Available online at www.sciencedirect.com
ScienceDirect
Fuzzy Sets and Systems ••• (••••) •••–•••
FUZZY sets and systems
www.elsevier.com/locate/fss

A soft computing approach to big data summarization

Grégory Smits [a,*], Olivier Pivert [a], Ronald R. Yager [b], Pierre Nerzic [a]

[a] *IRISA - University of Rennes 1, UMR 6074, Lannion, France*
[b] *Machine Intelligence Institute - IONA College, New Rochelle, NY, USA*

Received 27 June 2016; received in revised form 16 February 2018; accepted 28 February 2018

Abstract

The added value of a dataset lies in the knowledge a domain expert can extract from it. Considering the continuously increasing volume and velocity of these datasets, efficient tools have to be defined to generate meaningful, condensed and human-interpretable representations of big datasets. In the proposed approach, soft computing techniques are used to define an interface between the numerical and categorical space of data definition and the linguistic space of human reasoning. Based on the expert's own vocabulary about the data, a personal summary composed of linguistic terms is efficiently generated and graphically displayed as a term cloud offering a synthetic view of the data properties. Using dedicated indexing strategies linking data and their subjective linguistic rewritings, exploration functionalities are provided on top of the summary to let the user browse the data. Experimentations confirm that the space change operates in linear time wrt. the size of the dataset making the approach tractable on large scale data.
© 2018 Elsevier B.V. All rights reserved.

Keywords: Data personalisation; Linguistic summaries; Soft computing; Knowledge extraction; Visualization; Specificity measure

1. Introduction

Data analysis is a crucial task at the center of many professional activities and now constitutes a support for decision making, communicating and reporting. Considering the continuously increasing volume and velocity of these datasets, domain experts (as insurers, data journalists, communication managers, decision makers, etc.), who are not, most of the time, data or computer scientists, need efficient tools that help them turn data into useful knowledge. This explains the recent growing interest for so-called Agile Business Intelligence (ABI) systems that reconsider classical data integration processes to favor pragmatic approaches that make domain experts self-reliant in the analysis of raw data.

A dataset generally consists of a large collection of items described by numerical and categorical attributes. A way to assist experts in their fastidious task of data-to-knowledge translation is to define efficient strategies that generate meaningful, condensed and human-interpretable representations of the data. To be very useful, such representations should give an insight into the data properties and make it easy for the domain expert to identify the most representative

[*] Corresponding author.
E-mail addresses: gregory.smits@irisa.fr (G. Smits), olivier.pivert@irisa.fr (O. Pivert), yager@panix.com (R.R. Yager), pierre.nerzic@irisa.fr (P. Nerzic).

https://doi.org/10.1016/j.fss.2018.02.017
0165-0114/© 2018 Elsevier B.V. All rights reserved.

properties of the dataset. In this sense, when a dataset is so large that it cannot be easily perused and analyzed by the user, data summarization is of a particular interest to obtain a big picture of the data distribution on the different dimensions. Such a summary should also offer exploration functionalities to let the expert interactively browse the dataset from its summary and discover interesting properties possessed by different data subsets.

The approach proposed in this paper falls in with the original essence of soft computing as it aims to create an interface between the numerical/categorical space of data definition and the symbolic space of human reasoning. Based on an expert vocabulary materialized by fuzzy partitions and linguistic variables, efficient algorithms are provided to rewrite the data according to subjective linguistic terms taken from the expert's own vocabulary. To help the expert identify the linguistic terms that best describe a data subset, we provide a specificity-like measure that quantifies the representativity of a term wrt. the concerned data subset. The result of the rewriting process is materialized by a so-called rewriting vector that tells us about the distribution of the data according to the different linguistic terms of the expert's vocabulary. The fact that the numerical and categorical properties possessed by items of the dataset are all rewritten into linguistic terms makes it possible to envisage unified and global views of the data. Contrary to most of the existing approaches to ABI that are indeed only able to generate two or three-dimensional graphical views of the data, using e.g. histograms, charts or color maps, novel graphical representations of the linguistic rewriting vectors may be defined to provide the expert with a complete and concise view of the data, on several dimensions, using linguistic terms instead of numbers.

The main contributions of this paper concern the proposal of:

- efficient algorithms for data rewriting using linguistic terms taken from the expert's vocabulary,
- a measure that quantifies the informativeness of the linguistic terms wrt. a data subset,
- dedicated storage and indexing strategies,
- a condensed and human-interpretable graphical view of the data using linguistic terms.

The rest of the paper is organised as follows. After having compared the context and objective of the proposed approach wrt. to existing works (Sec. 2), Sec. 3 introduces some preliminary notions to this approach. Then, Sec. 4 details the rewriting process as well as storage strategies to associate items with their respective linguistic rewritings. Section 5 shows how the linguistic rewriting of a dataset may be displayed as a term cloud. Before concluding, Sec. 6 gives the results of an experimentation on a large dataset that confirm the efficiency of the proposed rewriting and visualization strategy.

1.1. Approach overview

Fig. 1 depicts the different steps of the approach proposed in this work. Faced with a raw dataset to analyze, the expert is first solicited to define, on attributes of interest, linguistic terms that will form his/her vocabulary and that will then be used to handle the data as well as the extracted knowledge. Each item from the dataset to analyze is first linguistically rewritten according to the user's vocabulary, these rewritings being stored in a database. A global rewriting vector is also computed for the whole dataset to represent the distribution of the items wrt. the different linguistic terms of the vocabulary. This dataset rewriting vector, that is very small compared to the size of the dataset, is then graphically rendered to the user, as a so-called term cloud, so as to provide a linguistic and subjective summary of the dataset. A term cloud offers a graphical view of the data distribution wrt. the terms of the user's vocabulary. Thanks to the fact that one stores the items and their associated rewritings in a database, efficient and intuitive data exploration functionalities are offered to the user to let him/her browse the data by successively selecting terms from the cloud. The cloud shown in Fig. 1 contains the terms that describe the items satisfying a previously selected property, here 'price is acceptable'. A measure is used to quantify the informativeness of each term appearing in a cloud so as to display the most informative terms in the eye-catching zone of the view, its center.

2. Related work

ABI is a recent issue raised by the need for domain experts to quickly analyze a large number of raw datasets and decide if their integration in the corporate datamart is useful or not. Driven by user needs, this issue has first been addressed by the main data management companies (e.g. Amazon with QuickSight, Pentaho's Agile BI, Microsoft

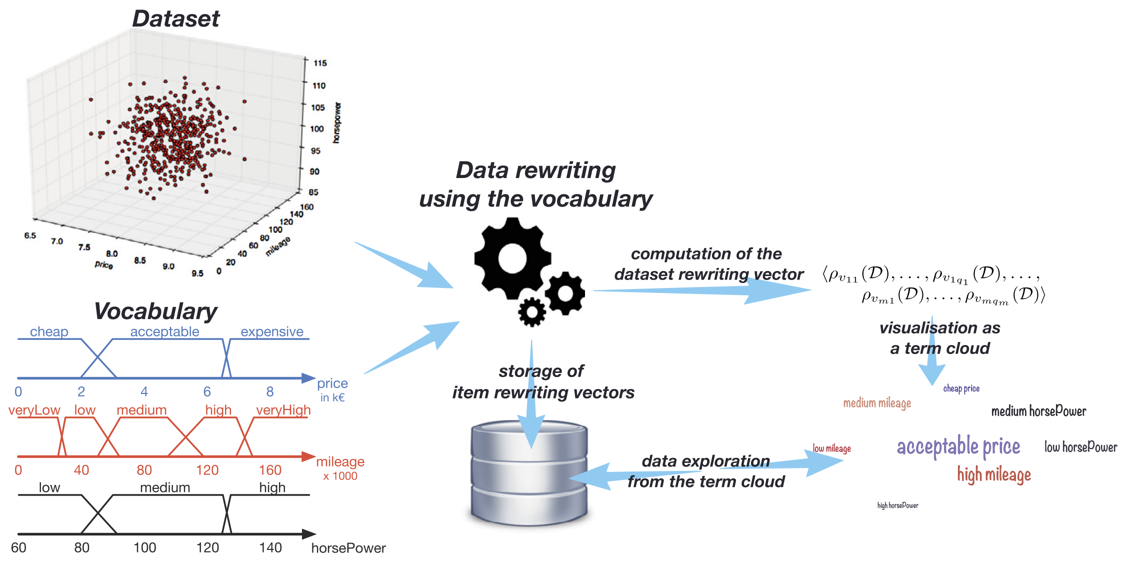

Dataset
price
mileage
horsepower
Vocabulary
cheap
acceptable
expensive
price in k€
veryLow
low
medium
high
veryHigh
mileage x 1000
low
medium
high
horsePower
Data rewriting using the vocabulary
computation of the dataset rewriting vector
$\langle \rho_{v_{11}}(\mathcal{D}), \ldots, \rho_{v_{1q_1}}(\mathcal{D}), \ldots, \rho_{v_{m1}}(\mathcal{D}), \ldots, \rho_{v_{mq_m}}(\mathcal{D})\rangle$
visualisation as a term cloud
storage of item rewriting vectors
data exploration from the term cloud
medium mileage
cheap price
medium horsePower
low mileage
acceptable price
low horsePower
high mileage
high horsePower

Fig. 1. Overview of the proposed data analysis approach.

with PowerBi, etc.) and more recently by academics [1]. The overall objective of an ABI approach is to extract useful and precise knowledge from raw data and to generate interpretable dashboards (as Oracle's dashboards, idashboards, etc.), most often using data mining [2] and statistical techniques combined with data visualization strategies [3].

