ESGI 2016 – Cloud is Mine Report

Fatima Al Harbi, Zacharie Alès, Sabah Dimassi,
Guillerme Duvillié, Vincent Labatut, Céline Lacaux,
Elvys Linhares, Sang Thi Nguyen & Florentina Nicolau

July 18, 2016

1 Introduction

In this section, we introduce the company and the proposed problems. We then describe the provided database and identify some of its limitations which, to our opinion, prevent any effective resolution of the problem (Section 2). We nevertheless propose a purely theoretical solution (Section 3), but do not put it in practice because of the incomplete data. Finally, we explain how our method could be extended and improved (Section 4).

1.1 Cloud is Mine & AppVizer

Cloud is Mine is a French consulting firm specialized in cloud computing. It focuses on Small office/home office as well as Small and medium-sized enterprises.

AppVizer is a service provided by Cloud is Mine. It allows comparing Software as a service (SaaS) solutions, i.e. on-demand software, in order for the client company to find the tools the most relevant to its needs. The client enters a description of the targeted tool, using various optional fields: software name, features, language, etc. The search engine returns a list of SaaS with their description.

For now, the outputs of the search engine depends only on the user request. Cloud is Mine wants to leverage its data base to enhance its AppVizer service, by adding some data mining features allowing to implement various recommendation features.

1.2 Problem statement

Cloud is Mine identified two problems related to these targeted recommendation features.

Software recommendation. The first problem is the recommendation of relevant SaaSs to a given user. This recommendation would take the form of a list of SaaSs (possibly an ordered one). Basically, two situations can occur.

First, if the user is anonymous, only its request can be considered: the recommendation should therefore be based on the SaaS selected by previous users with similar requests. So, to solve this problem, it is necessary to be able to compare users through their requests.

Second, if the user is identified, then we can have access to his profile, including his request history. The system should take advantage of this additional information to enhance user comparison, and therefore improve its recommendations.

Feature recommendation. The second problem is the recommendation of features to software companies. Indeed, the firms developing SaaSs want to identify the features typically desired by certain types of users, in order to include them in their products and thus increase the demand.

Again, the identification of types of users requires to be able to compare users. They can then be grouped, for example through cluster analysis.

2 Data description

In this section, we described the database provided by Cloud is Mine. We identify several important limitations, corresponding to missing information, which prevent any reasonable resolution of the proposed problems.

2.1 Terminology

Here are the definitions of some concepts which appear in the database and are important to the problem:

- *Service*: a software (SaaS).
- *Edition*: a version of some software (free, paid, recent/older version, ...).

2.2 Data

Figure 1 shows the structure of the provided database. The most relevant table in this context is *user_search_history* which contains all the requests performed on appvizer and for each of them, the software selected or compared by the user, if any.

2.3 Limitations

User feedback. It is important to notice that both proposed problems require access to a *feedback* of some sort. For instance, a measure indicating how satisfied the user is from the results returned for his request, or the SaaSs the user selected among these results. However, there is no such feedback in the current AppVizer database.

