CARI Conference

Scaling the ConceptCloud Browser to Large Semi-Structured Data Sets

Joshua Berndt, Bernd Fischer, Arina Britz

CSIR Center for AI Research
Stellenbosch University
South Africa

RÉSUMÉ. Les ensembles de données semi-structurés, tels que les révisions de produits ou les données de journaux d'événements, deviennent simultanément plus largement utilisés et en même temps de plus en plus volumineux. Cet article décrit ConceptCloud, un navigateur interactif flexible pour les ensembles de données semi-structurés, mettant l'accent sur les modifications architecturales à une architecture basée sur serveur apportées pour accommoder des ensembles de données en constante croissance. ConceptCloud utilise une combinaison d'une visualisation intuitive du nuage de tags avec un treillis des concepts sous-jacent pour fournir une structure formelle pour la navigation dans un ensemble de données sans connaissance préalable de la structure des données ou compromettre l'évolutivité.

ABSTRACT. Semi-structured data sets such as product reviews or event log data are simultaneously becoming more widely used and growing ever larger. This paper describes ConceptCloud, a flexible interactive browser for semi-structured datasets, with a focus on the recent trend of implementing server-based architectures to accommodate ever growing datasets. ConceptCloud makes use of an intuitive tag cloud visualization viewer in combination with an underlying concept lattice to provide a formal structure for navigation through datasets without prior knowledge of the structure of the data or compromising scalability. This is achieved by implementing architectural changes to increase the system's resource efficiency

MOTS-CLÉS : architecture client-serveur, données semi-structurés, nuage de tags, treillis de concepts

KEYWORDS : client-server architecture, semi-structured data, tag cloud, concept lattice

1. Introduction

ConceptCloud [5] is a visualisation and exploration tool for semi-structured data sets, such as software revision control meta-data or product reviews. It uses a concept lattice [2] built from the data set as underlying navigation structure but presents the data itself in form of a tag cloud that concisely summarizes the users current selection in one view and allows further navigation through tag selection and deselection, without constricting the user to pre-defined search paths.

ConceptCloud began as an interactive browser based tool for Git and SVN repositories [1, 4] and has been extended to accept further semi-structured data sets in XML and JSON files. However, its application to large data sets (e.g., the ACM Digital Library, see [6]) have shown the limitations of its original client-based architecture. We have thus re-designed and re-implemented the system to use a new server-based architecture, which also necessitated some user interface changes. In this paper we describe the new architecture and interface and show that it yields 10x performance improvements. Specifically, we present the formal background for the ConceptCloud System, limitations of the original implementation, changes made, and preliminary experimental results over a wine review data set.

2. Formal Concept Analysis

Formal Concept Analysis (FCA) is a theory of data analysis that uses lattice-theoretic methods to investigate abstract relations between objects and their attributes. In FCA, information is represented as a binary cross table, or *context*, where the rows denote objects, eg. products and the columns attributes eg. price or ratings. ConceptCloud uses the concept lattice derived from semi-structured data as its navigation structure. An incidence relation $\mathcal{I}$ indicates which objects in the table have which attributes.

Definition 1 *A formal context is a triple ($\mathcal{O}, \mathcal{A}, \mathcal{I}$) where $\mathcal{O}$ and $\mathcal{A}$ are sets of objects and attributes, respectively, and $\mathcal{I} \subseteq \mathcal{O} \times \mathcal{A}$ is an arbitrary incidence relation.*

Definition 2 *Let ($\mathcal{O}, \mathcal{A}, \mathcal{I}$) be a context, $O \subseteq \mathcal{O}$, and $A \subseteq \mathcal{A}$. The common attributes of O are defined by $a(O) = \{a \in \mathcal{A} | \forall o \in O : (o, a) \in \mathcal{I}\}$, the common objects of A are denoted by $\omega(A) = \{o \in \mathcal{O} | \forall a \in A : (o, a) \in \mathcal{I}\}$.*

Formal concepts are pairs of objects and attributes $\langle O, A \rangle$, where $O \subseteq \mathcal{O}$ and $A \subseteq \mathcal{A}$ such that O is the set of all objects that have all attributes from A and A is the set of attributes that are common to all objects in O.