After having concentrated most of the efforts on the storage of massive data, the DB community now has to face the issue of helping users make the most of large amounts of data. A way to assist domain experts in their data-to-knowledge translation task is to provide concise and human interpretable summaries that give an insight into the data properties. MacroBase [4] is an example of such a tool that provides a linear process to data analysis that aims at helping users identify the main trends in massive data. Another common strategy of data summarization relies on the use of clustering strategies [5] to automatically capture the main trends in a dataset. Despite the efficiency of clustering approaches to exhibit the structure and main properties of high-dimensional datasets, a problem arises when it comes to visualizing the knowledge produced. This is why dedicated visualization strategies have been defined to propose concise and user-friendly views about the data distributions on the different dimensions [6]. Some of these visualization strategies offer functionalities to perform data exploration [7]. Due to the fact that it is very difficult to graphically render, in an interpretable way, a dataset on more than three dimensions, data visualization interfaces generally rely on a first step to reduce the number of dimensions considered, using PCA [8] for instance. Then, users may select pairs of dimensions on which the data distribution will be displayed to identify possible correlations.

It has been shown in many applicative contexts, from database querying [9] to data summarization [10], that the personalization of the manipulated and extracted knowledge significantly improves its interpretability and informativeness. Much attention has been paid by the soft computing community to data summarization using personalized linguistic explanations [11]. E.g. the SaintEtiq system [12] also aims at providing users with interpretable explanations of a dataset but through a hierarchy of summaries discriminated by a typicality-like measure of informativeness. A formal flexible querying language has then been defined on top of the summaries provided by SaintEtiq [13] to execute both conjunctive and disjunctive queries. However, the context of ABI raises new issues that have not been addressed by the existing approaches of data summarization and exploration, as e.g. the interpretability and visualization of the linguistic summaries [14] and the scalability of the linguistic rewriting process. Another motivation for the use of soft computing to knowledge extraction comes from the availability of a large set of measures that may be used to aggregate or describe the knowledge extracted [15]. Among them, and especially in our context of linguistic summarization, the notion of specificity introduced in [16] is of a particular interest as it quantifies the informativeness of a linguistic explanation. This notion of specificity has for instance been extended to quantify the informativeness of a referring expression [17], an expression that describes the location of an object in a visual view. Here, this notion of specificity is extended to our concern so as to determine the extent to which a linguistic term, taken from the expert's vocabulary, is a main and interesting representative of a given data subset.

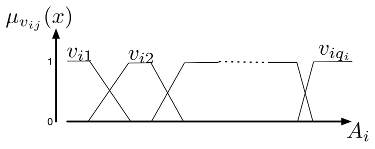


Fig. 2. Numerical partition.

Whereas many approaches exist to automatically extract association rules from data [18,19], the specificity-guided data exploration functionality we provide on top of the summaries also allows the expert to identify, through an interactive exploration of the term cloud, correlations between terms.

In the approach proposed, soft computing is first used to formalize, by means of fuzzy partitions, the subjective linguistic terms that form the expert's vocabulary and that are used to linguistically rewrite the data. The term cloud, built from the rewriting of a data subset, acts as a graphical summary of the items. Term clouds are generally used as a concise graphical representation of textual documents [20] or tags shared by users in a social network [21]. Despite the fact that it is an eye-catching representation of a document, the structure of a term cloud is meaningless and only depends on the size of the screen [20]. In the approach proposed, the terms involved in the cloud are spatially arranged according to their degree of informativeness so as to make the identification of the most interesting explanations of a dataset easy. This work opens a new direction of data exploration and knowledge extraction from linguistic summaries and their graphical representation.

3. Preliminaries

In this work, a dataset $\mathcal{D}$ is assumed to be of a tabular structure, and forms a multiset containing the description of n items $\{x_1, x_2, ..., x_n\}$. Each item, say x, belonging to a universe X, is described by m attributes $A_1, A_2, ..., A_m$, where $x.A_i$ denotes the value taken by the item x on the domain D_i of attribute A_i.

3.1. Domain expert's vocabulary

The approach takes as input a raw dataset and asks the domain expert to define his/her vocabulary on the attributes of interest. It is obviously a demanding task for the expert to express and formalize his/her vocabulary, but in this context of SSBI, the personalization of the data-to-knowledge translation process is crucial. A domain expert indeed rarely starts a data analysis task without any prior knowledge about the concerned corporate data and without any *a priori* idea about the kind of conclusion he/she would like to be able to draw, based on this data analysis. Defining the meaning of the linguistic terms is a way for the expert to integrate his/her own knowledge in the data analysis process [22].

Beside the fact that defining ones own vocabulary will ease the understanding of the extracted knowledge, it also gives the expert the possibility to specify the attributes/dimensions of interest from a list, that may be large, of features describing each item. This manual dimension selection step may be used as a complement to a statistical feature selection strategy [8]. So as to speed up the translation of data into knowledge and to improve the interpretability of the linguistic representation generated from the data, we advocate, in this context, a manual and subjective definition of the linguistic terms instead of an automatic elicitation of descriptors from data [23,24].

Concretely, defining a vocabulary consists in discretizing the definition domain of some interesting attributes using strong fuzzy partitions associated with linguistic variables [25]: formally, for attribute A_i, $i = 1..m$, q_i denotes the number of modalities involved in its partition and $\mathcal{V}_i = \{v_{i1}, \ldots v_{iq_i}\}$ their associated fuzzy sets. The strong partition property imposes that $\forall i = 1..m$, $\forall x \in D_i$, $\sum_{k=1}^{q_i} \mu_{v_{ik}}(x.A_i) = 1$. It is also imposed that an item cannot satisfy more than two modalities on each dimension, and these two modalities have to be adjacent in the case of numerical attributes. Each modality, say v_{ij}, is described linguistically by a label denoted by l_{ij}. These linguistic labels, that generally correspond to adjectives of a natural language (e.g. 'cheap', 'recent', 'high', 'insignificant', etc), form the expert's vocabulary. Fig. 2 and Table 1 illustrate partitions defined respectively on a numerical and a categorical domain.

The first step toward the construction of a linguistic summary and its graphical view is to rewrite each item of the dataset according to the expert's vocabulary.

Table 1
Categorical partition.

	$\mathcal{D}_i$			
$\mathcal{V}_i$	$e_1 \in \mathcal{D}_i$	$e_2 \in \mathcal{D}_i$	...	$e_h \in \mathcal{D}_i$
v_{i1}	$\mu_{v_{i1}}(e_1)$	$\mu_{v_{i1}}(e_2)$	...	$\mu_{v_{i1}}(e_h)$
v_{i2}	$\mu_{v_{i2}}(e_1)$	$\mu_{v_{i2}}(e_2)$	...	$\mu_{v_{i2}}(e_h)$
...	...	...	...	
v_{iq_i}	$\mu_{v_{iq_i}}(e_1)$	$\mu_{v_{iq_i}}(e_2)$	...	$\mu_{v_{iq_i}}(e_h)$

3.2. *Data rewriting using terms of the vocabulary*

The data-to-knowledge translation process proposed in this paper relies on a first step aimed at rewriting the data in terms of the expert's vocabulary. This rewriting step consists in computing vectors of linguistic labels taken from the expert's vocabulary at the item and dataset levels.

Definition 1 *(item rewriting vector).* An item x may be rewritten in terms of a vocabulary $\mathcal{V}$ as a vector of $\sum_{k=1}^{m} q_k$ membership degrees. This vector, denoted by $RV_x^{\mathcal{V}}$, is of the form $\langle \mu_{v_{11}}(x.A_1)\ldots, \mu_{v_{1q_1}}(x.A_1), \ldots, \mu_{v_{m1}}(x.A_m), \ldots, \mu_{v_{mq_m}}(x.A_m)\rangle$, among which at most $2 \times m$ degrees are non-zero.