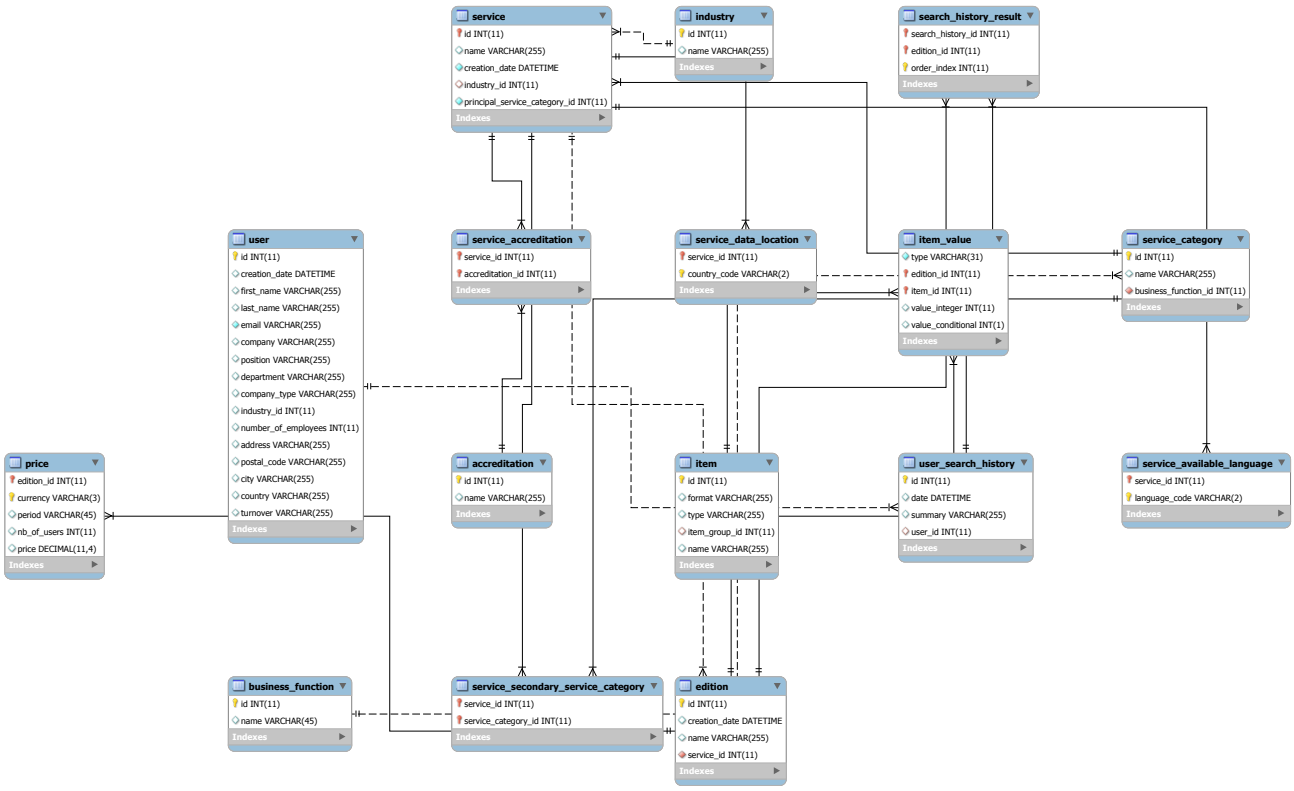

Figure 1: Relational diagram of the *Cloud is Mine* database.

To our opinion, this is the most important limitation. Since this prevent us from effectively testing the methods we propose to solve the problem, we only focus on its theoretical description in this report.

Session tracking. The other major limitation in the provided data is the absence of any tracking of the Web sessions. Since most of the users are anonymous, this would allow to at least characterize a user using not only his latest request, but also the ones he posted right before. Indeed, a user generally builds his request incrementally: first with one or a few fields, then by adding other fields depending on the returned results, in order to decrease the number of SaaSs proposed by the search engine.

In the current database, there is no way to link these different increments: each one is considered as a separate, anonymous request. Modifying the website to perform such a tracking and store in the database seems very doable, many Web frameworks implement these features (e.g. EJB). The cost is marginal,

and this would enhance significantly the relevance of the recommendations.

3 Problem resolution

Due to the previously mentioned limitations, we will only propose some theoretical methods to solve the problems at hand, but will not test them on the provided data, since this is not possible due to their incompleteness. This section describes the proposed methods.

In order to identify similar requests, we propose to represent each session by an oriented graph and to define similarity measures between these graphs.

3.1 Definitions

Let $G(V, A)$ be an oriented graph. The *indegree* of a node $i \in V$ (denoted by $deg^+(i)$) is the number of arcs which ends to vertex i. Similarly, the *outdegree* of a node $i \in V$ (denoted by $deg^-(i)$) is the number of arcs which starts from vertex i. A node is said to be a *source* (resp. *sink*) if its indegree (resp. outdegree) is equal to 0.

3.2 Session representation

A session can be represented by an oriented graph with a unique source node which contains the user information (or no information in the case of an anonymous user). Each other node in the graph represents a request. Intuitively, there is an arc from vertex u to vertex v if the request represented by v is a *prefix* of the request represented by u. In this case node v contains the additional information of this request as well as the information stocked in node u. Thus, in this representation an arc represents a set of research criterion. Each node also contains the softwares selected or compared by the user, if any. Figure 2 represents the graph associated to one hypothetical session.