Definition 3 *Let $\mathcal{C}$ be a context. c = $\langle O, A \rangle$ is called a concept of $\mathcal{C}$ iff $\alpha(O) = A$ and $\omega(A) = O$. $\pi_O(c) := O$ and $\pi_A(c) := A$ are called extent and intent of c, respectively. The set of all concepts of $\mathcal{C}$ is denoted by $B(\mathcal{C})$.*

Concepts are partially ordered by inclusion of extents such that a concepts extent includes the extent of all of its subconcepts ; the intent-part follows by duality.

Definition 4 *Let $\mathcal{C}$ be a context, $c_1 = \langle O_1, A_1 \rangle$, $c_2 = \langle O_2, A_2 \rangle \in B(C)$. c_1 and c_2 are ordered by the subconcept relation, $c_1 \leq c_2$, iff $O_1 \subseteq O_2$ or equivalently, $A_2 \subseteq A_2$. The structure of $B(C)$ and $\leq$ is denoted by $\mathcal{B}(\mathcal{C})$.*

The basic theorem of FCA states that the structure induced by the concepts of a formal context and their ordering is always a complete lattice [2]. Such concept lattices have strong mathematical properties and reveal structural and hierarchical properties of the original data. They can be computed automatically from any given relation between objects and attributes. The greatest lower bound or meet and least upper bound or join can also be expressed by the common attributes and objects.

Theorem 1 *Let $\mathcal{C}$ be a context, then $\mathcal{B}(\mathcal{C})$ is a complete lattice, called the concept lattice of $\mathcal{C}$. Its meet and join operation for any set $\{\langle A_i, B_i\rangle | i \in I\} \subset B(\mathcal{C})$ of concepts are given by :*
$\bigwedge_{i\in I}(O_i, A_i) = (\bigcap_{i\in I} O_i, \alpha\,(\omega(\bigcup i \in I A_i)))$
$\bigvee_{i\in I}(O_i, A_i) = (\omega(\alpha(\bigcup_{i\in I} O_i)), \bigcap_{i\in I} A_i)$

Each attribute and object has a uniquely determined defining concept in the lattice. The defining concepts can be calculated directly from the attribute or object, respectively, and need not be searched in the lattice.

Definition 5 *Let $\mathcal{B}(\mathcal{O}, \mathcal{A}, \mathcal{I})$ be a concept lattice. The* defining concept *of an attribute $a \in \mathcal{A}$ is the greatest concept c such that $a \in \pi_A(c)$ holds. It is denoted by $\mu(a)$. The defining concept of an object $o \in O$ is the smallest concept c such that $o \in \pi_O(c)$ holds. It is denoted by $\sigma(o)$.*

Efficient algorithms exist for the computation of the concept lattices and the meet and join of concepts in the lattice [3]. For a detailed introduction to FCA see [2].

3. ConceptCloud

ConceptCloud [5] is a browser for semi-structured datasets which allows the user to navigate, via tag clouds, through a dataset in what is known as an *explorative search*. This type of exploration requires no predefined knowledge of the domain or dataset. The user iteratively selects an attribute or object tag in a tag cloud, and the ConceptCloud system adjusts the tag cloud to display all other tags attached to objects possessing the selected attribute tag(s). This is achieved by maintaining a focus concept from which a tag cloud is created.

Formally, the *focus concept* $c = \langle O, A\rangle$ is the concept whose extent is the set of objects that share the set of currently selected attributes, F, within the tag cloud, such that $\alpha(\omega(F)) = \pi_A(c) = A$. the new focus concept. concept and A new is the attributes of the new focus concept.

The focus concept can be further refined by iteratively adding elements to F. When an additional attribute is added to F, we update the focus concept by computing the meet, as per Theorem 1, of the current focus concept c and the concept introduced by the additional attribute. In Section 4 we will discuss how this was changed.

