have the same importance for the consumer when comparing attribute values to select similar alternative products.

We stress that the definition of similar alternative products is rather fuzzy, and it is completely dependent on the consumers' judgement. Such a fuzzy property makes the task of finding similar alternative products a challenging one.

Our goal is to find a function $\mathcal{S}:P^2\rightarrow\mathbb{R}$, where $\mathcal{S}(x,y)\geq\tau$, if $\langle x,y\rangle\in S$, otherwise $\mathcal{S}(x,y)<\tau$, where τ is a threshold value that separates pairs of similar alternative products from the remaining pairs. Our motivation for this is twofold. First, we aim at enabling costumers to find in the catalog products that are similar alternatives to a

Attribute	Notebook 1	Notebook 2
RAM	4GB	4GB
Cache	3MB L3 Cache	3MB L3 Cache
Drives	DVD	DVD
Dimensions	2,5x25,3x38,1cm	2,7x 24,5x34,2 cm
Motherboard	Intel® HM77 Express	Intel® HM77 Express
HD	500GB	500GB
Screen	15,6"	14"
Brand	Gateway	Acer
OS	Windows 8	Windows 8
Video	Intel® HD Graphics	Intel® HD Graphics
Collor	Black	Black
Processor	Intel Core i5	Intel Core i5
Chipset	Intel® HM77 Express	Intel® HM77 Express
Bus	1333 MHz	1333 MHz

Attribute	Smartphone 1	Smartphone 2
GPRS	No	No
Memory Cards	micro SD 4GB	Micro SD 4GB
Calendar	yes	yes
Memory	256MB	128MB
Sound	MP3, FM radio	MP3, MP4
Brand	Multilaser	Multilaser
Keyboard	Qwerty	Qwerty
Band	Quadriband	GSM
Color	White/Rose	White/Rose
Dimensions	11,15x4,6x1,5 cm	11x5,6x1,8cm
Camera	0.3MP	1.3MP
Bluetooth	Yes	Yes

Figure 1 – Pairs of similar alternative products identified by users in our experiments. For simplifying the presentation, only a subset of the actual set of attributes of each category is illustrated.

given product. Second, we want to make this task feasible to be carried out in cases were the catalog is large.

3.2 General Approach

We propose that the product comparison function $\mathcal{S}(x, y)$ can be computed by comparing the values of each attribute for products x and y, and combining results of comparing each attribute to reach a conclusion on how similar x and y are. More precisely, let s_k be a function that computes how close are the values of attribute A_k in x and y. We look for a function $\mathcal{S}(x, y)$ that combines functions $s_1, \ldots, s_n$ to produce an *comparison score value*.

For each attribute, the comparison of their values in different products is carried out using a specific similarity function, which depends on the class of the attribute,e.g., if the attribute is numerical, categorical, etc. In Chapter 4 we describe the attribute-specific similarity function we adopted in our work.

As in the case of other problems in which the solution is heavily dependent on the users' judgement, we claim that using a machine learning technique to induce the function $\mathcal{S}$ from a few examples given by users is an interesting alternative.

We propose and experiment using Genetic Programming (GP) (KOZA, 1992) to induce function $\mathcal{S}$ based on examples of similar alternative products provided by users. GP is a well-known optimization and machine learning technique that has been successfully applied for several problems that involve learning functions for comparing multi-attribute objects. In particular, recent GP-based methods have been proposed for the problem of record deduplication (CARVALHO et al., 2012; ISELE; BIZER, 2012), which is related, although different, from the problem we address here.

Considering the application scenario of an e-commerce Web site, identifying and recruiting some costumers and have them finding some pairs of similar alternative products in a large catalog is hardly viable. Moreover, such cumbersome procedure would have to be repeated for each site and each category. To avoid it, we propose an alternative strategy we describe below.

3.3 Obtaining Training Data

Considering a product category C and the catalog of an e-commerce web site. Consider also a *log* of user's activities in this site, which is divide into user sections. Let $V_{i,j}$ be the set of products visited by a user U_i during some user session S_j on the site. Thus, the set $V_{i,j}^2 = V_{i,j} \times V_{i,j}$ is the set of all pairs of products that were visited by a same user U_i in the same session S_j. We define $UV = \bigcup_{\forall i,j} V_{i,j}^2$ as the set of all user-viewed product pairs (UVP). The set UV can be extracted from the user log of the Web site considering a certain period of time. In practice, the identification of users and sessions in an e-commerce Web log is based on some standard methods (SRIVASTAVA et al., 2000). The description of these methods is out of the scope of this thesis, but we used them while carrying out our experiments.

We claim that there exists a non-empty intersection $S \cap UV$ between the set S of similar alternative pairs and the set UV of user-viewed product pairs. This concept is illustrated in Figure 2. The intuition behind this claim is that one of the most common reasons for users to visit two products during a session is because they are similar alternatives.

The approach we propose in our work essentially consists in using pairs in this intersection as examples to induce the product comparison function $\mathcal{S}$.

Now, to identify such pairs, we propose a strategy to find a set of HC of *highly co-related products pairs (HCP)* among user-viewed product pairs, that is $HC \subseteq UV$. In

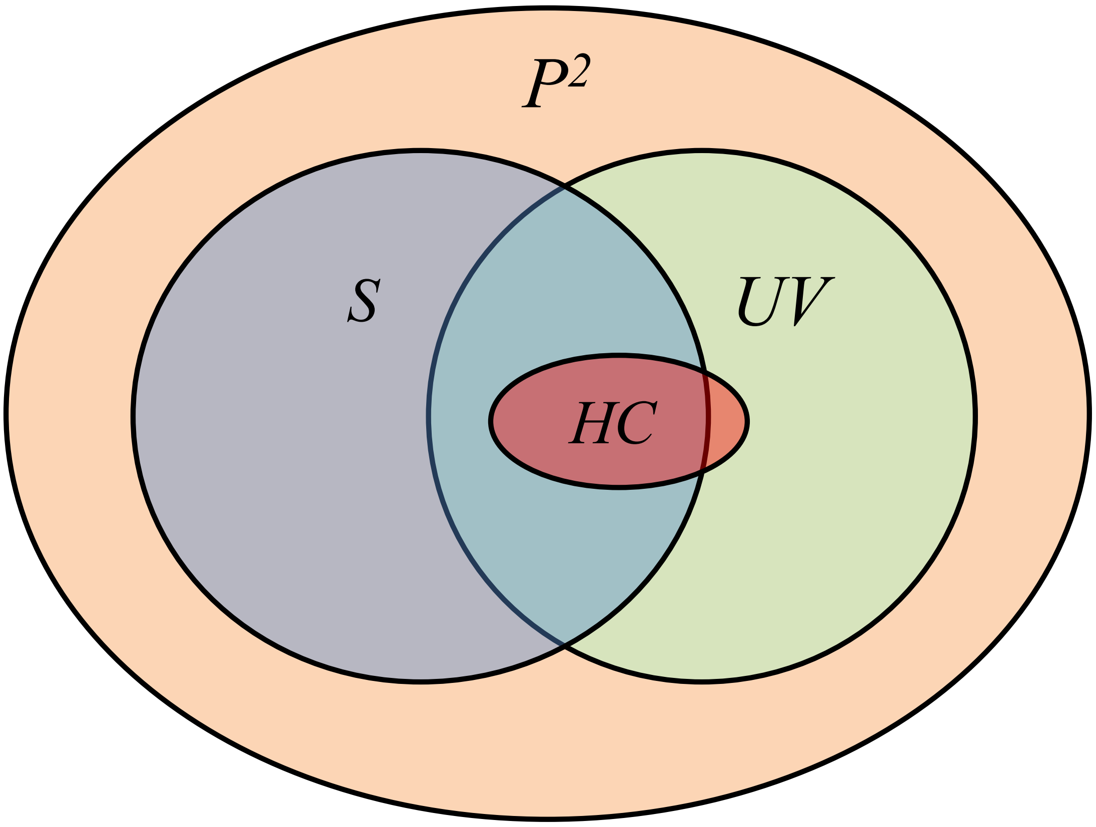

Figure 2 – Illustration of the main concepts related to our assumptions to obtaining training data.

this work we show empirical evidences that suggest that at least some of the pairs in HC are similar alternatives, that is $HC \cap S \neq \emptyset$. The strategy we propose for generating HC tries to ensure that this set is a representative subset of UV and that most, if not all, of its elements are also in S. Under such a condition, we may use the pairs in HC as examples to induce the product comparison function $\mathcal{S}$. As described in the Chapter 6, our strategy for generating HC relies in a number of factors, such as the frequency of the pairs, the significance of the pairs among all possible pairs and the average similarity of the attributes values for the products in the pair.

3.4 GPClerk

Figure 3 summarizes our method, called *GPClerk*, which we developed based on the approach described above.