3.3 Similarity measures

Based on similar requests, it seems possible to predict the softwares which are likely to be interesting for the current user of the website. In order to identify the sessions which are similar to the current one, similarity measures need to be defined.

In this section, we first present a similarity measure based on the comparison of the last created sink node of the graph associated to the current session and a sink node from a previously stored session. Secondly, we briefly describe how the whole structure of the two corresponding oriented graphs could be taking into account to improve the quality of this similarity.

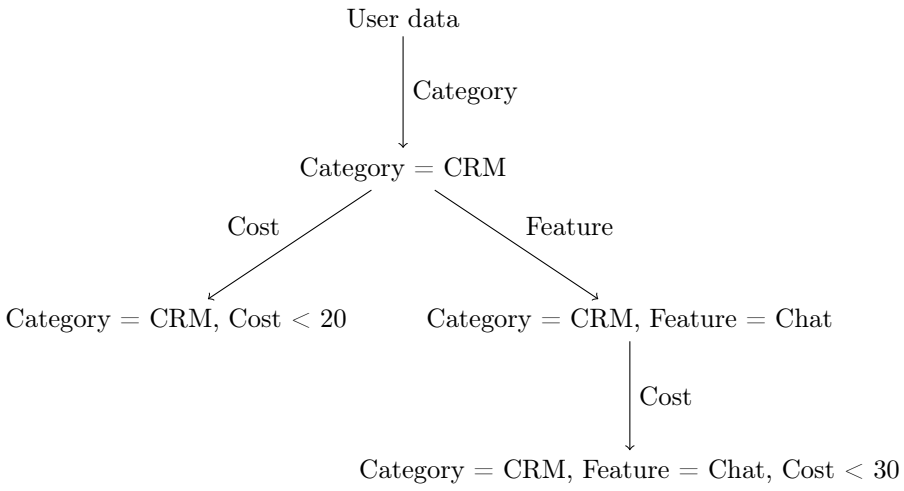

Figure 2: Graph representation of an hypothetical session

3.3.1 Similarity between two sink nodes

Table 1 represents the potential information available at a given node. The field are clustered in type according to their relevance for the similarity measure. For example, the most important type is the first one which contains the software category (*i.e.,* the type of software sought by the user).

Let s^1 and s^2 be the two sink nodes that we want to compare. Let s_i be the value of field i in a given node s. Let F_t ($t \in \{1, 2, 3, 4\}$) be the set of fields contained in type t. A coefficient c_i is associated to each type and each field.

The similarity between s^1 and s^2 is given by:

$$sim(s^1, s^2) = \sum_{t \in T} c_t \sum_{j \in F_t} c_f \; sim(\; s^1_f, s^2_f)$$

For each type

For each field

Add the similarity

To compute this similarity the value of all the coefficients as well as the similarity *sim* must be defined. First, we know that the coefficients associated to the types must satisfy: $c_1 > c_2 > c_3 > c_4$ (since the relevance of a type is inversely proportional to its value).

Secondly, the frequency of appearance of each field in the requests can be used to thinly define their coefficient. We can see in Figure 3 that the use of each field is heterogeneous. We postulate that the less frequent fields are the

Type	Field
1	Software category
2	Industry (company type)
	Number of users
	Features
	Budget
	User company
3	Certifications
	Department
	Position (of the user in the company)
	Activity sector
	Interoperability
4	Country
	Mobility
	Languages
	Support and formation
	Security
	Data localisation

Table 1: Field used to compute the similarity between two requests regrouped by type (from the most relevant type to the less relevant).

most relevant. Thus, the value of a coefficient could be indexed to the inverse of the frequency of its field. The only exception to this rule is the field "Software category" which is the most relevant but also the most frequent as it can not be empty in a request.

Similarly, the similarity $sim(s_i^1, s_i^2)$ of two values s_i^1 and s_i^2 of a given field i can be defined according to the frequency of these two values in the database. For example, Figure 4 represents the number of occurrences of each value of the field "Size of the company". Moreover, the similarity is equal to 0 if no value has been filled by the user for this field (*i.e.,* $sim(\emptyset, s_2^i) = sim(s_1^i, \emptyset) = sim(\emptyset, \emptyset) = 0$).

3.3.2 Similarity between two oriented graphs

The accuracy of the similarity between two sessions could be improved by not only taking into account two of its sink nodes, but the two whole graphs.