The explorative search process corresponds to the process of stepping through a concept lattice, wherein the selection of an attribute moves us to the point in the lattice where all linked objects contain that attribute. As we select further attributes we move further down the lattice. If we deselect an attribute we move back up the lattice and have access to a different set of attributes and objects. This corresponds to the refinement of the focus concept by adding and removing elements from F. This approach was tested in a user study conducted in [6] and found that users were able to answer complex scientometric questions using ConceptCloud with a mean correctness of 73%, with the users' prior research experience having no statistically significant effect on results. For further detail see [1]. ConceptCloud presents the data in the form of a tag cloud, where the frequency of each tag denotes its importance. Each tag cloud is a word cloud-like window, wherein all of the objects and attributes in the lattice are represented as tags, words whose size denote their importance within this window. More specifically in ConceptCloud, each tag in a tag cloud's size is based on the frequency of its occurrence within the sub selection of the dataset, coloured differently based on its category (namely the type value of the attribute or object). Tag clouds are constructed, to help distinguish the different properties of the data set, by taking the extent of the focus concept $c = \langle O, A \rangle$, then for each $o_i \in O$, we determine its defining concept c_i, see Definition 5. We then collect all the intents of these defining concepts. These are the attributes we display in the tag cloud. Finally we add the objects to the tag cloud so that they may be directly selected or searched within the tag cloud. Our initial focus concept will have no selected attributes, and thus the tag cloud created from it will contain tags representing all attributes and objects. Formally we have (here $\uplus$ denotes multiset union) :

Definition 6 *The tag cloud from a concept c = (O, A) ∈ B($\mathcal{C}$) is defined as* $\tau\ (c) = O \uplus \biguplus_{o \in O} \pi_A(\sigma(o))$.

By constructing the objects in the tag cloud, we induce subconcepts of the focus concept, from which the tag cloud was derived, and all concepts having a non-bottom meet with that focus concept.

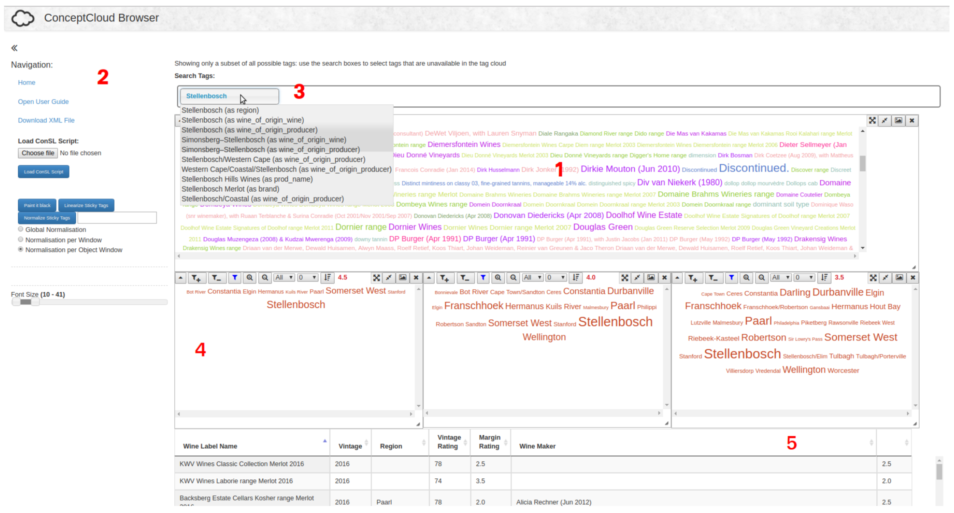
The initial implementation of the ConceptCloud system was geared towards exploration of the metadata of software repositories [1, 5]. As such it was not built with scaling in mind since the metadata within a software repository forms a comparatively small semi-structured dataset. When the use of the application shifted from analysis of these repositories to analysis of other semi-structured datasets[6], it became apparent that some of the design choices initially made were no longer feasible. One such choice was to have each tag cloud display tags representing all attributes and objects without any limit. The lack of a display limit also exists in the displayed representation of the context table, which too will display all attributes and objects. In practice this is increasingly resource intensive for larger datasets.