The first step ① of the method is simply mining the user log to obtain the set UV of all user-viewed product pairs. Next, ② the method carries out the mining of the set HC of highly co-related products pairs. In addition, it also identifies a set of product pairs, we call NC, that are estimated to be not correlated at all. This process

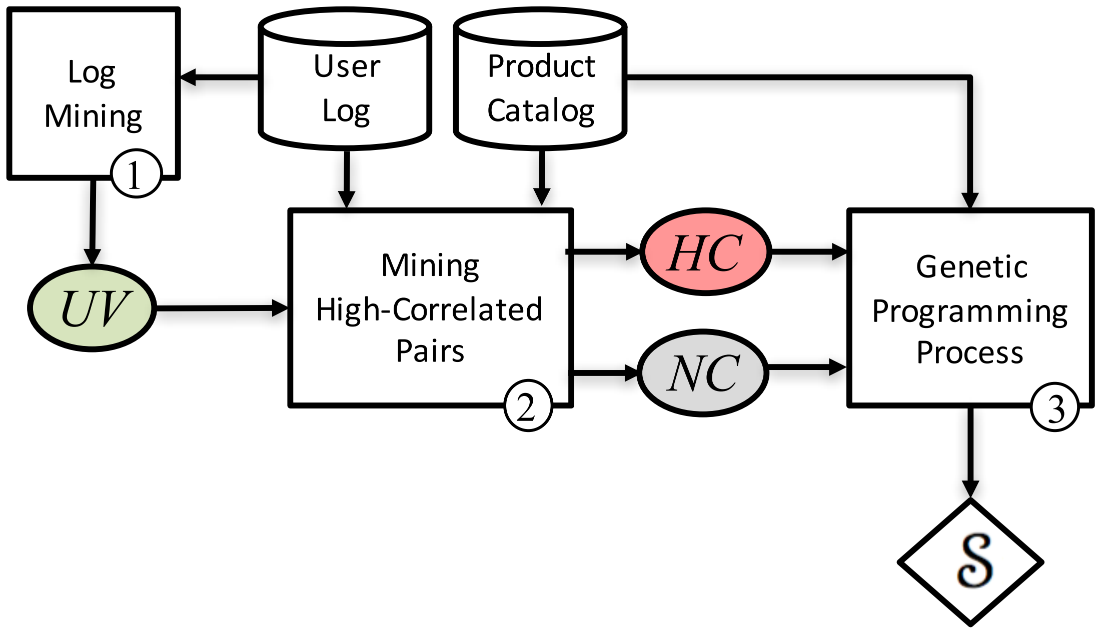

Figure 3 – Main steps in GPClerk.

leverage information taken both from the user log (e.g., frequency of products and pairs, etc.) and from the product catalog (e.g., attribute values). Finally, ③ the pairs in *HC* and *NC* are used, respectively, as positive and negative training examples for the genetic programming process that generates a product comparison function $\mathcal{S}$. The last two steps are described in the details in the next chapters.

Once the function $\mathcal{S}$ is generated, it can be used to identify pairs of similar alternative products. In the application scenario we consider, one of the products, say x, is given (e.g., by a costumer) and we want to find similar alternative products in the same category.

Currently, in our method, we perform a simple pairwise comparison between x and all other $y \in P$. As the number of true pairs of similar alternative products S is typically a small portion of P^2, some form of blocking technique (CHRISTEN, 2012) can be adapted to avoid this brute force procedure. This is left as future work.

3.5 Multiple Product Comparison Functions

The fact that the sets *HC* and *NC* can be obtained in an unsupervised fashion allows us to select several subsets of these sets and generate different product comparison functions, say $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k$ from these subsets. These functions can then be combined using some ensemble technique, resulting in a stronger prediction (WITTEN; FRANK, 2005) for pairs of similar alternative products. To do so, our method is used for generating n distinct product comparison functions according to the process described above, using different subsets of *HC* and *NC* selected at random. Then, the best $k < n$

functions are selected and are combined using a simple voting scheme (bagging) to predict if two products in a given pair are similar alternatives. This procedure was used in one of our experiments, in which we used $n = 20$ and $k = 10$.

4 Attribute-Specific Similarity

In this chapter we describe the attribute-specific similarity functions we adopt in our work. As described earlier, these functions are combined through a function $\mathcal{S}$ to obtain score for product comparison.

4.1 Attribute Taxonomy

In our work, each attribute of a given product category is assigned to a single class of a simple attribute taxonomy comprising four classes, namely: *Numerical*, *Categorical*, *Multicategorical* and *Dimensional*. This taxonomy was created based on previous work by Kagie et. al. (KAGIE; WEZEL; GROENEN, 2008b; KAGIE; WEZEL; GROENEN, 2008a) and in our own experience in dealing with e-commerce catalogs. The original taxonomy by Kagie et. al. in (KAGIE; WEZEL; GROENEN, 2008b) included only *Numerical* and *Categorical* attributes. It was extended in (KAGIE; WEZEL; GROENEN, 2008a) to include the *Multicategorical* class. We further expanded it with the *Dimensional* class to handle the common case of attributes that describe the dimensions of products, displays, etc.

To determine the class of each attribute, a number of different approaches could have been used. In our case, we opted for using a simple strategy in which the values expected for the attributes in a given class are described by a regular expression we call *Domain Descriptors*. Domain descriptors are similar to the *Data Frames* used by Embley et. al. in several methods (e.g., in (AL-MUHAMMED; EMBLEY, 2007)) and provide a description on how values of attributes of the four classes above are written.

The classification of an attribute is carried out as follows. Let A_k be an attribute that occurs for products $p_1, \ldots, p_n$ in a given category. For instance, attribute *Scene Modes* occurs in the description of many products in the *Cameras* category.

First, for all products $p_i (1 \leq i \leq n)$, we take the value v_{ik} for A_k occurring in p_i. Next, we perform several cleaning and standardization operations over set of values v_{ik} of A_k taken from products. These operations include duplicate values removal, white space and case normalization, among others. The result is a set of values $o_{1k}, \ldots, o_{mk} (m \leq n)$ which we call the *occurrences* of A_k. Notice that by doing so we assume that all values of A_k have the same semantics in all p_i. For instance, we assume that the attribute *Scene Modes* has the same semantics in the description of all products in the *Cameras* category.

Finally, we test each occurrence $o_{1k}, \ldots, o_{mk}$ against each domain descriptor

ϵ_ℓ (ℓ=1,...,4) and associate attribute A_k with the attribute class C_ℓ whose domain descriptor ϵ_ℓ recognizes the majority of its occurrences.

Although simple, this classification procedure is very effective as we demonstrate in experiments we carried out and reported in (HOFFMANN; SILVA; CARVALHO, 2015).

4.2 Similarity Functions

For each one of the four attribute classes we defined an appropriate similarity function. In the following, each function s^{α}_{xyk} computes the similarity of values v_{xk} and v_{yk}, which are the respective values of products x and y for the attribute A_k, whose class is $\alpha \in \{N, C, M, D\}$.

For the *Numerical* class, the similarity function is defined as the absolute difference between the values of the attribute in the two products, as shown in Equation 4.1.

$$s^{N}_{xyk} = 1 - \frac{|v_{xk} - v_{yk}|}{\max(v_{xk}, v_{yk}),} \tag{4.1}$$

For the *Categorical* class, the similarity function is defined as

$$s^{C}_{xyk} = 1(v_{xk} = v_{yk}) \tag{4.2}$$

implying that objects having the same value get a similarity score of 1 and 0 otherwise.

For the *Multicategorical* class, the similarity function is computed using the Jaccard coefficient (CHA, 2007), as follows.

$$s^{M}_{xyk} = \frac{|v_{xk} \cap v_{yk}|}{|v_{xk} \cup v_{yk}|} \tag{4.3}$$

In this case, v_{xk} and v_{yk} denote the sets of individual categorical values composing the actual values. For instance, v_{xk} would be { *Auto, On, Off, Slow Syncro*, ... } for the attribute *Flash Modes* in the camera *Sony W830* of Table 1.

For the *Dimensional* class, the similarity function is the normalized euclidean distance over the dimension values, as described in Equation 4.4.

$$s^{D}_{xyk} = 1 - \left[\sum_{d=1}^{M} ((v^{d}_{xk})' - (v^{d}_{yk})')^2 \right]^{\frac{1}{2}} \tag{4.4}$$

where M is the number of dimensions found in the values of the attribute and v^{d}_{xk} is the value for dimension d in v_{xk} (the same applies to v^{d}_{yk}). For computing this function, each dimension is mean-centred and normalized using

$$(v^{d}_{xk})' = ((v^{d}_{xk}) - \mu^{d})/\sigma^{d} \tag{4.5}$$

μ^d and σ^d are, respectively, the mean and the standard deviation of the set of values of dimension p in all values of attribute k, for the products in the category.

It is important to notice that we do not make use of the price attribute. Despite the fact that it is a relevant attribute, its value is not fixed, like the other ones (e.g. dimensions, weight, etc.), as it can eventually be changed on commercial interests (e.g. Black-Friday, Christmas, etc.) nonetheless. In a nutshell, our approach only relies on the product attributes that are strictly related to its technical features and physical characteristics.