Definition 2 *(set rewriting vector).* A set rewriting vector represents how much each linguistic term covers a set $s \subseteq \mathcal{D}$ of items. As an aggregation of item rewriting vectors, a set rewriting vector tells us about the distribution of the vocabulary elements on the items belonging to s. Such a vector, denoted by $RV_s^{\mathcal{V}}$, is formalized in the following way:

$$\langle \rho_{v_{11}}(s), \ldots, \rho_{v_{1q_1}}(s), \ldots, \rho_{v_{m1}}(s), \ldots, \rho_{v_{mq_m}}(s)\rangle,$$

where $\rho_{v_{ij}}(s) = \frac{\sum_{x\in s} \mu_{v_{ij}}(x)}{|s|}$.

Such vectors provide a description of an item and a dataset, respectively, in the linguistic space. It is worth noticing that the dataset rewriting vector may be stored in a non-normalized way, i.e. by not dividing the sum of the satisfaction degrees by the cardinality of s, so as to be able to take into account evolutions of the dataset. Modifications of the vocabulary would imply, on the other hand, to recalculate the dataset rewriting vector.

4. From the description space to the summarization space

Two implementation strategies are described in this section, and experimented in Sec. 6, the aim being to generate the item rewriting vectors and the global rewriting vector of a data subset to analyze. The first implementation relies on a classical sequential processing of the dataset to summarize that can be executed locally on a single computer, whereas the second one distributes the rewriting process over a cluster of machines.

4.1. *Sequential computation of the rewriting vectors*

In the first implementation strategy described by Algorithm 1, items of the dataset $\mathcal{D}$ are processed one by one (line 2) to first build their rewriting vector (line 5) that is then used to directly update the dataset rewriting vector. So as to provide data exploration strategies based on the linguistic summaries (Sec. 4.4), the rewriting vectors have to be stored in such a way that the items satisfying a given term may be efficiently retrieved (Sec. 4.3).

This rewriting process performs in the worst case in $|\mathcal{D}| \times |\mathcal{V}|$ steps. The second loop (line 4) may indeed be broken prematurely when the cumulative satisfaction degrees of an item x wrt. elements of a given partition $\mathcal{V}_i$ is equal to 1 (i.e. when $\sum_{j=1\ldots q_i} \mu_{v_{ij}}(x) = 1$), as in this case one knows that the remaining modalities of the concerned partition are not satisfied at all by the item. Considering the fact that $|\mathcal{V}|$ is generally of a reasonable size and much smaller than $|\mathcal{D}|$, Algorithm 1 thus performs in linear time wrt. $|\mathcal{D}|$. Considering the fact that the initial dataset is not sorted nor structured, it is worth noticing that a randomly selected subset of the whole dataset may be used to generate the initial summary, thus leading to a sublinear complexity. The goal of this first summary is to generate a linguistic overview of the data that will be used by the expert to start a data exploration process.

```
Data: dataset $\mathcal{D}$, expert's vocabulary $\mathcal{V}$
Result: Dataset Rewriting Vector $\langle \rho_{v_{11}}(\mathcal{D}), \ldots, \rho_{v_{1q_1}}(\mathcal{D}), \ldots, \rho_{v_{m1}}(\mathcal{D}), \ldots, \rho_{v_{mq_m}}(\mathcal{D})\rangle$
$RV_{\mathcal{D}}^{\mathcal{V}} \leftarrow \emptyset$;
forall the $x \in \mathcal{D}$ do
    $RV_x^{\mathcal{V}} \leftarrow \emptyset$
    forall the $v \in \mathcal{V}$ do
        compute $\mu_v(x)$
        add $\mu_v(x)$ in $RV_x^{\mathcal{V}}$
        if $v \notin RV_{\mathcal{D}}^{\mathcal{V}}$ then
            $RV_{\mathcal{D}}^{\mathcal{V}}[v] = 0.0$
        end
        $RV_{\mathcal{D}}^{\mathcal{V}}[v] = RV_{\mathcal{D}}^{\mathcal{V}}[v] + \mu_v(x)$
    end
    $store(x, RV_x^{\mathcal{V}})$
end
forall the $v \in \mathcal{V}$ do
    $RV_{\mathcal{D}}^{\mathcal{V}}[v] = RV_{\mathcal{D}}^{\mathcal{V}}[v]/|\mathcal{D}|$
end
return $RV_{\mathcal{D}}^{\mathcal{V}}$
```

Algorithm 1: Sequential computation of the dataset rewriting vector.

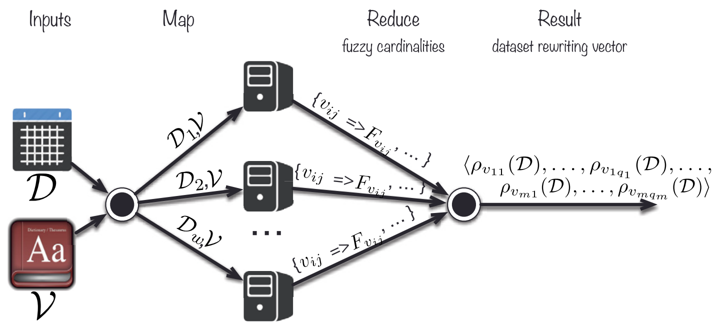

Inputs
Map
Reduce
fuzzy cardinalities
Result
dataset rewriting vector
$\mathcal{D}$
Dictionary / Thesaurus
Aa
$\mathcal{V}$
$\mathcal{D}_1,\mathcal{V}$
$\mathcal{D}_2,\mathcal{V}$
$\mathcal{D}_w,\mathcal{V}$
$\{v_{ij} => F_{v_{ij}}, \ldots\}$
$\{v_{ij} => F_{v_{ij}}, \ldots\}$
$\{v_{ij} => F_{v_{ij}}, \ldots\}$
$\langle \rho_{v_{11}}(\mathcal{D}), \ldots, \rho_{v_{1q_1}}(\mathcal{D}), \ldots, \rho_{v_{m1}}(\mathcal{D}), \ldots, \rho_{v_{mq_m}}(\mathcal{D})\rangle$

Fig. 3. Distributed computation of the dataset rewriting vector.

4.2. Distributed computation of the rewriting vector

When dealing with real big datasets containing millions of items, sequential algorithms and local storage strategies are not realistic and have to be reconsidered in favor of processing and storage strategies distributed on multiple processing units.

As illustrated by Fig. 3, the dataset $\mathcal{D}$ to summarize is distributed on the w computers that form a cluster. Each processing unit stores a subset $\mathcal{D}_i$ of $\mathcal{D}$: $\bigcup \mathcal{D}_i = \mathcal{D}$. Using a map-reduce architecture [26], each subset $\mathcal{D}_i$ is processed by a dedicated processing unit whose goal is to build the rewriting vectors (Def. 1) of the items it is in charge of. The partial and local rewriting vectors of the data subsets processed by the different nodes of the cluster are sent to the "reduce node" and aggregated in order to obtain the final dataset rewriting vector (Def. 2) that constitutes the summary of the dataset.

The algorithm that performs this rewriting in a map-reduce way is very simple and only relies on three steps:

1. the rewriting of each item $x \in \mathcal{D}$:
$$\{RV_x^{\mathcal{V}}\} = map(lambda\, x : rewrite(x, \mathcal{V}), \mathcal{D}),$$
where $rewrite(x \in \mathcal{D}, \mathcal{V})$ is a function ($x \in \mathcal{D} \times \mathcal{V} \rightarrow [0,1]^{|\mathcal{V}|}$) that returns the rewriting vector ($RV_x^{\mathcal{V}}$) of x wrt. $\mathcal{V}$,
2. the aggregation of the item rewriting vectors into a non-normalized dataset rewriting vector:
$$RV'^{\mathcal{V}}_{\mathcal{D}} = reduce(lambda\, \mathcal{D}_i, \mathcal{D}_j : map(lambda\, \mu, \mu' : \mu + \mu', \mathcal{D}_i, \mathcal{D}_j), \{RV_x^{\mathcal{V}}\}),$$

Table 2
Key/value-based storage strategy for the item rewriting vectors.

Key (term)	Value (item identifiers)
medium_horsePower_1	$\langle rowID_1, \ldots, rowID_i, \rangle$
medium_horsePower_2	$\langle rowID_j, \ldots, rowID_k, \rangle$
...	...

3. the normalization of the dataset rewriting vector:

$$RV_{\mathcal{D}}^{\mathcal{V}} = map(lambda\, \rho_i : \rho_i / |\mathcal{D}|, RV_{\mathcal{D}}^{\prime\mathcal{V}}).$$

To make the most of the distributed environment, the second step, whose aim is to aggregate the results returned by the different processing units, is partially achieved by each processing unit, in a step that uses Hadoop *combiner*. The final *reduce* statement is thus performed on w partial rewriting vectors. Again, considering the fact that $|\mathcal{V}|$ is generally of a reasonable size and much smaller than $|\mathcal{D}|$, Algorithm 1 thus performs in linear time wrt. $|\mathcal{D}|$ and more precisely wrt. $\frac{|\mathcal{D}|}{w}$.