4. Scalable ConceptCloud Architecture

In order to reduce the resource intensive nature of ConceptCloud, changes to the initial implementation had to be made. A fixed sized subset of all tags was selected to represent the underlying concept lattice. This in turn necessitated a way for the user to interact with tags that may not be displayed in the initial window. The logical choice was to incorporate autocomplete based search functionality, as mentioned in Section 4.2. For this to function correctly a caching structure and separation of the data in memory was required. For this we implemented a postgresql database, which table's are generated based off the structure of the input dataset. The number of tags in a tag cloud was limited to the top 5000 tags, by frequency in the extent of the focus concept for that tag cloud. This limit was imposed to maintain a functional interface and not overwhelm the user. Additionally the attributes to be displayed in the context table representation was configured and limited. The context table representation, known as the *table view* limits the displayed results but allows the user to page through the list of all results. Finally the system creates its concepts on the fly, meaning that the overhead of creating the full lattice is avoided, allowing for generally responsive tag cloud creation and rendering.

The architectural changes presented are a generalization of the architecture used in [6], a highly specialized version of ConceptCloud used to visualize the DBLP, Computer Science Bibliography. This dataset, at the time of writing, has over 4 million records. The scalable ConceptCloud Architecture presented is not specialized to any specific dataset and may be used for any well formed JSON dataset. It will correctly generate the required caching databases and create the required tag clouds. Additionally the user interface was updated to better function with the new architecture.

4.1. ConceptCloud User Interface



ConceptCloud Browser
Navigation:
2
Home
Open User Guide
Download XML File
Load ConSL Script:
Choose file
No file chosen
Load ConSL Script
Paint it black
Linearize Sticky Tags
Normalize Sticky Tags
Global Normalisation
Normalisation per Window
Normalisation per Object Window
Font Size (10 - 41)
Showing only a subset of all possible tags: use the search boxes to select tags that are unavailable in the tag cloud
Search Tags:
Stellenbosch
3
Stellenbosch (as region)
Stellenbosch (as wine_of_origin_wine)
Stellenbosch (as wine_of_origin_producer)
Simonsberg–Stellenbosch (as wine_of_origin_wine)
Simonsberg–Stellenbosch (as wine_of_origin_producer)
Stellenbosch/Western Cape (as wine_of_origin_producer)
Western Cape/Coastal/Stellenbosch (as wine_of_origin_producer)
Stellenbosch Hills Wines (as prod_name)
Stellenbosch Merlot (as brand)
Stellenbosch/Coastal (as wine_of_origin_producer)
1
4
4.5
Bot River Constantia Elgin Hermanus Kuils River Paarl Somerset West Stanford Stellenbosch
Robertson Sandton Somerset West Stanford Stellenbosch Wellington
Riebeek-Kasteel Robertson Sir Lowry's Pass Somerset West Stanford Stellenbosch Stellenbosch/Elim Tulbagh Tulbagh/Porterville Villiersdorp Vredendal Wellington Worcester
5

Wine Label Name	Vintage	Region	Vintage Rating	Margin Rating	Wine Maker	
KWV Wines Classic Collection Merlot 2016	2016		78	2.5		2.5
KWV Wines Laborie range Merlot 2016	2016		74	3.5		2.0
Backsberg Estate Cellars Kosher range Merlot	2016	Paarl	78	2.0	Alicia Rechner (Jun 2012)	2.5

Figure 1. *Navigation User Interface*

The navigation user interface consists of the following components :

– The *main window* (1) wherein the tag clouds are displayed. On initial execution this displays the initial tag cloud viewer with the default focus. The tag cloud within the main window displays the top 5000 most relevant tags. A user selecting a tag in this window causes the focus concept to be recalculated and the viewer to then display the point in the lattice wherein the updated focus concept is relevant.

– The *Navigation Menu* (2) provides the user with various utilities relating to saving the lattice as well as uploading a ConSL script [4] to automate the display of the viewers. The further options allow the user to change the scaling of the tags within the tag cloud viewers. 4.1.

– *Search Functionality* (3) was introduced as only the top 5000 tags are displayed in the main window, the user may wish to interact with tags that are not currently displayed. As the user inputs their search terms, ConceptCloud displays an auto-completed list of terms and their category. This is done by making use of the caching database. Selecting one of these terms updates the focus concept, and as a result, the main window together with any other tag cloud viewer that contains the selected term as its focus concept. This action is identical to if they were to select a displayed tag in the main window.