5 Genetic Programming Process

In this chapter, we present the genetic process used in GPClerk. Before describing the process it self, we review some major concepts on Genetic Programming.

5.1 Genetic Programming Basic Concepts

Genetic Programming (GP) (KOZA, 1992; BANZHAF et al., 1998) is one of the best-known evolutionary programming techniques. It is a direct evolution of programs or algorithms used for the purpose of inductive learning (supervised learning). GP, as well as others evolutionary techniques, is known for its capability of working with multi-objective problems, that are normally modelled as environment restrictions during the evolutionary process (BANZHAF et al., 1998). It is also widely known for their good performance on searching over very large – possibly infinite – search spaces, where the optimal solution in many cases is not known, usually providing near-optimal answers (KOZA, 1992; BANZHAF et al., 1998).

Individuals and Genetic Operations

Usually, a GP process evolves a population of length-free data structures, often called *individuals*. Each individual represents a single solution to a given problem. In our particular case, individuals correspond to the product comparison functions defined in Chapter 3.

As in other GP-based solutions, in here we adopt trees to represent such individuals. For this, a set of terminals and functions must be defined (KOZA, 1992). Terminals are inputs, constants or zero argument[1] that correspond to leaf nodes in the tree. The function set is the collection of operators, statements, and basic or user-defined functions that can be used by the GP evolutionary process to manipulate the terminal values. These functions are placed in the internal nodes of the tree.

During the evolutionary process, individuals are handled and modified by genetic operations such as *evaluation*, *reproduction*, *selection*, *crossover* and *mutation* (KOZA, 1992). This process occurs in an iterative way that is expected to spawn better individuals (solutions to the proposed problem) in the subsequent generations.

The *evaluation* operation assigns to an individual a value that measures how suitable that individual is to the proposed problem. In our case, individuals are evaluated on how well they predict that two products are similar alternatives, using

[1] Terminals nodes that have an arity of zero.

the set of functions and terminals available. The resulting value is also called *raw fitness* and the evaluation functions are called *fitness functions*.

Reproduction is the operation that copies individuals without modifying them. Usually, this operator is used to implement an elitist strategy (KOZA, 1992), that is adopted to keep the genetic code of the fittest individuals across the changes in the generations.

The *selection* operation applies a criterion for choosing the individuals that should be in the next generation. After the evaluation operation, each solution has a fitness value that measures how good or bad it is to the given problem. Using these values, it is possible to decide whether an individual should be in the next generation.

The *crossover* operation allows genetic content (e.g., subtrees) exchange between two parents, in a process that can generate two or more children. In a GP evolutionary process, two parent trees are selected according to a matching (or pairing) policy and, then, a random subtree is selected in each parent. Child trees are the result from the swap of the selected subtrees between the parents.

Finally, the *mutation* operation has the role of keeping a minimum diversity level of individuals in the population, thus avoiding premature convergence. Every solution tree resulting from the crossover operation has an equal chance of suffering a mutation process. In a GP tree representation, a random node is selected and the corresponding subtree is replaced by a new randomly created subtree.

Generational Evolutionary Algorithm

GP evolutionary processes are usually guided by a *generational evolutionary* algorithm. This means that there are well-defined and distinct generation cycles. The steps of this algorithm are the following:

1. Create an initial population with random generated individuals.
2. Evaluate all individuals in the current population, assigning a numeric rating or *fitness value* to each one.
3. If the termination criterion is fulfilled, then, execute the last step. Otherwise continue.
4. Reproduce the best B individuals into the next generation population.
5. Using a selection process, select M individuals that will compose the next generation with the best parents.
6. Apply the genetic operations to all individuals selected. Their offspring will compose the next population. Replace the existing generation by the generated population and go back to Step 2.

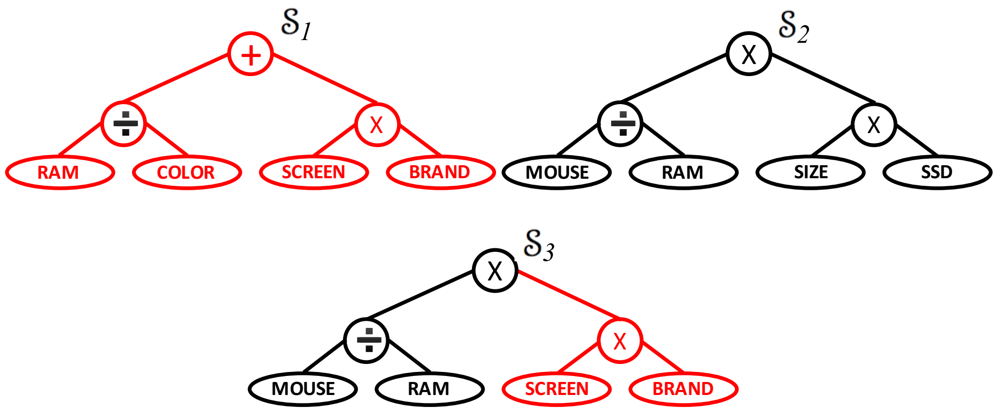

Figure 4 – Examples of trees representing product comparison functions. Tree $\mathcal{S}_3$ results from a crossover operation involving Trees $\mathcal{S}_1$ and $\mathcal{S}_2$.

7. Present the best individual(s) in the population as the output of the evolutionary process.

5.2 Evolving Product Comparison Functions

Product Comparison Functions

As already mentioned, the individuals in our GP process are product comparison functions for products in a given product category C. Each function is represented as a tree whose leaves correspond to attributes of the category and whose internal nodes represent simple arithmetic operations (i.e., $+,-,\times,\div$,exp). Integer constants from 0 to 9 are also allowed as leaves.

In Figure 4, we illustrate trees representing product comparison functions on the Notebook category. For instance, in Tree $\mathcal{S}_3$ the leaves represent attributes ***MOUSE***, ***RAM***, ***SCREEN*** and ***BRAND***.

Given a pair of products, x and y, an comparison function $\mathcal{S}$ computes an *similarity score* between them as follows. First, for each attribute A_k in the leaves of $\mathcal{S}$, it takes the value of the similarity function s^X_{ijk} between the values of attribute A_k for products i and j. This similarity is computed according to the class $X \in \{N, C, M, D\}$ of A_k, as described in Chapter 4, generating a normalized real number value between 0.0 and 1.0. For instance, in Tree $\mathcal{S}_3$, the similarity function s^C_{ijk} will be used for ***MOUSE*** and ***BRAND***, which are categorical attributes, s^N_{ijk} will be used for ***RAM***, a numeric attribute, and s^D_{ijk} will be used for the dimensional attribute ***SCREEN***.

Once these values are known, the operators defined in internal nodes up to the root node are applied to produce an *similarity score* that measures if two products x and y can be considered as similar alternatives according to this particular product comparison function.

Our goal is to evolve a suitable product comparison functions $\mathcal{S}$, so that $\mathcal{S}(x,y) \geq \tau$, if x and y are similar alternative products, according to the discussion in Chapter 3, and $\mathcal{S}(x,y) < \tau$, otherwise, where τ is an arbitrary threshold value.

To evolve such functions, we propose the PEFEvol algorithm, which is a Generational Evolutionary Algorithm we have designed for using in GPClerk. This algorithm is presented next.

The PEFEvol Algorithm

The specific GP process we use in our work is described by the algorithm PEFEvol presented in Figure 5.

```
Algorithm PEFEvol

Training Pairs
  $\mathcal{T}^+$: set of pairs of products regarded as similar alternatives.
  $\mathcal{T}^-$: set of pairs of products not regarded as similar alternatives.

Parameters
B: Number of best individuals to be selected in each generation
N: Number of new individuals to be generated in each generation
P: Number individuals allowed in each generation $P = B + N$
G: Number of generations
$\tau$: Threshold value for product similarity

 1: $\mathcal{P}_0 \leftarrow$ Initial random population of $P$ individuals;
 2: for $j = 1$ to $G$ do
 3:     for each individual $\mathcal{S}_i \in \mathcal{P}_{j-1}$ do
 4:         $f_i \leftarrow fitness(\mathcal{S}_i, \mathcal{T}^+, \mathcal{T}^-, \tau)$
 5:     end for
 6:     $\mathcal{B} \leftarrow B$ best individuals from $\mathcal{P}_{j-1}$
 7:     $\mathcal{N} \leftarrow N$ new individuals from genetic operations over $\mathcal{P}_{j-1}$
 8:     $\mathcal{P}_j \leftarrow \mathcal{B} \cup \mathcal{N}$
 9: end for
10: $\mathcal{B} \leftarrow B$ best individuals from $\mathcal{P}_G$
11: return $\mathcal{B}$
```

Figure 5 – Algorithm for Generating Comparison Functions

The algorithm takes as input training sets with examples of pairs of negative and positive pairs of a same category. It also takes a number of parameters that guide the evolution process.