4.3. Storage of the rewriting vectors

To provide data exploration functionalities on top of the linguistic summaries (Sec. 4.4), it is necessary to link each item to its rewriting vector so as to be able to move back from the linguistic space to the numerical/categorical definition space. The storage strategy used for the rewriting vectors has to make it possible to efficiently retrieve the subset of items that somewhat satisfy a given term of the vocabulary. Three storage strategies are presented in this section and compared in Section 6.

4.3.1. Key/value storage strategy

The first strategy is vocabulary-driven and consists in associating with each term, considered as a key, the list of item identifiers, forming the value, that satisfy it. However, as a partition that discretizes the definition domain of an attribute generally involves only a few modalities (around 4), this key-value storage strategy cannot be directly implemented as the lists of item identifiers may be too large to be stored and processed efficiently. A classical solution [27] to overcome this problem is to set a maximal size for a value list and to duplicate the keys (i.e. terms) concatenated with an identifier so as to make them unique. Table 2 illustrates this principle considering the term *medium_horsePower*. Distributed storage systems, as Hadoop HBASE e.g., offer a functionality to efficiently retrieve all the keys in a given range.

4.3.2. Bitmap storage strategy

A second strategy that may be envisaged to store the rewriting vectors consists in building a bitmap vector for each item that completes the initial raw description of the item. Such a bitmap vector contains as many bits as there are terms in the vocabulary, the first bit indicates if the item somewhat satisfies the first modality of the partition discretizing attribute A_1, the second bit the second modality of this same partition, and so on. Due to the fact that strong fuzzy partitions are used (Sec. 3.1), the bitmap vector is sparse as it may contains up to $2m$ bits set to 1 (m is the number of attributes on which the expert has defined a linguistic vocabulary). Efficient compression strategies may be used to reduce the size needed to store the bitmap vectors without penalizing the query execution efficiency [27], but these well-known techniques are not studied in this article. Table 3 illustrates how the bitmap vectors associated with each item are stored in an additional column. Binary operators are used during a distributed sequential scan to retrieve the items that satisfy a given term, or in other words the items having a bit set to 1 in their rewriting vector for the concerned term.

4.3.3. A hybrid key/value-bitmap storage strategy

Due to the fact that an item may satisfy up to two adjacent terms of each partition defined over a numerical domain, the number of distinct rewriting vectors one may obtain in theory is close to $(2 \times N - 1)^K$, where K is the average

Table 3
Bitmap storage strategy for the item rewriting vectors.

RowID	A_1	...	A_m	Bitmap rewriting vector
t_1	$t_1.A_1$	...	$t_1.A_m$	$\langle 011010001100010 \rangle$
t_2	$t_2.A_1$	...	$t_2.A_m$	$\langle 110010001000010 \rangle$
...	...	...	...	...

Table 4
Hybrid key/value-bitmap storage strategy of the item rewriting vectors.

Bitmap rewriting vector	RowIDs interval
$\langle 011010001100010 \rangle$	$[1, i]$
$\langle 110010001000010 \rangle$	$[i_{+1}, j]$
$\langle 110010001000011 \rangle$	$[j_{+1}, k]$
...	...

number of terms per partition and discarding the case of partitions defined over categorical domains. However, it has been shown in [28] that in practice the number of possible linguistic rewritings is significantly smaller than the theoretical upper bound. Some combinations of linguistic descriptions cannot indeed be observed in the real world, e.g. there is no car with a *veryHigh horsePower* and a *veryLow consumption*. Moreover, the fuzzy-set-based modelling of expert vocabularies introduces a legitimate linguistic indistinguishability between items whose initial numerical/categorical values are close. It is indeed reasonable to use a same term, e.g. *high consumption*, to linguistically qualify two consumption values of respectively 25.4 and 27.1 liters per mile.

To make the most of the two aforementioned properties, a third hybrid storage strategy is proposed. The idea is to rank and store the items according to their bitmap rewriting vectors so that items satisfying the same terms of the vocabulary have contiguous row identifiers. Then, a key/value table may be used to link a bitmap vector (the key) to the lower and upper bounds (value) of the row identifiers that are rewritten by this vector (Table 4).

4.4. Data exploration on top of linguistic summaries

Besides the fact that the linguistic summary (i.e. dataset rewriting vector) gives a synthetic and subjective overview of the dataset, it may also serve as a starting point for data exploration. From the dataset rewriting vector, the expert in charge of the data analysis task first identifies a term of interest he/she would like to know more about. Once a first term selected from the linguistic summary, say v, one retrieves the set $\mathcal{D}_v$, based on the storage strategy in use (Sec. 4.3):

$$\mathcal{D}_v = \{x \in \mathcal{D} | \mu_v(x) > 0\}.$$

The subset $\mathcal{D}_v$ is rewritten according to the expert's vocabulary in order to obtain its rewriting vector $RV^{\mathcal{V}}_{\mathcal{D}_v}$. To quantify how much a term, say v', covers the set $\mathcal{D}_v$, one combines the satisfactions degrees of each item in $\mathcal{D}_v$ wrt. v and v' using the min t-norm:

$$\rho_{v'}(\mathcal{D}_v) = \frac{\sum_{x \in \mathcal{D}_v} \min(\mu_v(x), \mu_{v'}(x))}{|\mathcal{D}_v|}.$$

The expert may continue the data exploration process by selecting a term in the rewriting vector $RV^{\mathcal{V}}_{\mathcal{D}_v}$ until he/she has built a data subset of interest.

5. Representativity-driven summary visualization

In this section we show how to turn a dataset rewriting vector into an interpretable summary of the dataset. Whereas most of the existing approaches to data summarization using a subjective vocabulary generate natural language descriptions of the data, we advocate a graphical representation that gives a concise view of the whole dataset on all the

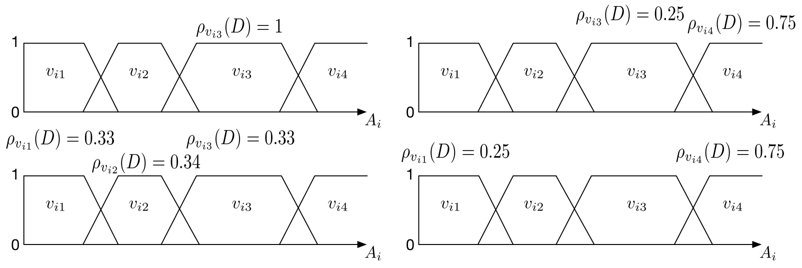

$\rho_{v_{i3}}(D) = 1$
$\rho_{v_{i3}}(D) = 0.25$
$\rho_{v_{i4}}(D) = 0.75$
$\rho_{v_{i1}}(D) = 0.33$
$\rho_{v_{i3}}(D) = 0.33$
$\rho_{v_{i2}}(D) = 0.34$
$\rho_{v_{i1}}(D) = 0.25$
$\rho_{v_{i4}}(D) = 0.75$
1
0
v_{i1}
v_{i2}
v_{i3}
v_{i4}
A_i

Fig. 4. Examples for the quantification of the representativity of the proposition "$\mathcal{D}\, is\, v$".

dimensions of interest. To reach this goal, this section addresses two underlying tasks: How to help the expert identify the linguistic terms that are the most informative for describing a data subset? How to graphically render the linguistic rewriting of a dataset and how to provide data exploration functionalities on top of it?

5.1. Representativity of the statement $\mathcal{D}$ is v

To be very useful, a data summary should help domain experts quickly identify the most informative properties of the dataset on each considered dimension. To do so, we define in this section a measure quantifying the extent to which a linguistic term, that covers a subset of the concerned data, is a good representative of the properties possessed by the data on a given dimension. The semantics of the measure we propose for quantifying this representativity is the following. One says that a linguistic term, say v, perfectly represents a data subset on a given dimension, if all of the items are completely rewritten as v. On the contrary, a term that does not cover any item from the dataset is obviously not representative at all. In between these two opposite situations, one would like to discriminate between different terms from the same partition that somewhat cover the dataset. The basic idea is the following: the more a term covers the dataset and the closer it is from the other terms of the same partition that also cover the data, the more representative.