– *Sticky Tag Cloud Viewers* (4) are sub-windows of the main window that contain each displayed tag cloud, and once created always appear below the main window, they can however be moved from their initial position. Each Sticky Tag Cloud Viewer contains the displayed tags for the sticky concept for that viewer, referred to as the sticky tag. A sticky tag is an object or attribute to which the viewer is fixed. Selecting a new focus concept will adjust these windows to use the union of the sticky and selected focus concept as their focus concept. The sticky tag is displayed next to the Tag Cloud viewer's menus in red. The menus exist to adjust the display of the contained tags. These viewers allow the user to have multiple differentiated views, eg. viewing the ratings of wines across multiple vintages, with a window for each vintage or rating.

– The *Table View* (5) displays the underlying context table for the concept lattice connected to the initial tag cloud viewer. Selecting and deselecting tags will cause this to update to reflect the concept table corresponding to the concept lattice of the focus concept(s). In many datasets there is a multitude of attributes for each object in the context table, often too many to display concisely. The attributes to be displayed in the Table View can be configured to solve this. The results appear in a fixed page size list, further easing resource usage.

4.2. Navigation Architecture

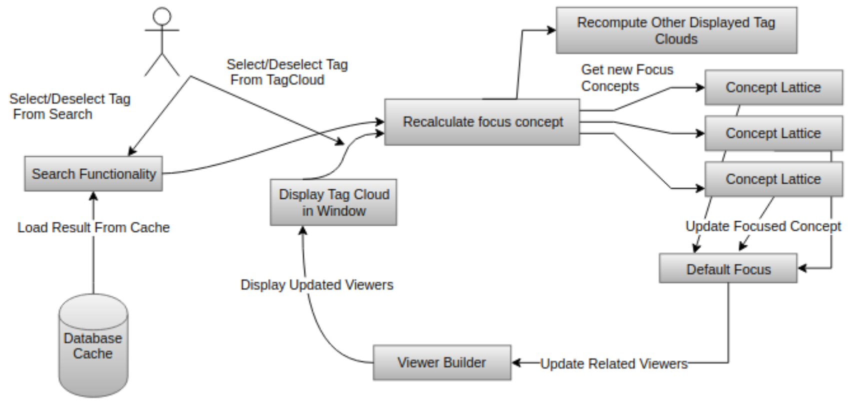


Figure 2. *Navigation Architecture*

An explorative search in ConceptCloud is the process whereby the user selects and deselects tags in a tag cloud allowing them to step through the underlying concept lattice. The user is additionally able to create sub windows, which are additional tag cloud viewers with stickied tags. These sub windows have one attribute set for it and will display the related subsection of the concept lattice. Selecting a new focus concept from a tag in any of the viewer windows, including the initial tag cloud, causes the selected tag union with stickied tag for each window (as the tag represents either an attribute or an object), to become the focus concept of each viewer. The related portion of the lattice is displayed if stickied tag and new focus concept are not disjoint. Otherwise an empty viewer window is displayed. The architecture of the navigation subsystem is outlined in figure 2. Limiting the displayed tags necessitated that we have two ways to interact with the tags. One based directly on the selection of a displayed tag in a tag cloud, and another based on searching for a tag that may or may not be displayed within a tag cloud. In this way we are able to maintain the original ConceptCloud functionality. The architecture of the navigation subsystem is shown in Figure 2 :

– Select / Deselect Tags from the tag cloud : The user clicks a tag within one of the displayed tag clouds, this then causes all tag cloud windows to take this new selected tag and use it to recalculate their focus concept, causing each tag cloud window/viewer to update their controllers to request the relevant section of the underlying concept lattice should it exist. A new tag cloud is then constructed by the viewer builder and displayed, in the case of multiple concepts, if no intersection in the lattice between these concepts exists, an empty tag cloud is displayed in the corresponding viewer. Deselection of a tag is when the user clicks the highlighted red tag in the tag cloud, this removes it from all the tag cloud window's focus concept and updates the lattice for each window accordingly.These actions correspond directly to the explorative search mentioned in Section 3.