The process begins with a initial population of product comparison functions generated entirely at random (Line 1). The evolutionary process itself corresponds to the Loop 2–9, which iterates a fixed number of times according to parameter G.

In the Loop 3–5, each individual $\mathcal{S}_i$ in the current population $\mathcal{P}_{j-1}$ is evaluated by a fitness function, assigning a fitness value f_i to it. The fitness function works by computing an similarity score for each pair in the training sets using $\mathcal{S}_i$ and verifying how well $\mathcal{S}_i$ performs in the tasks of correctly predicting if the products in a pair are similar alternatives or not. Given its importance to the process, we briefly postpone the discuss on the details of the fitness function we adopt.

The next population of product comparison functions is generated in Line 8. This population will have P individuals, including the B best individuals of the current population, according to their fitness values (Line 6) and N new individuals generated by apply genetic operations over the individuals of the current population (Line 7). Based on previous results of on evolving functions of this kind (CARVALHO et al., 2006; CARVALHO et al., 2008; CARVALHO et al., 2012) and after preliminary tests, in our experiments we adopted a setup with P=200, B=40 and N=160.

In the end of the process, the algorithm outputs the B best individuals from the last population $\mathcal{P}_G$. In our experiments we used $G = 40$, also based on our past experience.

Fitness Evaluation

A crucial step in the Generational Evolutionary Algorithm we adopt is the evaluation of each individual in the current population according to a *fitness function*, so that the fittest individuals can be selected for contributing to the next generation.

In our setting, this function is used to express, as a single value, how well a specific individual performs in the task of predicting a pair of similar alternative products based on the values of their common attributes. Thus, in our work we adopted the *accuracy metric*.

Let T be a set of training products pairs, so that $\mathcal{T} = \mathcal{T}^+ \cup \mathcal{T}^-$, where $\mathcal{T}^+$ is a set of alternative product pairs and $\mathcal{T}^-$ is a set of product pairs that are not similar.

For each pair $\langle x, y \rangle \in T$, the fitness evaluation computes a similarity score $S = \mathcal{S}(x, y)$. If $\mathcal{S}(x, y) \geq \tau$, x and y are considered as similar alternatives products.

Let $\mathcal{P}^+$ and $\mathcal{P}^-$ be respectively the set of pairs considered as similar alternatives and those considered otherwise by a comparison function $\mathcal{S}$. We measure the accuracy of $\mathcal{S}$ as:

$$\text{accuracy} = \frac{|\mathcal{P}^+ \cap \mathcal{T}^+| + |\mathcal{P}^- \cap \mathcal{T}^-|}{|\mathcal{T}|}$$

The choice of the threshold value τ to separate similar alternative pairs from those that are not is an import issue in our method. However, as observed in (CARVALHO et al., 2012), GP-based algorithms are typically able to evolve functions adapted to arbitrary threshold values in a wide range. That is, the generated functions become progressively adapted to a fixed threshold value given as input. Based on this observation, we use a fixed value of τ=1 in all of our experiments.

5.3 Practical Issues

As described in Chapter 3, considering the application scenario of an e-commerce Web site, is hard to obtain reliable training data to evaluate the fitness of the individuals being generate. As discussed earlier, recruiting some costumers and have them finding some pairs of similar alternative products in a large catalog is hardly viable. Moreover, such cumbersome procedure would have to be repeated for each site and each category. In the next chapter we describe the details of an alternative strategy we propose for obtaining training data.

Another important issue is that, in practice, as discussed Chapter 3.5, our method can be used to evolve several product comparison functions using Algorithm PEFEvol. In this case, a few best functions are selected and combined using a simple voting scheme (bagging) to determine if two products in a given pair are similar alternatives. We explore this strategy in one of our experiments.

6 Obtaining Training Data

As described in Chapter 3, to avoid the user's effort in labelling training data for the GP process, we rely on user behaviour information taken from the log for obtaining training data. In this chapter, we detail the process of mining high-correlated pairs, the HC set, and non-correlated pairs, the NC set, which are used, respectively, as positive and negative training examples for the GP process described in Chapter 5. This corresponds to Step 2 in our method GPClerk (Figure 3).

In a nutshell, given the set UV of all user-viewed product pairs of a same category in a certain period of time, we apply two consecutive filters. The first filter, the *Relevance Filter* applies a relevance metric based on the frequency and on the estimated significance of pairs, to spot the most relevant pairs from UV. Then, a second filter, we call the *Semantic Filter* is applied to filter out pairs which, although relevant in comparison to the other pairs mined from log, have disparate values in their attributes, and thus, cannot be adjudged as similar alternatives. These two filters are described henceforth.

6.1 Relevance Filter

The relevance filter tries to identify among all pairs in the set UV of Figure 2, those that are likely to be also in E, that is, those that are more relevant in comparison to the other pairs of products viewed together.

Intuitively, we may consider that the pairs which are frequently viewed together are likely to be pairs of similar alternative products, even though they probably cover small subset of the similar alternative pairs. However, frequency alone is not enough to indicate similarity. In particular, there are some products that are frequently viewed together with many different products, that is, they occur in many distinct pairs mined from the log. This is the case, for instance, of very popular products, or those promoted by marketing campaigns (e.g., products advertised on the main page of the site), which will be frequently viewed by many users, thus appearing along with many other products. We claim that pairs containing these products are less *significant* than those pairs formed by products that co-occur with fewer other products. Thus, our measure of relevance for a pair of products combines frequency with significance.

Pairing Frequency

For a pair of products $\langle x,y \rangle$, its raw frequency $f(x,y)$ is simply the number of times these products were "viewed" together. For smoothing purposes, we use a popular variation of sub-linear frequency scaling, define as:

$$pf(x,y) = \begin{cases} 1 + \log f(x,y) & \text{if } f(x,y) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{6.1}$$

In our particular setting, we call this formula the *pairing frequency* of products x and y.

Pair Significance

Following the ideas described in (TATTI, 2007), the significance of a pair can be evaluated by verifying how "surprising" is the occurrence of the products composing in comparison to the other products.

Given a product x, let $P(y|x)$ be the probability of a pair be composed by product y given this pair contains x. If this probability is high for many distinct y, then the occurrence of x in a pair is not "surprising". On the other hand, this occurrence is more meaningful if this distribution is more biased. As in (TATTI, 2007), we used the *entropy* of the distribution of $p(y|x)$ to characterize this diversity. This entropy can be calculated as:

$$H = - \sum_{y \in \text{ all pairs } \langle x,y \rangle} p(y|x) \log p(y|x)$$

The values of probabilities $p(y|x)$ can be estimated empirically from the pairs mined from log by using

$$p(y|x) = \frac{N_{x,y}}{N_x},$$

where $N_{x,y}$ is the number of pairs containing x and y and, N_x is the number of pairs containing x along with any other product. Thus,

$$H = - \sum_{y \in \text{ all pairs } \langle x,y \rangle} \frac{N_{x,y}}{N_x} \log \frac{N_{x,y}}{N_x} \tag{6.2}$$

In Eq. 6.2, as we are trying to measure how diverse are the pairs in which product x appears, we only need to count distinct pairs of products, that this$N_{x,y}$ is 1 if there is at least one pair $\langle x,y \rangle$ mined from the log, and it is zero otherwise. In the same way, N_x is the the number of distinct pairs that include product x. Thus, we can simplify Eq. 6.2 as follows.

$$H = - \sum_{y \in \text{all pairs } \langle x,y \rangle} \frac{1}{N_x} \log \frac{1}{N_x} \tag{6.3}$$

We can simplify Eq. 6.3 by replacing the sum over all pairs $\langle x,y \rangle$ with N_x since the sum would be executed N_x times. Thus, the simplified equation can be written as follows.

$$H = -N_x \times \frac{1}{N_x} \log \frac{1}{N_x} = \log N_x \tag{6.4}$$

Given two products x_1 and x_2, we say that the most significant one for our purposes is the one with the *lowest* value computed by Eq. 6.4. Thus, in practice instead of comparing the entropy corresponding to each product, we need an absolute metric that gives highest values to products with the lowest values for Eq. 6.4. This metric, we call *product significance*, is given by:

$$sig(x) = \log N - \log N_x = \log \frac{N}{N_x}, \tag{6.5}$$

where N is the total number of distinct pairs of products extracted from the log.

Then, considering that the joint entropy of a set of two variables is less than or equal to the sum of the individual entropies of these variables, for ranking purposes, we can define the significance of a pair of products $\langle x,y \rangle$ as the sum of the significance of their component products. This is define by Eq. 6.6.

$$sig(x,y) = \log \frac{N}{N_x} + \log \frac{N}{N_y} = \log \frac{N^2}{N_x \times N_y} \tag{6.6}$$

Estimated Pair Relevance

The estimated relevance of a pair is then defined by Eq. 6.7, which combines the pairing frequency (Eq. 6.1) and the pair significance (Eq. 6.6).

$$r(x,y) = pf(x,y) \times sig(x,y) \tag{6.7}$$

For each distinct pair of products extracted from the log, that is, those in the set UV of Figure 3, our method first computes its relevance according to Eq. 6.7. Then, these pairs are ranked and just a few of them in the top of the rank are selected.