One possible way of doing so could be to merge the sink nodes of each oriented graph in one node and to use the similarity measure presented in section 3.3.1. However, this approach would not take into account the structure of the graphs. Further investigation are required in order to define such similarity measure.

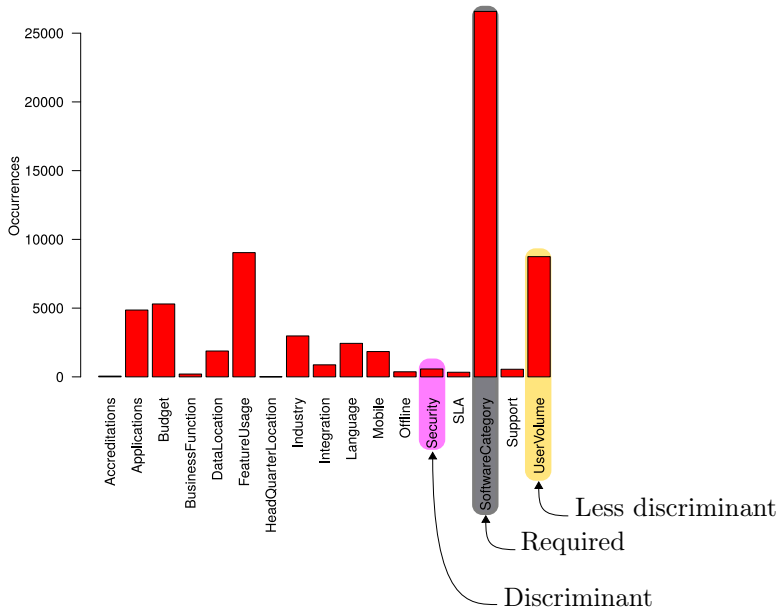

Figure 3: Number of occurrences of each field in the requests contained in the database.

4 Conclusion and future work

The method we propose implies to track the session of the users and store them. This change on the database can be easily implement and its cost seems reasonable to us in comparison to the benefit it will provide. The similarity measures we present will then allow to compare the sessions of users and address the problems identified by Cloud is Mine. We believe, up to the choice of relevant coefficients, it will lead to an efficient method to cluster the users and also to enhance the accuracy of the software or feature recommendation. In our opinion, it will also improve the rapidity of the recommendations given to an user.

The limitations of our work is of course the application of the proposed methods to real data. If our suggestion on the database is followed by Cloud is Mine, our first following task will then be to test our proposed method on the new database. The study of this new database will improve the estimation of the frequencies of the occurrences of each field in the request, and then help to choose the coefficients in the similarity measure. As future work, we propose to compare not only the similarity of two sink nodes but of the whole user sessions. This improvement of the method is based on the notion of distance between directed graphs.

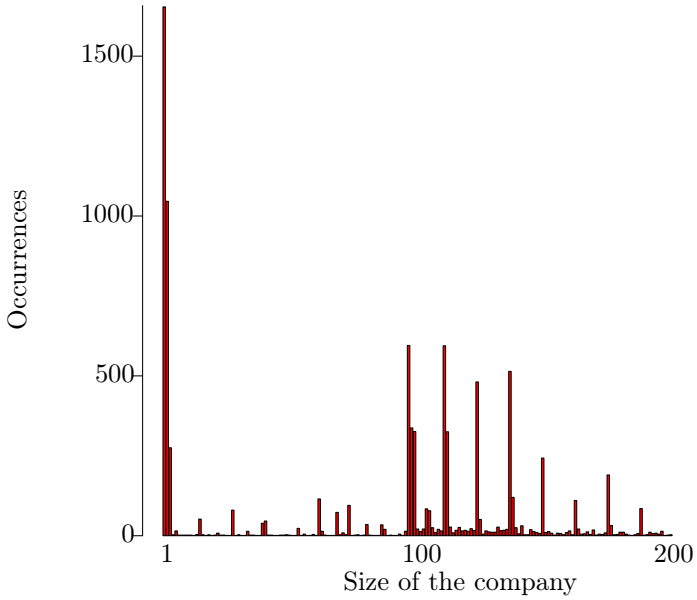

Figure 4: Number of occurrences of each value of the field "Size of the company" in the requests contained in the database.