– Select / Deselect Tag From Search : An autocomplete based search which starts autocompletion after three characters, providing the user with the tag name and the category in which they wish to search. As the user enters text into the search bar, the search functionality performs a lookup in the database cache, and provides a list of closest tag name matches, and their categories to the user in the form of a dropdown list. The user may then click a tag from the list. From this point onwards the system acts as if a tag had been selected from a tag cloud as before. All tag cloud windows add the selected tag to their focus concept and all underlying lattices are updated. The viewer builder constructs new viewers with the updated lattice sections, causing all tag clouds to be updated with the relevant data. Deselection works identically as above.

4.3. Experiments

To show the difference in the architectures, we ran a series of experiments with a typical application driven semi-structured dataset. A series of typical user actions were taken, automated and then timed to display the differences in execution times for the different architecture.

The dataset used, contained 16306 wine reviews, where each review has the following attributes : name, varietal, vintage, review year, review, reviewer, points, price, country, location, region, winery, review phrases. Where the final field, review phrases, is a key-phrase extraction of the review field. This dataset was chosen as it succinctly displays the difference in performance between the two architectures.

For the experiments the following actions serve as our experiments ; initial rendering and the creation of new windows for high, medium and low volume tags. All times given are in milliseconds. These are all run on a machine with the following processing specifications, a 6th Generation Intel Core i7-6700HQ (3.5GHz) and 8Gb DDR4 2133Mhz.

For each architecture, the server and client response times are measured. The results, averaged across 20 runs were as follows :

User Action	Old Architecture (ms)	New Architecture (ms)
Initial Rendering	4863	378
Category Change	182	42
New High Volume Tag Render	7822	488
New Medium Volume Tag Render	4904	374
New Low Volume Tag Render	4218	314

The user actions carried out involved the following ; changing the category filter to va-

rietal, creating a new tag cloud with the United States as the high volume tag (count of 5335), creating a new tag cloud with 2005 vintage as the medium volume tag (count of 1782) and creating a new tag cloud with 2001 as a vintage for the low volume tag (count of 190).
We note that for each action the execution time is in each case at least an order of magnitude faster for the server-based architecture when compared to the same operation on the old architecture, on an identical dataset. The large speedup is due to the much lower resource cost of the new architecture, as we are no longer rendering the entire dataset, but only the top 5000 tags and a far smaller table view. Even when rendering less than 5000 tags, the fact that the initial cloud and table view are so resource intensive in the old client-based architecture means creating any additional tag clouds will suffer. The new server based architecture does not have have this issue.

5. Future Work

In this paper we described ConceptCloud, an interactive browser for semi-structured datasets and the changes made to it to enable it to more easily deal with large datasets. We showed that changes made resulted in a large speedup that made using large datasets feasible. For future work there are plans to move the ConceptCloud application from a client server application using a web client, to a client server with a mobile client. Currently we are working on adding support for processing ontological datasets, and using ontologies to enrich the data within the dataset. Finally we are adding support for geolocation data, and specialized interactions such as opening a tag on a map based on it's geolocation.

6. Bibliographie

[1] G J. GREENE AND B. FISCHER, « Interactive Tag Cloud Visualization of Software Version Control repositories », *Software Visualization (VISSOFT), 2015 IEEE 3rd Working Conference*, 2015.

[2] B. GANTER AND R. WILLE, « Formal Concept Analysis - Mathematical Foundations », *Springer, 1999.*

[3] ZAKI, MOHAMMED AND HSIAO, CHING-JUI AND OTHERS « CHARM : An Efficient Algorithm for Closed Association Rule Mining », *1999.*

[4] G J. GREENE, M. ESTERHUIZEN, B. FISCHER « Visualizing and Exploring Software Version Control Repositories using Interactive Tag Clouds over Formal Concept Lattices », *Information & Software Technology volume volume 87, 223–241 Elsevier, 2017.*

[5] G J. GREENE, B FISCHER « Conceptcloud : A Tag Cloud Browser for Software Archives. », *ACM SIGSOFT International Symposium on Foundations of Software Engineering, 22, 759–762 ACM, 2014.*

[6] M. DUNAISKI, G J. GREENE, B. FISCHER « Exploratory Search of Academic Publication and Citation Data using Interactive Tag Cloud Visualizations », *Scientometrics, Volume 110(3), 1539–1571 Elsevier, 2017.*