There are many possible strategies to select these few top pairs. In the experiments we report here, we normalize all scores by the highest relevance value obtained and then we select only pairs with a normalized relevance score above a threshold value. For all experiments in all categories, we used 0.7 as the threshold value. This value was empirically chosen after a preliminary experiments we performed to validate our method.

6.2 Semantic Filter

The Relevance Filter described above is based on statistical properties that make some pairs stand out from the others. This filter does not consider the values of the attributes of the products in composing the pair. Thus, there can be cases in which a frequent and significant pair has products that cannot be similar, since their attributes have very disparate values.

To detect and filter out these spurious pairs, we propose applying a *Semantic Filter* over the pairs selected by the Relevance Filter. Thus, this filter outputs both the set of positive and negative examples for the genetic programming process. In terms of the Figure 3, these sets of examples correspond to the sets HC and NC.

The Semantic Filter works as follows. For each relevant pair $\langle x,y \rangle$, we compare the values of the attributes common to x and y using a function we call *Naive Product Comparison* (NPC) function, defined in Eq. 6.8.

$$\mathrm{NPC}(x,y) = \frac{\sum_{A_k \in \mathcal{A}_x \cap \mathcal{A}_y} s^{\alpha}_{xyk}}{\max\{|\mathcal{A}_x|, |\mathcal{A}_y|\}} \tag{6.8}$$

In this formula, $\mathcal{A}_x$ ($\mathcal{A}_y$) is the set of attributes with non-null values in x (y), and s^{α}_{xyk} is the attribute-specific similarity functions for attribute A_k, whose class is α, as defined in Chapter 4.

Eq. 6.8 corresponds to a product comparison function that computes a linear combination of the values obtained by attribute-specific similarity functions applied to the attributes common to x and y.

The idea here is that if the result of this simple linear combination is very high, then values of the attributes common to x and y are must be very close and they are likely to be similar.

Thus, we build the set HC of positive examples by including in it all relevant pairs of products $\langle x,y \rangle$, for which $NPC(x,y) \geq \beta_+$, where β_+ is a pre-defined threshold value.

The NPC function is also used to compose the set NC of pairs of non-correlated products, but using a different strategy. We select from all possible pairs of products, a random sample of pairs $\langle u,v \rangle$, for which $NPC(u,v) \leq \beta_-$, where β_- is also a pre-defined threshold value. To ensure a balance and bound the number of negative pairs, which is naturally much larger than the number of positive pairs, the random sample has the double number of pairs in HC, that this, $|NC| = 2 * |HC|$.

In all experiments we report in this work, we used $\beta_+ = 0.5$ and $\beta_- = 0.1$. These values were determined after a few initial validation experiments.

We notice that our NPC function is based on the *General Coefficient of Similarity* proposed by Gower (GOWER, 1971) and later used by Kagie et. al in (KAGIE; WEZEL; GROENEN, 2008b; KAGIE; WEZEL; GROENEN, 2008a) However, this coefficient is unsuitable to deal with products that have few common attributes. For instance, if two products have many distinct attributes with null values but just one common attribute and this attribute have the same non-null value for both products, Gower's coefficient results in high value. The NPC function tries to overcome this problem by normalizing the sum of attribute values similarities by the maximum number of attributes with non-null values between the two product, as defined in Eq. 6.8.

7 Experimental Results

In this chapter we report the experimental results we achieved in the evaluation of our method using real datasets of 5 different popular product categories. In the following sections, we first describe the details on the experimental setup. Next, we present two sets of experiments. The first set corresponds to those we carried out to validate the *PEFEvol* algorithm described in Chapter 5.2. The second set of experiments aims at validating the whole process of *GPClerk*, including the strategy for obtaining training data described in Chapter 6.

7.1 Experimental Setup

Table 2 presents the main features of the experimental datasets used in our experiments. These datasets were provided by Neemu[1], a company that develops search and recommendation technology for major e-commerce sites in Brazil. They comprise five different popular product categories: *Cameras* (CAM), *Refrigerators* (REF), *Laptops* (LAP), *Smartphones* (SMP) and *TVS* (TVS). The datasets in each category includes a set of products with their attribute values and a set of user viewed product pairs extracted from a seven months access log.

The product records available in these datasets often provide many attributes that are not related to the product characteristics themselves. For instance, attributes related to the packing of the products such as, packing dimension, package contents, etc., are very common. Thus, we disregarded these attributes in our experiments.

As a simple pruning strategy, we removed all attributes that are not found in at least 30% of the products records from a given category. In addition, a few cleaning procedures were used to remove noise from attribute values (e.g., spurious characters, etc.).

Table 2, shows, for each dataset, the number of products available (Products), the number of attributes initially available (Initial), and the number of attributes remaining after the pruning (Remaining). Notice that, even though many attributes were removed, still the number of attributes considered is large to be handled manually by humans.

This table also presents the number of attributes per each classes of our taxonomy (Chapter 4), that is: Numerical (N), Categorical (C), Multicategorical (M) and Dimensional (D). Notice that the large majority of the attributes is categorical. This

[1] <http://www.neemu.com>

Dataset	Products	#Attributes		#Attr/Class			
		Initial	Remaining	N	C	M	D
CAM	451	178	27	5	15	6	1
REF	129	53	39	3	34	1	1
LAP	335	76	26	5	19	1	1
SMP	126	105	38	2	29	6	1
TVS	187	96	31	6	22	2	1

Table 2 – Main features of the experimental datasets.

trend was observed in all categories. Also, only a single dimensional attribute was found in each category.

Table 3 presents the main parameters we used in the configuration of the GP process. These parameters are fairly standard and were the same used in experiments with other related GP-based methods (e.g.,(CARVALHO et al., 2012)).

Parameter	Value
Number of Runs	20
Max Number of Generations	40
Population	200
Initial Random Population Method	Ramped Half-and-Half
Reproduction Rate	20%
Selection Strategy	Ranking
CrossOver Rate	80%
CrossOver Method	Random SubTree Exchange
Mating (Pairing)	Random
Mutation Rate	2%
Mutation Operation	Random SubTree Insertion
Max Random Tree Depth	4

Table 3 – GP Parameters used in the experiments

7.2 PEFEvol Validation

In this first experiment our goal is to validate our evolutionary process. For this, we took sets of positive and negative training pairs selected by users (see Chapter 7.4) and executed several experiments using a standard cross validation procedure. For each round, each of the training datasets, positive and negative, was randomly shuffled and separated in two subsets, one for training (2/3) and one for validation (1/3). The results are presented in Table 4.

In Table 4, column "Pairs" shows the number of positive and negative pairs used for training in each category. Notice that the number of negative pairs is always double the number of positive pairs. This was enforced to bound the number of negative

Dataset	Pairs (+/-)	Accuracy	Precision	Recall	F-Measure
CAM	72/144	0.976	0.956	0.970	0.963
REF	55/110	0.824	0.749	0.725	0.737
LAP	48/96	0.938	0.930	0.887	0.908
SMP	56/112	0.935	0.867	0.894	0.880
TV	53/106	0.902	0.847	0.864	0.855

Table 4 – PEFEvol validation with user-provided training.

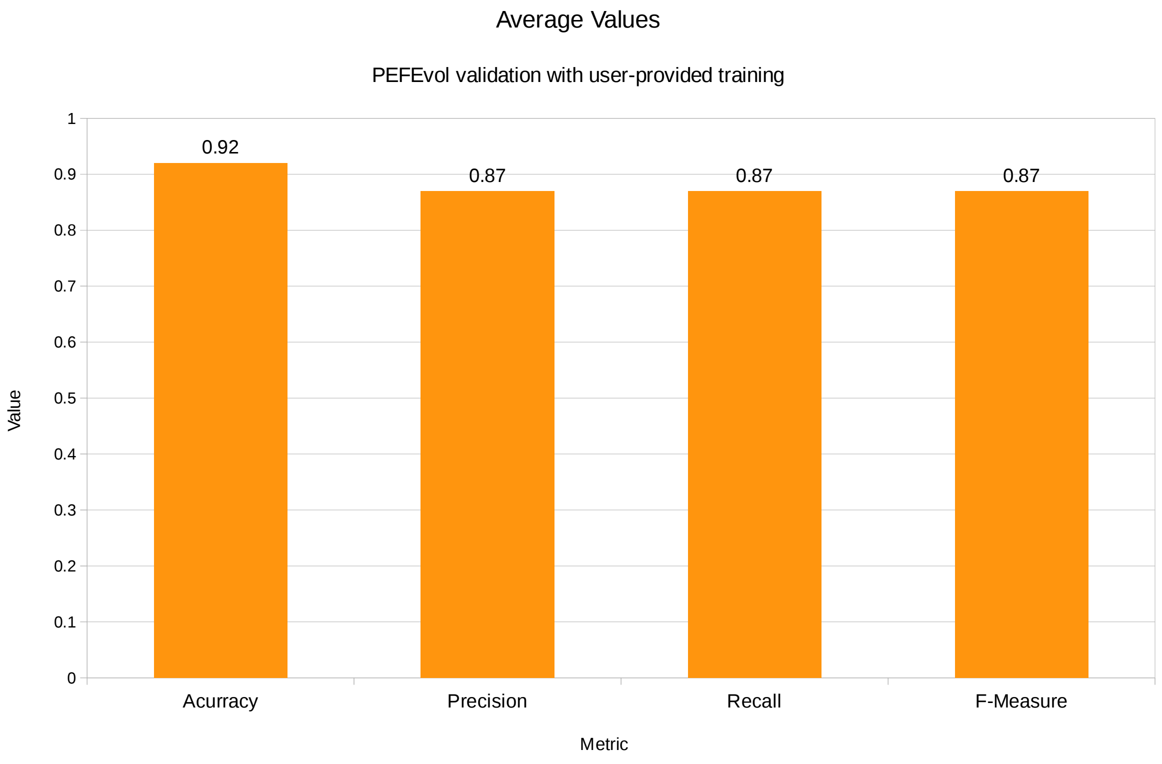

Figure 6 – Average values for all metrics in PEFEvol validation with user-provided training.