Specificity measures, introduced in [16], are used to quantify the informativeness of a linguistic statement of the form *A is v*. In [17], this notion of specificity has been extended to quantify the informativeness of a referring expression. In its initial definition, the specificity of a statement *A is v* is maximal if v is completely precise and refers to only one possible value. E.g. saying that 'Paul is 18' is more informative than saying that 'Paul is around 20'. In the context of referring expression generation, the highest specificity degree is obtained when a referring expression describes, without ambiguity, only one object in a visual scene. In our context, the specificity of a linguistic term is maximal when all of the items completely satisfy it and no other term in the same dimension.

However, to be able to discriminate between two terms that cover a dataset in the same proportions, we combine two notions: the coverage of the concerned dataset and the distance of the term wrt. the other covering terms on the same dimension.

To illustrate the meaning of such a representativity measure, let us consider that 90% of the cars described in the dataset have a *veryLow Mileage* and that the remaining cars have, in an equal proportion (5% e.g.), a *low mileage* or a *veryHigh mileage*. The term *veryLow Mileage* is obviously considered as the most informative one because of its high coverage of the dataset, but we argue that *low mileage* should be considered more informative than *veryHigh mileage* because of its closeness wrt. the main description of the data (i.e. *veryLow Mileage*).

Fig. 4 gives some illustrative situations one would like to differentiate. The top left situation in Fig. 4 constitutes the most informative situation where the whole dataset may be summarized by l_{i3}, the linguistic term attached to the partition element v_{i3}. In the bottom left case, v_{i1}, v_{i2} and v_{i3} summarize $\mathcal{D}$ on attribute A_i in somehow the same proportion. However, v_{i2} has to be considered as more representative than v_{i1} and v_{i3} because of its central position wrt. the other explanations. In the two examples on the right part of Fig. 4, v_{i4} is, in terms of coverage, the main explanation for $\mathcal{D}$ on A_i as $\rho_{v_{i4}}(\mathcal{D}) = 0.75$. However, for the case on the top, v_{i4} constitutes a more representative summarizer of the dataset as the other possible explanation (v_{i3}) describes a very close property. One may indeed, in this last case, conclude that the whole dataset is v_{i4} or around.

Concretely, one extends the coverage of a term by taking into account its neighborhood so as to identify adjacent terms that, combined together, form a major explanation of the dataset. A statement $\mathcal{D}$ *is* v is thus individually characterized by its coverage of the dataset, and is completed by a degree of representativity.

The representativity of v wrt. $\mathcal{D}$, which is denoted by $R(v, \mathcal{D})$, is maximal on a given dimension if all the items of $\mathcal{D}$ are completely described by v and thus not covered at all by any other term of the same partition. In this case, v completely represents $\mathcal{D}$ on the concerned dimension. The informativeness of a proposition $\mathcal{D}$ *is* v is minimal if no item of $\mathcal{D}$ satisfies v, even to a low degree. Finally, the closer the term v is to the other explanation(s), the more informative.

The function that quantifies the representativity of the statement $\mathcal{D}\,is\,v_{ij}$ is of the form $R : (v_{ij} \in \mathcal{V} \times \mathcal{D}) \to [0, 1]$, and may be described by three properties, as done in [16]:

1. $R(v_{ij}, \mathcal{D}) = 1$ iff. $\rho_{v_{ij}}(\mathcal{D}) = 1$ and $\forall v_{ik}, k = 1..q_j, k \neq j,\ \rho_{v_{ik}}(\mathcal{D}) = 0$,
2. $R(v_{ij}, \mathcal{D}) = 0$ iff. $\rho_{v_{ij}}(\mathcal{D}) = 0$,
3. if $\rho_{v_{ij}}(\mathcal{D}) = \rho_{v_{ik}}(\mathcal{D}) \wedge d(v_{ij}, \mathcal{D}) \leq d(v_{ik}, \mathcal{D})$ then $R(v_{ij}, \mathcal{D}) \geq R(v_{ik}, \mathcal{D})$,

where $d(v, \mathcal{D})$ measures the distance between v and the other terms of the same partition taking into account their relative coverage of $\mathcal{D}$. Inspired by the distance measure proposed in [22], the distance $d(v_{ij}, \mathcal{D})$ is all the lower as v_{ij} covers a large part of $\mathcal{D}$ and is close to the other explanations for attribute A_i:

$$d(v_{ij}, \mathcal{D}) = \frac{1}{X} \sum_{k=1..q_i,\, \rho_{v_{ik}}(\mathcal{D})>0} \frac{|(j - \rho_{v_{ij}}(\mathcal{D})) - (k - \rho_{v_{ik}}(\mathcal{D}))|}{q_i}, \tag{1}$$

where X is the number of terms (v') of $\mathcal{V}_i$ such that $\rho_{v'}(\mathcal{D}) > 0$. A possible definition of a representativity function at the item level is the following:

$$R(v, \mathcal{D}) = \top(\rho_v(\mathcal{D}), 1 - d(v, \mathcal{D})), \tag{2}$$

where $\top$ is any t-norm such that $T(x, y) > 0$ when $x \neq 0$ and $y \neq 0$. The probabilistic t-norm $T(x, y) = x \times y$ is used in this work.

5.2. View of a summary as a term cloud

A summary is composed of a set of linguistic terms taken from the expert's vocabulary, each term (say v_{ij}) being characterised by three measures quantifying:

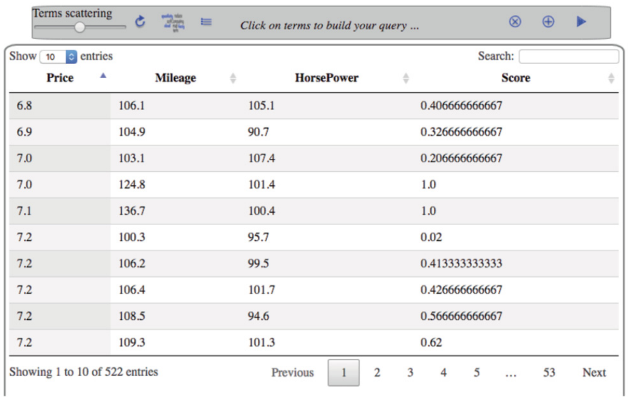
- its coverage wrt. $\mathcal{D}$ ($\rho_{v_{ij}}(\mathcal{D})$),
- its representativity wrt. to the other explanations on the same dimension ($R(v_{ij}, \mathcal{D})$),
- and its position in the partition used to emphasize extreme values (j).

We propose to display a summary as a cloud of linguistic terms. A term is graphically materialized by a textual tag, whose size, color and place in the cloud are proportional to its coverage, representativity and position (in the partition). Thus, contrary to classical tag clouds where the position and distance between terms are meaningless [20], the structure of the generated cloud is determined according to the relative representativity of its terms so as to make easier the identification of the more (resp. least) informative explanations.

Let S and s be respectively the predefined maximal and minimal font sizes for terms, then the size of a term l_{ij} in the cloud is set to $max(s, \rho_{v_{ij}}(\mathcal{D}) \times S)$.

To highlight the most informative terms of the summary, we suggest to display those having the highest specificity degree at the center of the cloud, which is known to be the first eye-catching zone of a picture [20]. Terms involved in the summary are thus ranked in decreasing order of their representativity, and then arranged on a spiral curve (Fig. 6) whose equation is:

$$\begin{bmatrix} x \\ y \end{bmatrix} = z \times u \begin{bmatrix} cos\, u \\ sin\, u \end{bmatrix},$$


Terms scattering
Click on terms to build your query ...
Show 10 entries
Search:

Price	Mileage	HorsePower	Score
6.8	106.1	105.1	0.406666666667
6.9	104.9	90.7	0.326666666667
7.0	103.1	107.4	0.206666666667
7.0	124.8	101.4	1.0
7.1	136.7	100.4	1.0
7.2	100.3	95.7	0.02
7.2	106.2	99.5	0.413333333333
7.2	106.4	101.7	0.426666666667
7.2	108.5	94.6	0.566666666667
7.2	109.3	101.3	0.62

Showing 1 to 10 of 522 entries

Previous 1 2 3 4 5 ... 53 Next

Fig. 5. Description of the cars having an ‘acceptable price’.

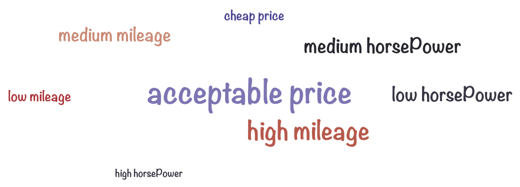


Fig. 6. Summarization of the cars having an ‘acceptable price’.

where u is the index of the term in the list, rank-ordered according to representativity. The most specific term of the summary is placed at the origin $(0, 0)$ of the Cartesian plane. Using the indices instead of the specificity degrees for computing the coordinates of the terms linearizes the gaps between the terms but leads to a more interpretable view as it avoids an overlapping of terms having very close representativity degrees. z is a zooming parameter that may be used to expand or shrink the spiral, and this parameter is initially set such that no term overlaps too much with others.