pairs, which is naturally much larger than the number of positive pairs. A excessive imbalance between the positive and negative sets could harm the learning process.

Table 4 also shows the values of obtained for accuracy, precision, recall and f-measure. All values corresponds to averages taken after 20 rounds of training and validation.

As it can be observed in Table 4, good results were obtained in all categories. Figure 6 shows overall averages values for the 5 categories above 0.8 in all metrics. In particular, recall that the accuracy measure is used as the fitness function in our GP process. The only metrics presented in Table 4 with values lower than the average showed in Figure 6 are the precision, recall, and, as a consequence, f-measure, for

"REF". In this category, which corresponds to refrigerators, products have many boolean attributes, whose values are only *YES* or *NO*, thus the similarity between attributes tends to be high for many attributes, leading to some false positive pairs.

Dataset	Accuracy			Precision			Recall			F-measure		
	HC	R	S	HC	R	S	HC	R	S	HC	R	S
CAM	**0.957**	0.888	0.951	**0.965**	0.911	0.956	**0.908**	0.737	0.896	**0.936**	0.815	0.925
REF	**0.815**	0.753	0.675	**0.735**	0.696	0.555	0.729	0.501	**0.749**	**0.732**	0.583	0.638
LAP	**0.967**	0.882	0.946	0.941	0.873	**0.945**	**0.963**	0.754	0.885	**0.952**	0.809	0.914
SMP	**0.902**	0.881	0.882	**0.850**	0.822	0.805	0.838	0.801	**0.850**	**0.844**	0.811	0.827
TV	**0.856**	0.763	0.829	**0.805**	0.681	0.766	0.732	0.574	**0.746**	**0.767**	0.623	0.756

Table 5 – PEFEvol validation with the Relevance Filter (R), with the Semantic Filter (S), and with both filters (HC).

Overall, the results obtained in this experiment indicate that our algorithm PEFEvol is able to evolve suitable product comparison functions when it receives reliable training data from users. However, in our proposed method GPClerk, this training data is automatically provided by the strategy described in Chapter 6. This scenario was addressed in another experiment we carried out and whose results are presented in Table 5 and Figure 7. Before analysing these results, we discuss the training data used for this experiment, whose details are presented in Table 6.

In Table 6, column "UV" presents the number of distinct User-viewed Product Pairs mined from the log and column "HC" presents the number of distinct pairs generated to be used as positive examples after applying the Relevance and the Semantic Filters, as described in Chapter 6, there is, the set *HC*. The number of pairs in set *NC*, which provides the negative examples, is double the value of "HC" for the same reason as the previous experiment.

Dataset	#Pairs			
	UV	HC	Relevance	Semantic
CAM	6068	51	98	206
REF	6073	272	396	3006
LAP	10169	134	345	341
SMP	1681	82	136	488
TVS	9289	281	665	1344

Table 6 – Training data automatically generated for PEFEvol

Also in Table 6, columns "Relevance" and "Semantic" show the number of pairs select by each of the respective filter taken in isolation. We used these sets of pairs to

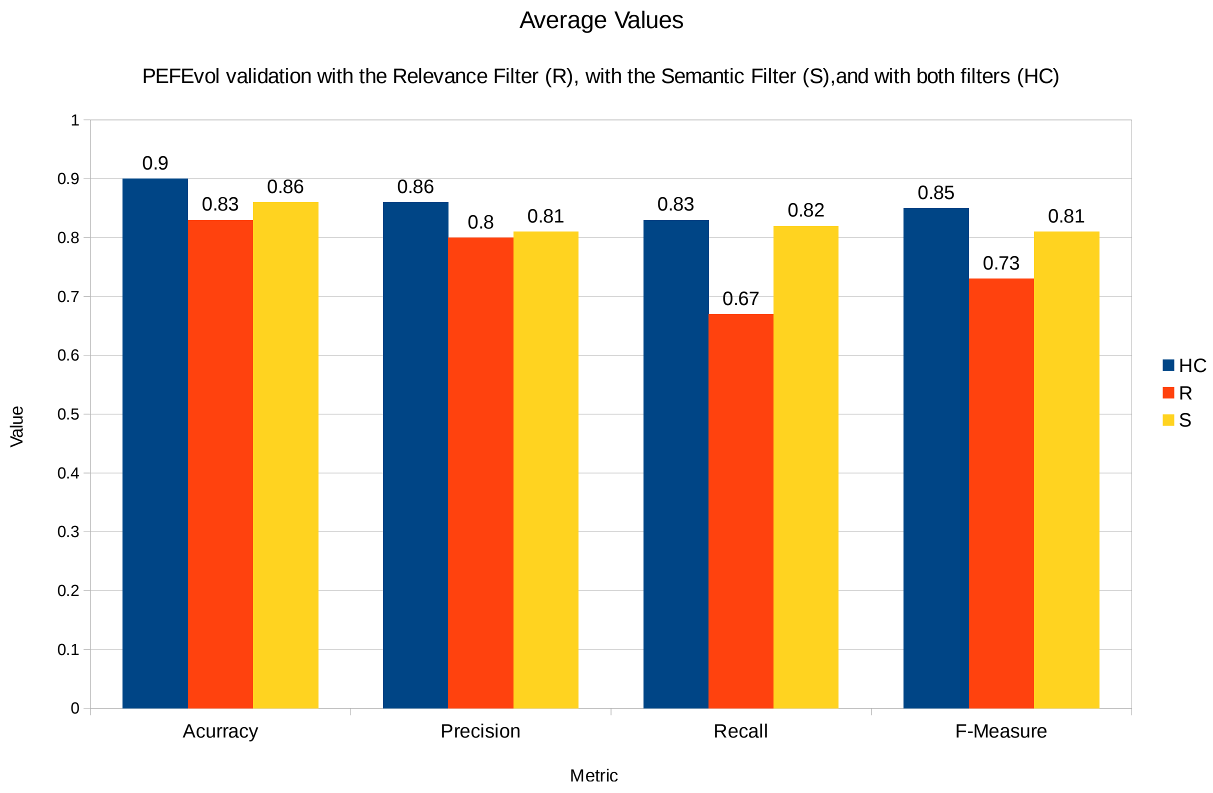

Figure 7 – Average values for all metrics in PEFEvol validation with the Relevance Filter (R), with the Semantic Filter (S), and with both filters (HC).

verify the effectiveness of each filter taken in isolation in comparison to using both filters.

To generate the sets of pairs presented in Table 6, for all categories, we used 0.7 as the threshold for the Relevance Filter, and $\beta_+ = 0.5$, $\beta_- = 0.1$ for the Semantic Filter. These values were determined after a few initial trial experiments.

Table 5 presents validation results for PEFEvol when training pairs are generated using the Relevance Filter only (R), with the Semantic Filter only (S), and with both filters (HC), as we do in GPClerk. The best results for each metric in each category are in boldface. We notice that, for all metrics, the overall averages obtained with both filters are better then with each filter individually as can be seen in Figure 7. Also, these averages are very similar to those obtained with user-provided training (see Figure 6). In fact, Table 5 shows that in the large majority of the cases, the results obtained with both filters are better than with a single filter. The exceptions are a few cases in which the Semantic Filter alone led to a better result.

To explain this, notice that the Semantic Filter alone is useful for detecting, on the training data, spurious products pairs, that is, those that in spite of being seen together in the log, cannot correspond to similar alternative products, since many of their common attributes do not have similar values. However as discussed above, in some categories, as in the case of REF, products have many boolean attributes, whose values are only *YES* or *NO*, thus the similarity between attributes tends to be high for many attributes, leading to some false positive pairs, harming the precision metric. Indeed, notice that the lower precision values for REF category were obtained when the semantic filter is used alone. In fact, in this category, the precision values with the semantic filter alone are the worst. Thus, the Relevance Filter has an important role of selecting only product pairs that are significant, among all pairs of products that are seen together.