To differentiate the explanations on the different dimensions, a color is associated with each dimension and the position j of a term v_{ij} in the partition $\mathcal{V}_j$ determines the opacity of its color. The idea is to highlight the terms corresponding to extreme values of a definition domain, considering that they generally constitute surprising, thus potentially interesting, explanations. The opacity of a term v_{ij} is thus defined according to j and wrt. q_i (i.e. the number of elements in the partition $\mathcal{V}_i$). If $j \leq \frac{q_i}{2}$ (resp. $j > \frac{q_i}{2}$) then the opacity is set to $\frac{100}{j}\%$ (resp. $\frac{100}{q_i - j + 1}\%$).

Each term in the cloud acts as a selection criterion to focus on data properties of interest. A click on a term raises a query to retrieve the items satisfying the concerned condition from the database . The result set may be displayed in its initial numerical/categorical space of description (Fig. 5) or in the linguistic space of summarization (Fig. 5). Figs. 5 and 6 respectively show the result of a query, executed from the term cloud illustrated in Fig. 6, looking for cars of an acceptable price.

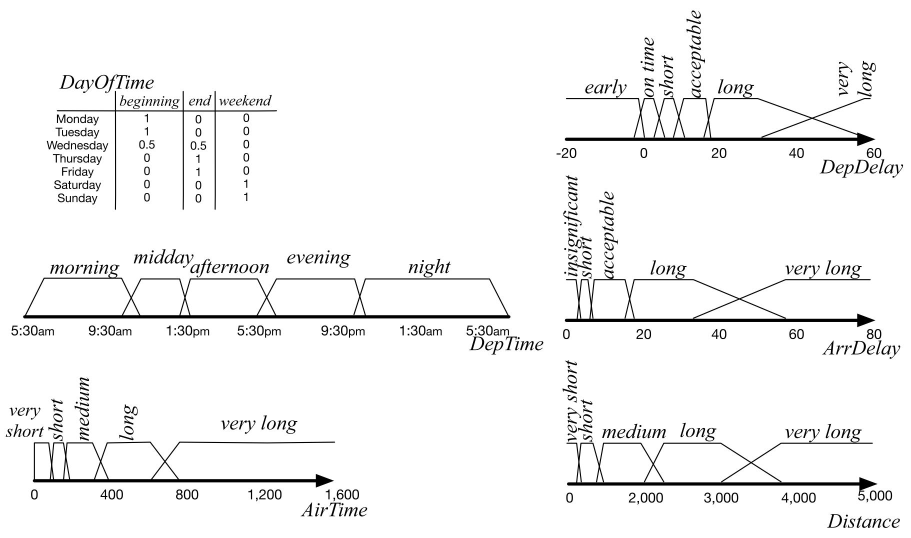
6. Experimentations

The two approaches (sequential and distributed) to the linguistic rewriting of data have been implemented in Java, the sequential one running on an Intel(R) Core(TM) i5-4590 CPU @ 3.30GHz-4 cores with 6GB of RAM, and the distributed one on top of a Hadoop[1] cluster of 5 machines (one master and 4 slaves) of the same configuration. In this last case, data are stored in a HBASE table.

Experimentations have been conducted to assess the efficiency of:

- the linguistic rewriting method in terms of processing time,
- the storage of the rewriting vectors in terms of space and access.

[1] Hadoop 2.7.1.



DayOfTime

	beginning	end	weekend
Monday	1	0	0
Tuesday	1	0	0
Wednesday	0.5	0.5	0
Thursday	0	1	0
Friday	0	1	0
Saturday	0	0	1
Sunday	0	0	1

Fig. 7. Extract of the expert vocabulary for the US flights dataset.

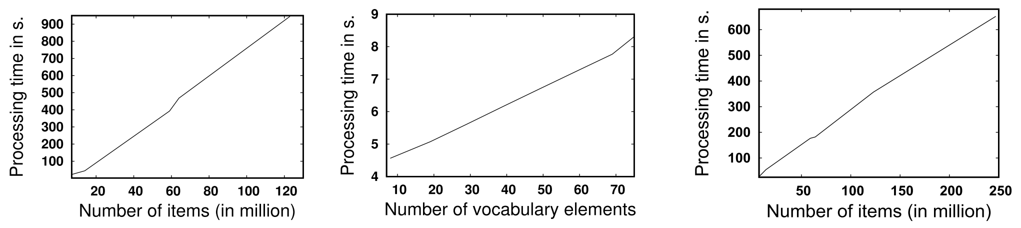


Fig. 8. (Left) Sequential rewriting/Dataset, (center) Sequential rewriting/Vocabulary and (right) Distributed rewriting/Dataset.

A real dataset about commercial flights published by the Research and Innovative Technology Administration (RITA) and Bureau of Transportation Statistics (BTS) has been used.[2] This dataset consists of 22 CSV files (for a total of 11GB) detailing 123,534,991 flights throughout the United States from 1987 to 2008.

A common sense vocabulary involving 75 terms has been defined for the attributes: {*DayOfWeek, DepTime, AirTime, ArrDelay, DepDelay, Distance, Month, DayOfMonth, TaxiIn, TaxiOut, CarrierDelay, WeatherDelay, SecurityDelay, LateAircraftDelay, Origin, Dest*}, the other attributes being discarded for this experimentation. Partitions and their associated linguistic variables defined on the attributes *DayOfWeek*, *Month*, *DayOfMonth*, *Origin* and *Dest* indicate respectively the part of the week concerned (beginning, middle, end, weekend), the season, the part of the month, and for the last two, airports are described as *big*, *small*, etc., according to their number of passengers. Fig. 7 illustrates a part of this vocabulary.

6.1. Linguistic rewriting of a dataset

Fig. 8 (left) shows that the time taken by the sequential rewriting algorithm (Section 4.1) linearly increases wrt. the size of the dataset $\mathcal{D}$. Using a same file of about 7 million items and vocabularies of different sizes, Fig. 8 (center) illustrates that the time needed to perform a sequential rewriting also linearly depends on the size of the vocabulary considered. The observed times confirm the efficiency of the space change and that large datasets may be efficiently summarized by the sequential algorithm, at least as long as they can fit in memory.

[2] http://stat-computing.org/dataexpo/2009/the-data.html.

6.2. Using the distributed strategy of rewriting

A second experimentation has then been conducted on a Hadoop cluster of 4 processing nodes. Considering the relatively modest performances of each node, the goal of this experimentation is only to show the scalability of the approach and not the efficiency of this specific cluster architecture. As explained in Section 4.2, Hadoop combiners have been used to distribute the semi-reducers to obtain a partial rewriting vector on each node. The final reduce step thus only consists in merging 4 rewriting vectors.

Fig. 8 (right) gives the times needed to rewrite datasets of various sizes using the map-reduce-based algorithm (Sec. 4.2) and a vocabulary composed of 75 modalities. This experimentation shows that the complexity of the rewriting process linearly increases wrt. the size of the dataset and the size of the vocabulary, but that linearly decreases wrt. the number of processing units. The fact that the items rewriting and the construction of local rewriting vectors are performed by Hadoop mappers and combiners respectively makes it possible to improve the efficiency of the process by adding new computation units to the cluster. The reduce step, that can only be performed by the master node, is negligible, in terms of computation time, compared with the part of the process that is distributed onto the different slaves. The reduce step takes indeed around 3% of the total computation time. The rewriting of 7 million items takes in average 27 seconds (average computed after 10 runs) during which only 0.8 second is used by the final reducer. Thus, doubling the number of nodes commissioned in the cluster, almost divides by two the overall processing time. Based on this result, one can say that, using a Hadoop cluster of 40 computers as those used in this experimentation, it would only take less than 3 seconds to summarize a dataset containing 7 million of items and around a minute to rewrite 140 million of items.

It is worth recalling that the representativity degree of each term is computed based on the dataset rewriting vector only. Due to the fact that the number of terms in the vocabulary is generally rather low (between 50 to 100), the time taken by this step, whose complexity is quadratic wrt. the size of the vocabulary, is negligible.

6.3. Storage of the item rewriting vectors

Three storage strategies have been proposed in Sec. 4.4 to store the item rewriting vectors. The stored rewriting vectors are scanned to identify the items satisfying a certain term selected in the term cloud by the expert. We have compared the pros and cons of each of these strategies wrt. three criteria: i) the additional space needed to store the vectors, ii) how efficient it is to use them for retrieving the subset of items satisfying a given term or a conjunction of terms, and iii) how to maintain the storage structure when modifications of the dataset and/or the expert vocabulary occur.