7.3 GPClerk Evaluation

In our second experiment, our aim was to validate the whole process, including the strategy for obtaining training data (Chapter 6). For this, we first generated the sets HC and NC for each category, from User-viewed Product Pairs mined from the log. Then, we generate 20 random subsets with 2/3 of the pairs in HC and in NC. Next, each pair of subsets, say, HC_i and NC_i, are used as an input for the PEFEvol algorithm. Thus, 20 distinct product comparison functions were obtained. Finally, we used a simple voting scheme (bagging) to determine if two products in a given pair are similar alternatives or not. We took the 10 best of these functions, according to their fitness values, and evaluate each pair of products in a given category. Products in a pair were considered as similar alternatives, if at least 8 of the functions evaluate it as so. All these parameters were set empirically in initial validation experiments. Table 7 shows the total number of pairs evaluated (“All Pairs”) and the number of pairs considered as similar alternatives in each category (“Similar”).

Dataset	All Pairs	Similar	Sample	Precison	
				GPClerk	Random
CAM	101322	438	79	0.911	0.228
REF	7440	511	81	0.679	0.284
LAP	55543	731	85	0.565	0.318
SMP	7629	250	70	0.800	0.243
TVS	16548	537	82	0.646	0.317

Table 7 – Results of the user evaluation of GPClerk.

To measure the quality of the prediction made by the assembling of the 10 product comparison functions, 60 users were recruited. Each pair of products was evaluated by 3 users, and a product pair was considered (labelled) as representing similar alternative products, if at least 2 out of the 3 users found them as so. The evaluation was carried out using a sample of the pairs considered as positive by the functions. The size of sample was calculated to give 95% of confidence level in a 10

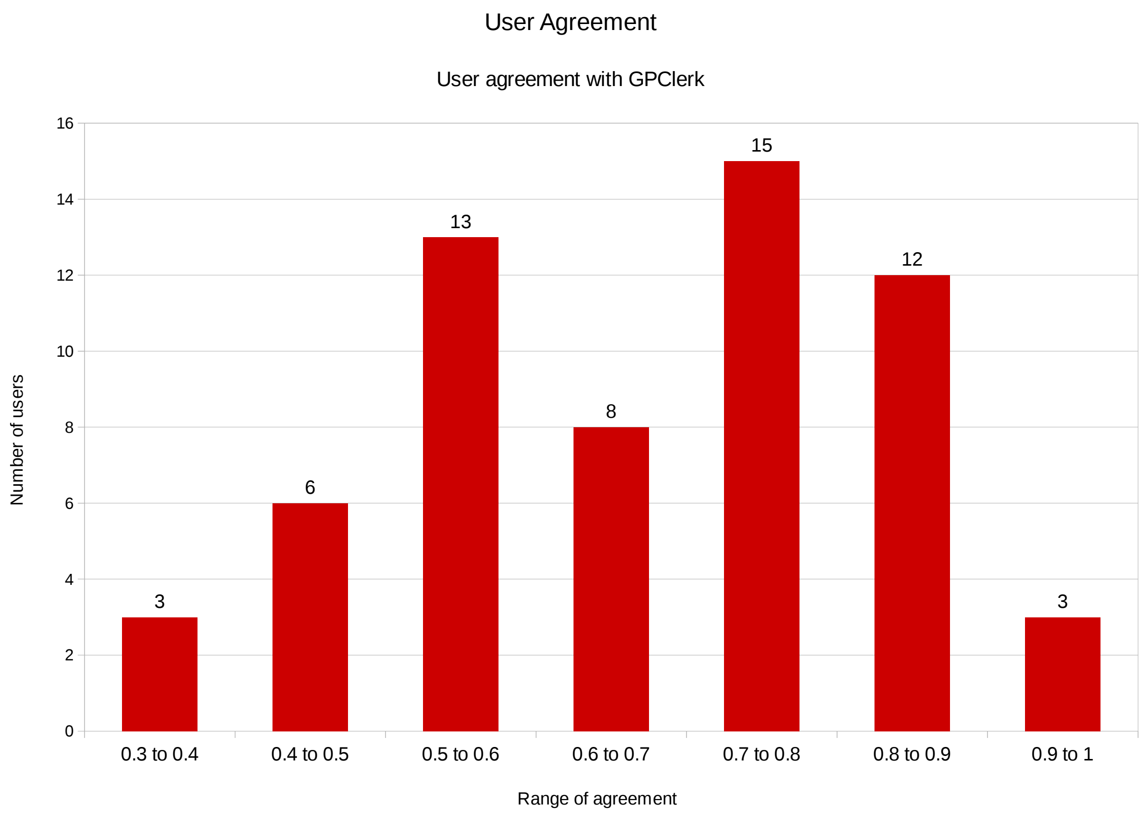

Figure 8 – Percentage of user agreement with GPClerk.

confidence interval. The users also evaluated a similar size sample, generated randomly. This made it possible to compare our GP based method to a simple random selection.

In Table 7 we present, for each category, the size of the sample evaluated by users ("Sample") and the precision achieved, both with GPClerk and with the random selection.

As it can be noticed, the GPClerk results are far superior to random's. For all datasets evaluated by the users, the GPClerk achieved satisfactory results. In the Camera Dataset, GPClerk achieved 0.91 of precision in according to user evaluation.

Figure 8 presents a graph that shows in which extent the users agreed with the similar alternative pairs identified by GPClerk. There was no such case in which the users completely disagreed from our method, as it can be seen in the lowest agreement values of 0.3 and 0.4 (from only 3 users out of the 60 participants). In fact, most of the users agreed with at least 50% of the pairs identified with GPClerk. In addition, half of the users agreed with at least 70% of the answers.

7.4 Result Analysis

The results achieved in the evaluation of the PEFEvol (Chapter 7.2) indicate that this algorithm is able to correctly learn patterns related to the similarity of the attributes of two products to predict whether these products are similar alternatives. For this, algorithm receives as input examples of pairs considered as similar alternatives. The examples labelled by users, used in the experiment reported in Table 4, were collected in our second experiment. We gathered the product pairs obtained in GPClerk user evaluation and submitted as input for the PEFEvol algorithm.

The second experiment shows that the strategy presented in Chapter 6 is effective for obtaining such examples in an unsupervised way. Importantly, the results of this second also validate the hypotheses we assume in Chapter 3. Indeed, according to the results in Table 7, the evaluation made by users confirms that there exists similar alternative pairs of products, and GPClerk is able on finding them, and these pairs can be identified by analysing the similarity of values of common attributes between products.

These results also confirmed that there are similar alternative pairs that do not belong to the set of product pairs mined from the log, i.e., those pairs occurring in the UV set of Figure 2. Table 8 shows details the precision achieve by the method considering only pairs that belong to UV and pairs that are not in UV. This leads us to conclude that, indeed, methods such as GPClerk that are able to spot pairs of similar alternative products in whole catalog, independently from the user behaviour, are needed.

Dataset	Precison	
	in UV	out of UV
CAM	0.923	0.906
REF	0.704	0.500
LAP	0.625	0.486
SMP	0.815	0.750
TVS	0.696	0.538

Table 8 – Precision achieved by GPClerk in and out of UV.

8 Conclusions and Future Work

In this chapter we present our conclusions and discuss directions for future work.

8.1 Conclusions

We described a novel method we designed, implemented and tested to find products that are similar alternatives to a given product. Although relatively new, this problem has already attracted attention both from the industry and from the academia.

Our method uses Genetic Programming (GP) to learn functions for comparing two products based on their attributes, so to tell whether the two products are similar alternatives or not. These functions are called here *product comparison functions*. We show GP is a suitable machine learning technique for capturing the notion of similarity from user provided examples. We call our method *GPClerk*. For this method, we proposed a GP-based algorithm called *PEFEvol* to evolute the product comparison functions from sets of training pairs of positive and negative examples.

We consider that providing training examples of similar alternative products is problematic in our setting, due to the potentially high number of pairs of products to consider, and to the degree of expertise required to compare values of more technical attributes (e.g., the shutter speed of a camera). To address this, GPClerk includes an unsupervised strategy we proposed to generate training examples. This strategy consists of mining logs for pairs of products of a same category that are frequently viewed together in same user section.

To evaluate our method, we carried out experiments using real datasets on 5 different popular product categories over hundreds of thousands product pairs. In a first set of experiments, we validate our evolutionary algorithm PEFEvol. The results indicate that the algorithm is able to successfully learn the concept of product comparison implicitly defined by instances in the training datasets, achieving F-measure values above 0.8, on average.

In a second set of experiments we carried out, the goal was verifying the effectiveness of the whole process of GPClerk, including the strategy for obtaining training data. This experiments was conducted with the help of users that verify the pairs of products identified as similar alternatives by our method. The results show that, considering all datasets, nearly 70% of the pairs indicated by our method as similar alternatives, were also identified by the users as so.