The first strategy, denoted *KV* here, is based on a classic key/value storage strategy associating lists of item identifiers with each vocabulary terms. Considering that a row identifier is defined on 4 bytes, allowing for the identification of more than 4 billions of items, and that each term of the vocabulary is associated with an identifier defined on 1 byte, then the space needed to store the rewriting vectors of a dataset $\mathcal{D}$ is equal to $|\mathcal{D}| \times 4 \times (2 \times |\mathcal{V}|) + |\mathcal{V}|$ bytes. Obtaining the list of row identifiers that satisfy a given term is done in constant and very short time using the key-range function of HBASE. To handle modifications of the dataset or the vocabulary, the concerned row identifiers simply have to be moved from one list to another. However, the *KV* storage strategy cannot be used to process efficiently, i.e. in a distributed way, the list of row identifiers associated with each term as this list as to be processed sequentially. Moreover, when one wants to obtain the list of row identifiers satisfying a conjunction of terms, then the intersection between several sets, one for each conjunct, has to be processed locally. For these last two reasons, this strategy that consists of a classical column representation of an index in NoSQL databases is not relevant in our case.

The second strategy, denoted by *BM*, consists in adding a new column in the HBASE table that contains the data. This new column contains the bitmap rewriting vectors of each item. The size of a bitmap rewriting vector is equal to $\frac{|\mathcal{V}|}{8}$ bytes, thus leading to an overhead in terms of space of $|\mathcal{D}| \times \frac{|\mathcal{V}|}{8}$ bytes. For a dataset of 7GB, the storage of the bitmaps in HBASE occupies only 300MB, leading to an overhead of around 4%. The expert vocabularies being generally of a reasonable size, the *BM* strategy requires less additional space than the *KV* strategy. Modifications of the data may be efficiently taken into account by revising the bitmap vectors of the concerned items only. However, a modification of the vocabulary leads to a recalculation of all the bitmap vectors. The construction and storage of such rewriting vectors for 7 million items takes in average 120 seconds with a distributed technique on HBASE. The identification of the subset of items satisfying a given term or a conjunction of terms relies on a sequential scan of the

Table 5
Comparison of the storage strategies *BM* and *KV-BM* in terms of construction time (in second), space occupation (in MB) and retrieval time of a subset of items (in second) for a dataset of 7 million items.

Strategy	Construction time	Space occupied	Retrieval time
BM	120	300	60
KV-BM	600	37	20

tables distributed in the different nodes of the cluster. Considering different combinations of conjunctions of items, we have observed that it takes around 60 seconds in average to retrieve all the items that satisfy a condition, using the functionalities offered by HBASE to scan the data in a distributed way.

Finally, the hybrid storage strategy, denoted by *KV-BM*, consists in maintaining an additional key/value table, keys being the bitmap translations of the distinct rewriting vectors and values being a couple of two row identifiers. Let N be the number of distinct rewriting vectors, then $N \times \frac{|\mathcal{V}|}{8} \times (2 \times 4)$ bytes are used by the *KV-BM* strategy.

As explained in Section 4.3.3, N is generally lower than the number of possible rewriting vectors due to correlations between the different features. Another interesting observation about the evolution of N wrt. the size of the dataset $\mathcal{D}$ is that N converges to a maximum, making N dependent on the applicative context (for the correlation) but not on the size of the dataset. To illustrate this phenomenon, we have computed the ratio between the number of items and the number of distinct rewriting vectors. For a dataset containing 7 million items, we have observed that 700,000 distinct bitmap rewriting vectors are necessary, thus leading to a ratio of 1 common bitmap for 10 items. For the whole dataset of 123 million items, 6,105,145 distinct vectors are obtained corresponding to a ratio of 1 common bitmap for 17 items. Thus, taking into account more items will not lead to a significant increase of the size of the storage structure.

Using the *KV-BM* strategy, the set of items satisfying a term or a conjunction of terms may be obtained very efficiently. A single scan of this table, that may be distributed, is used to determine all the ranges of row identifiers satisfying a given term and their retrieval may be distributed on the different nodes. Despite the conciseness of this strategy and its efficiency for data exploration, the main drawback of this strategy is that modifications of the data or the vocabulary entail a complete reordering of the items, so as to have adjacent identifiers for items that are rewritten in the same way, and a reconstruction of the *KV-BM* table. This strategy is thus mainly dedicated to the analysis of a dataset that does not evolve. For this experiment, the table used by the *KV-BM* strategy has been stored into a classical PostgreSQL table. As shown by the stack trace of the query engine given below, the retrieval of the different ranges row identifiers satisfying a given condition is very efficient (a few milliseconds). Then, retrieving the items stored in HBASE based on the key ranges is performed in a few seconds (20 seconds in average for 7 million items) despite the use of a sequential scan of the table as PostgreSQL is not able to use indexes when a function is used in the selection clause (*get_bit()* in our case).

```
bitmapRewriting_db=# explain analyze select * from bitmaprewritings
                                     where get_bit(bitmap,1) =1 and get_bit(bitmap,2) = 1;
                                                     QUERY PLAN
-------------------------------------------------------------------------------------------------
Seq Scan on bitmaprewritings (cost=0.00..54132.64 rows=49 width=32)
(actual time=223.058..223.058 rows=0 loops=1)
Filter: ((get_bit(bitmap, 1) = 1) AND (get_bit(bitmap, 2) = 1))
  Rows Removed by Filter: 1979032
Planning time: 0.069 ms
Execution time: 223.086 ms
```

Table 5 gives, for each storage strategy, the space occupied considering a dataset of 7 million flights and a vocabulary of 75 terms, as well as the time needed to access the subset of items satisfying a given term (average time observed using 10 different terms from the vocabulary).

These experiments indicate that the *KV-BM* strategy is scalable for big data sets whose content does not evolve. It is worth recalling that the rewriting of the dataset has to be performed once and in back office in order to build the structures necessary to the data exploration functionalities. Cache systems may be used to fasten the selection of a subset of items when a term is selected, but this technical parameter has not been experimented in this work.

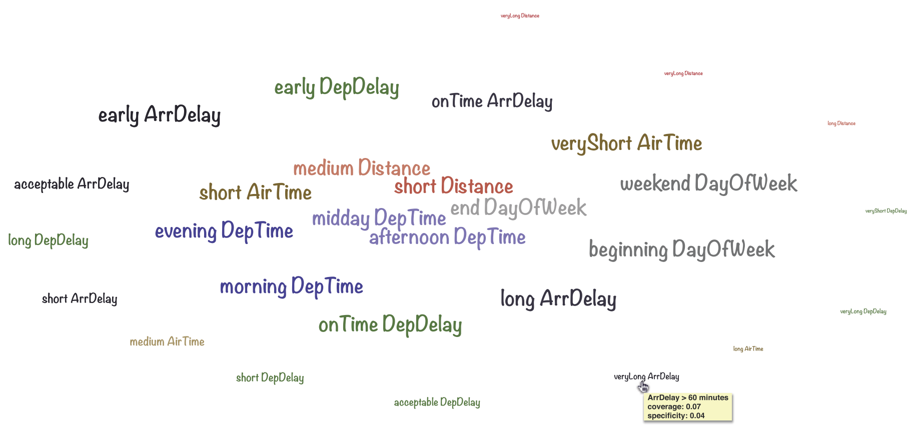


Fig. 9. Linguistic summary of the whole dataset as a term cloud.

6.4. *Interpretability of a term cloud*

Fig. 9 depicts the visualization as a term cloud of the linguistic summary generated for the whole set of 123 million flights. The initial goal of this work (i.e. to efficiently generate a concise summary of a large dataset) is achieved, as indeed, at a glance, one has a complete overview of the whole summary. The choice of a term cloud whose structure follows a spiral clearly contains two eye-catching zones. The first important zone is the center of the cloud that gathers the most specific explanations of the data. The second interesting zone concerns the cloud edge that exhibits the less specific explanations that may describe unexpected and surprising data properties.

By interacting with the cloud, users may also explore the data and determine correlations between linguistic properties about the data. For instance, if one wants to determine the main reasons of very long delays at arrival time, one simply has to click on the term 'veryLong ArrDelay' in order to execute a query on the DB storing the items and to retrieve the flights satisfying this property. This answer set, that may be very large (in the whole dataset 5,488,646 flights arrived with a very long delay), is linguistically rewritten and summarized as a term cloud as well. Contrary to existing knowledge visualization techniques that generally put into perspective two dimensions, a term cloud offers a global picture of a dataset on all the dimensions simultaneously. For instance, the term cloud depicted in Fig. 10 tells us a lot about very delayed flights. By looking at the center of the cloud, one straightforwardly remarks that flights arriving with a very long delay took off with a very long delay too and this, whatever the distance concerned. One may indeed conclude that the distance is not the reason for a very long delay at arrival as all the different linguistic characterizations of a distance (medium, short, long, very short) are of the same specificity and thus close to the center of the cloud. However, 'evening DepTime' and 'beginning DayOfWeek' constitute noteworthy properties of very delayed flights. We argue that such conclusions are easier to draw from personalized and interactive linguistic summaries rather than from the visualization of statistical measures proposed, e.g., by the competitors of the 'Airline on-time data challenge[3]' [29].

7. Conclusion

The crucial issue of helping domain experts make the most of their corporate datasets is addressed in this paper with the proposal of a novel kind of summarization approach. Based on an expert vocabulary modelled by fuzzy partitions and linguistic variables, data properties are translated into linguistic terms that are completed by two measures: the coverage, that quantifies the proportion of items concerned by a term, and the representativity. Linguistic terms and their two associated measures of coverage and representativity are graphically rendered to form a term cloud that gives the expert a concise view of the dataset to analyze. Experimentations show the efficiency of this soft-computing-based approach that creates in linear time a synthetic view of a large dataset.

[3] http://stat-computing.org/dataexpo/2009/posters/.

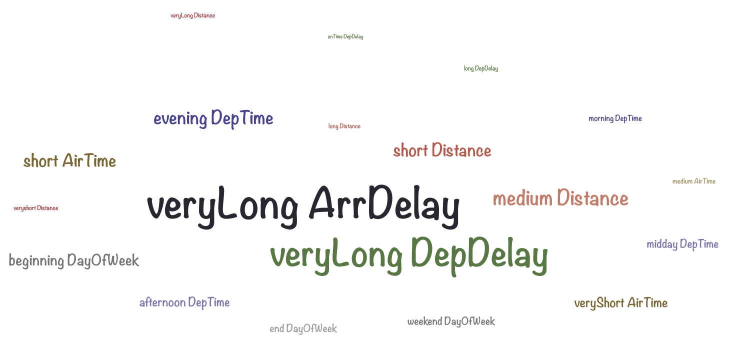


Fig. 10. Linguistic summary of the very delayed flights.

In this work we addressed the problem of rewriting a dataset using linguistic terms taken from the user's vocabulary. This rewriting step is at the heart of many existing soft-computing-based approaches to data management. To the best of our knowledge, it is the first time that this basic problem is addressed from an algorithmic point of view and that technical questions, as the storage and the indexation of the item rewriting vectors, are studied. The linguistic summarization of data has been largely addressed by the soft computing community. We however initiate in this work a novel direction toward the graphical representation of data summaries composed of linguistic terms. The ergonomics of the term clouds we provide is obviously highly perfectible and other measures may be defined to discriminate between the different terms involved in the dataset rewriting vector. These aspects constitute interesting perspectives for future work.

Moreover, the approach is currently used in a commercial prototype as an ABI tool for managing customer relationship. Based on feedbacks about the use of the approach in real-world use cases, we will improve or reconsider the way the dataset rewriting vector is rendered and we will also define other data exploration functionalities. To ease and speed up the analysis of a dataset, it may e.g. be interesting to directly summarize a dataset with conjunctions of terms, instead of atomic ones. But extracting more expressive knowledge comes with a computation cost overhead that has to be controlled so as to maintain the approach scalable.

References

[1] K. Collier, Agile Analytics: A Value-Driven Approach to Business Intelligence and Data Warehousing, Addison–Wesley, 2011.
[2] H. Chen, R.H. Chiang, V.C. Storey, Business intelligence and analytics: from big data to big impact, MIS Q. 36 (4) (2012) 1165–1188.
[3] U.M. Fayyad, A. Wierse, G.G. Grinstein, Information Visualization in Data Mining and Knowledge Discovery, Morgan Kaufmann, 2002.
[4] P. Bailis, E. Gan, S. Madden, D. Narayanan, K. Rong, S. Suri, Macrobase: prioritizing attention in fast data, in: Proceedings of the 2017 ACM International Conference on Management of Data, ACM, 2017, pp. 541–556.
[5] S. Nassar, J. Sander, C. Cheng, Incremental and effective data summarization for dynamic hierarchical clustering, in: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, ACM, 2004, pp. 467–478.
[6] H. Kobayashi, H. Suzuki, K. Misue, A visualization technique to support searching and comparing features of multivariate datasets, in: 2015 19th International Conference on Information Visualisation (IV), IEEE, 2015, pp. 310–315.
[7] C. Seifert, J. Jurgovsky, M. Granitzer, Facetscape: a visualization for exploring the search space, in: 2014 18th International Conference on Information Visualisation (IV), IEEE, 2014, pp. 94–101.
[8] I. Jolliffe, Principal Component Analysis, Wiley Online Library, 2002.
[9] G. Smits, O. Pivert, T. Girault, Reqflex: fuzzy queries for everyone, Proc. VLDB Endow. 6 (12) (2013) 1206–1209.
[10] F.E. Boran, D. Akay, R.R. Yager, An overview of methods for linguistic summarization with fuzzy sets, Expert Syst. Appl. 61 (2016) 356–377.
[11] J. Kacprzyk, S. Zadrożny, Linguistic database summaries and their protoforms: towards natural language based knowledge discovery tools, Inf. Sci. 173 (4) (2005) 281–304.
[12] R. Saint-Paul, G. Raschia, N. Mouaddib, Database summarization: the saintetiq system, in: 2007 IEEE 23rd International Conference on Data Engineering, IEEE, 2007, pp. 1475–1476.
[13] L. Ughetto, W.A. Voglozin, N. Mouaddib, Database querying with personalized vocabulary using data summaries, Fuzzy Sets Syst. 159 (15) (2008) 2030–2046.
[14] M.J. Gacto, R. Alcalá, F. Herrera, Interpretability of linguistic fuzzy rule-based systems: an overview of interpretability measures, Inf. Sci. 181 (20) (2011) 4340–4360.
[15] Z. Wang, G. Klir, Fuzzy Measure Theory, Springer Science & Business Media, 2013.
[16] R.R. Yager, Measures of specificity, in: Computational Intelligence: Soft Computing and Fuzzy-Neuro Integration with Applications, Springer, 1998, pp. 94–113.

[17] N. Marín, G. Rivas-Gervilla, D. Sánchez, Using specificity to measure referential success in referring expressions with fuzzy properties, in: 2016 IEEE International Conference on Fuzzy Systems (FUZZ–IEEE), IEEE, 2016, pp. 563–570.

[18] R. Agrawal, T. Imieliński, A. Swami, Mining Association Rules Between Sets of Items in Large Databases, SIGMOD Rec., vol. 22, ACM, 1993, pp. 207–216.

[19] M. Delgado, N. Marín, D. Sánchez, M.-A. Vila, Fuzzy association rules: general model and applications, IEEE Trans. Fuzzy Syst. 11 (2) (2003) 214–225.

[20] M. Hearst, D. Rosner, et al., Tag clouds: data analysis tool or social signaller?, in: Hawaii International Conference on System Sciences, Proceedings of the 41st Annual, IEEE, 2008, pp. 160–170.

[21] R.R. Yager, M.Z. Reformat, Using fuzzy sets to model information provided by social tagging, in: 2010 IEEE International Conference on Fuzzy Systems (FUZZ), IEEE, 2010, pp. 1–8.

[22] S. Guillaume, B. Charnomordic, P. Loisel, Fuzzy partitions: a way to integrate expert knowledge into distance calculations, Inf. Sci. 245 (2013) 76–95.

[23] S. Guillaume, B. Charnomordic, Generating an interpretable family of fuzzy partitions from data, IEEE Trans. Fuzzy Syst. 12 (3) (2004) 324–335.

[24] G. Smits, O. Pivert, M.-J. Lesot, Vocabulary elicitation for informative descriptions of classes, in: Fuzzy Systems Association and 9th International Conference on Soft Computing and Intelligent Systems (IFSA–SCIS), 2017 Joint 17th World Congress of International, IEEE, 2017, pp. 1–8.

[25] S. Guillaume, B. Charnomordic, P. Loisel, Fuzzy partitions: a way to integrate expert knowledge into distance calculations, Inf. Sci. 245, 76–95.

[26] J. Dean, S. Ghemawat, Mapreduce: simplified data processing on large clusters, Commun. ACM 51 (1) (2008) 107–113.

[27] A. Bhattacharya, Fundamentals of Database Indexing and Searching, CRC Press, 2014.

[28] O. Pivert, G. Smits, How to efficiently diagnose and repair fuzzy database queries that fail, in: Fifty Years of Fuzzy Logic and its Applications, Springer, 2015, pp. 499–517.

[29] R. Wicklin, R. Allison, Congestion in the sky: visualizing domestic airline traffic with SAS software, ASA Statistical Computing and Graphics Data Expo, 2009, 14.