Another interesting results is that GPClerk was able to find pairs of similar alternative products, even if they are not viewed together by users. This indicates that our method is a suitable complement to the traditional recommendation methods based on collaborative filtering techniques, in particular for dealing with the long standing cold start problem.

8.2 Future Work

Our work with GPClerk opened a number of opportunities for future work. We mention some of them in the following.

The product comparison functions generated by the PEFEvol algorithm, implicitly indicates which attributes are the most important to users, at least according to the training data provided. By generating several of these functions and combining them using some logistic regression technique, we plan to identify which are the attributes that are the most important in a specific category. Once this attributes are identified, we believe that a number of applications may follow. Among them, multi-objetive optimization techniques could be used to rank products according to the most important attributes, so that products can be compared by their features.

In another line of investigation, we believe that our general approach can be adapted to find products related by other kinds of relationships besides similarity. In particular, we plan to experiment this approach to find complementary products, as in (MCAULEY; PANDEY; LESKOVEC, 2015). This the case of accessories, supplies, etc.

Finally, currently GPCLerk requires that products being compared have their specifications available in a product catalog. However, we believe that it can also be used to compare products represented by textual unstructured product offer descriptions. This would require integrating in the process information extraction methods like those proposed in (CORTEZ et al., 2010; CORTEZ et al., 2011).

Bibliography

AL-MUHAMMED, M.; EMBLEY, D. Ontology-based constraint recognition for free-form service requests. In: *IEEE 23rd International Conference on Data Engineering*. [S.l.: s.n.], 2007. p. 366–375. Cited on page 39.

BANZHAF, W. et al. *Genetic Programming - An Introduction: on the Automatic Evolution of Computer Programs and Its Applications*. [S.l.]: Morgan Kaufmann Publishers, 1998. Cited on page 43.

BILENKO, M.; BASU, S.; SAHAMI, M. Adaptive product normalization: Using online learning for record linkage in comparison shopping. In: *Proceedings of the Fifth IEEE International Conference on Data Mining*. Washington, DC, USA: IEEE Computer Society, 2005. (ICDM '05), p. 58–65. ISBN 0-7695-2278-5. Cited on page 29.

BURKE, R. Knowledge based recommender systems. In: *Encyclopedia of Library and Information Science*. [S.l.: s.n.], 2000. v. 69. Cited on page 25.

CARVALHO, M. G. de et al. Learning to deduplicate. In: *Proceedings of the 6th ACM/IEEE-CS Joint Conference on Digital Libraries*. [S.l.: s.n.], 2006. p. 41–50. ISBN 1-59593-354-9. Cited 2 times on pages 27 and 47.

CARVALHO, M. G. de et al. Replica identification using genetic programming. In: *Proceedings of the 23rd Annual ACM Symposium on Applied Computing - SAC* . [S.l.: s.n.], 2008. Cited on page 47.

CARVALHO, M. G. de et al. A genetic programming approach to record deduplication. *IEEE Trans. Knowl. Data Eng.*, v. 24, n. 3, p. 399–412, 2012. Cited 7 times on pages 27, 31, 32, 35, 47, 48, and 56.

CHA, S.-H. Comprehensive survey on distance/similarity measures between probability density functions. *International Journal of Mathematical Models and Methods in Applied Sciences*, v. 4, n. 1, p. 300–307, 2007. Cited on page 40.

CHRISTEN, P. A survey of indexing techniques for scalable record linkage and deduplication. *IEEE Transactions on Knowledge and Data Engineering*, v. 24, n. 9, p. 1537–1555, 2012. Cited on page 37.

CORTEZ, E. et al. Joint unsupervised structure discovery and information extraction. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2011, Athens, Greece, June 12-16, 2011*. [S.l.: s.n.], 2011. p. 541–552. Cited on page 64.

CORTEZ, E. et al. ONDUX: on-demand unsupervised learning for information extraction. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2010, Indianapolis, Indiana, USA, June 6-10, 2010*. [S.l.: s.n.], 2010. p. 807–818. Cited on page 64.

GOPALAKRISHNAN, V. et al. Matching product titles using web-based enrichment. In: *Proceedings of the 21st ACM International Conference on Information and Knowledge*

Management. New York, NY, USA: ACM, 2012. (CIKM '12), p. 605–614. ISBN 978-1-4503-1156-4. Cited on page 29.

GOWER, J. A general coefficient of similarity and some of its properties. *Biometrics*, v. 27, n. 4, p. 857–874, 1971. Cited on page 53.

HAMMAR, M.; KARLSSON, R.; NILSSON, B. J. Using maximum coverage to optimize recommendation systems in e-commerce. In: *Proceedings of the 7th ACM Conference on Recommender Systems*. New York, NY, USA: ACM, 2013. (RecSys '13), p. 265–272. ISBN 978-1-4503-2409-0. Cited on page 25.

HOFFMANN, U.; SILVA, A. S. da; CARVALHO, M. G. de. Finding similar products in e-commerce sites based on attributes. In: *Proceedings of the 9th Alberto Mendelzon International Workshop on Foundations of Data Management*. [S.l.]: CEUR-WS.org, 2015. (CEUR Workshop Proceedings, v. 1378). Cited on page 40.

ISELE, R.; BIZER, C. Learning expressive linkage rules using genetic programming. *Proc. VLDB Endow.*, v. 5, n. 11, p. 1638–1649, 2012. Cited 3 times on pages 27, 31, and 35.

KAGIE, M.; WEZEL, M. van; GROENEN, P. J. Choosing attribute weights for item dissimilarity using clikstream data with an application to a product catalog map. In: *Proceedings of the 2008 ACM Conference on Recommender Systems*. [S.l.: s.n.], 2008. p. 195–202. Cited 4 times on pages 29, 30, 39, and 53.

KAGIE, M.; WEZEL, M. van; GROENEN, P. J. A graphical shopping interface based on product attributes. *Decision Support Systems*, v. 46, n. 1, p. 265 – 276, 2008. Cited 4 times on pages 25, 29, 39, and 53.

KATUKURI, J. et al. Recommending similar items in large-scale online marketplaces. In: *IEEE International Conference on Big Data*. [S.l.: s.n.], 2014. p. 868–876. Cited 4 times on pages 25, 29, 30, and 31.

KöNIK, T.; MUKHERJEE, R.; KATUKURI, J. Subjective similarity: Personalizing alternative item recommendations. In: *Proceedings of the 24th International Conference on World Wide Web*. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2015. (WWW '15 Companion), p. 1275–1279. ISBN 978-1-4503-3473-0. Cited 2 times on pages 29 and 30.

KÖPCKE, H. et al. Tailoring entity resolution for matching product offers. In: *15th International Conference on Extending Database Technology, EDBT '12, Berlin, Germany, March 27-30, 2012, Proceedings*. [S.l.: s.n.], 2012. p. 545–550. Cited on page 29.

KOZA, J. R. *Gentic Programming: on the Programming of Computers by Means of Natural Selection*. [S.l.]: MIT Press, 1992. Cited 4 times on pages 27, 35, 43, and 44.

LONG, B. et al. Enhancing product search by best-selling prediction in e-commerce. In: *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*. New York, NY, USA: ACM, 2012. (CIKM '12), p. 2479–2482. ISBN 978-1-4503-1156-4. Cited on page 25.

MCAULEY, J.; PANDEY, R.; LESKOVEC, J. Inferring networks of substitutable and complementary products. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. [S.l.: s.n.], 2015. p. 785–794. ISBN 978-1-4503-3664-2. Cited 5 times on pages 25, 29, 30, 31, and 64.

NGUYEN, H. et al. Synthesizing products for online catalogs. *PVLDB*, v. 4, n. 7, p. 409–418, 2011. Cited on page 29.

SCHAFER, J.; KONSTAN, J.; RIEDL, J. E-commerce recommendation applications. *Data Mining and Knowledge Discovery*, v. 5, n. 1-2, p. 115–153, 2001. Cited on page 25.

SCHEIN, A. I. et al. Methods and metrics for cold-start recommendations. In: ACM. *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. [S.l.], 2002. p. 253–260. Cited on page 25.

SRIVASTAVA, J. et al. Web usage mining: Discovery and applications of usage patterns from web data. *ACM SIGKDD Explorations Newsletter*, v. 1, n. 2, p. 12–23, 2000. Cited on page 35.

TATTI, N. Maximum entropy based significance of itemsets. In: *Proceeding of the 7th IEEE International Conference on Data Mining*. [S.l.: s.n.], 2007. p. 312–321. Cited on page 50.

WITTEN, I. H.; FRANK, E. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. [S.l.]: Morgan Kaufmann Publishers Inc., 2005. Cited on page 37.

YIN, H. et al. Challenging the long tail recommendation. *Proc. VLDB Endow.*, VLDB Endowment, v. 5, n. 9, p. 896–907, maio 2012. ISSN 2150-8097. Cited on page 25.