Scaling the ConceptCloud Browser to Very Large Semi-Structured Data Sets: Architecture and Data Completion

by

Joshua Berndt

Thesis presented in partial fulfilment of the requirements for the degree of Master of Science (Computer Science) in the Faculty of Science at Stellenbosch University

Supervisor:	Prof. B. Fischer
Co-supervisor:	Prof. K. Britz

December 2020

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: December 2020

Abstract

Scaling the ConceptCloud Browser to Very Large Semi-Structured Data Sets: Architecture and Data Completion

J. Berndt

Computer Science Division,
Department of Mathematical Sciences,
University of Stellenbosch,
Private Bag X1, 7602 Matieland, South Africa.

Thesis: MSc (Computer Science)

December 2020

Semi-structured data sets such as product reviews or event log data are simultaneously becoming more widely used and ever larger. This thesis describes ConceptCloud, a flexible, interactive browser for semi-structured datasets, with a focus on the improvements made to accommodate larger datasets, more intuitive data representation and the enrichment of the underlying data by way of data-imputation. ConceptCloud makes use of an intuitive tag cloud visualisation 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 scalability is achieved by the implementation of architectural changes to increase the system's resource efficiency. These changes are demonstrated by way of a case study on a dataset of wine reviews.

Semi-structured data sets such as product reviews or event log data often contain a geolocation aspect: for example, the location of the winery for wine reviews, or the accident location for traffic data. In this thesis, I describe ConceptCloud extensions which allow for the rendering of specialised geolocation data while providing alternate navigation paths through the dataset. I show that using biclusters can make the navigation bidirectional, and demonstrate this approach on a crime data set making use of a geolocation specialised map viewer.

Semi-structured data often contains implicit information which will be useful in driving data exploration if made explicit. I take advantage of domain ontologies to both allow implicit data in each input data set to be made explicit and verify and correct inconsistencies allowing for better data exploration. I demonstrate this approach with a continuation of the wine case study.

Uittreksel

J. Berndt
Rekenaar Wetenskap Afdeling,
Departement Wiskundige Wetenskappe,
Universiteit van Stellenbosch,
Privaatsak X1, 7602 Matieland, Suid Afrika.

Tesis: MSc (Rekenaarwetenskap)
Desember 2020

Semi-gestruktureerde datastelle soos produkbeoordelings of gebeurtenislog-data word terselfdertyd al hoe meer gebruik en word al hoe groter. Hierdie tesis beskryf ConceptCloud, 'n buigsame, interaktiewe blaaier vir semi-gestruktureerde datastelle, met die fokus op die verbeterings wat aangebring is om groter datastelle te akkommodeer, meer intuïtiewe datavoorstelling te bereik en die verryking van die onderliggende data deur gebruik van data-berekening. ConceptCloud maak gebruik van 'n intuïtiewe tag-wolkvisualisering-kyker in kombinasie met 'n onderliggende konseprooster om 'n formele struktuurte bou vir navigasie deur datastelle sonder voorafkennis van die struktuur van die data of om die skaalbaarheid in die gedrang te bring. Hierdie skaalbaarheid is bereik deur die implementering van argitektoniese veranderings om die stelsel se hulpbrondoeltreffendheid te verhoog. Hierdie verbeterings word by wyse van 'n gevallestudie op 'n datastel van wynoorsigte gedemonstreer.

Semi-gestruktureerde datastelle soos produkbeoordelings of gebeurtenislog-data bevat 'n ligginggewing-aspek: byvoorbeeld die ligging van die wynmakery vir wyn resensies, of die ongelukligging vir verkeersdata.

In hierdie tesis beskryf ons 'n ConceptCloud-uitbreiding wat voorsiening maak vir gespesialiseerde ligginggewing-data, aangesien ons almal navigasie-paaie deur die datastel wissel.

Ons wys dat die gebruik van biclusters die navigasie in twee rigtings kan laat plaasvind en demonstreer hierdie benadering op 'n misdaaddatastel wat gebruik maak van 'n gespesialiseerde geolokasie-kaart kyker.

Semi-gestruktureerde data bevat dikwels implisiete inligting wat nuttig sal wees om data-eksplorasie te dryf as dit kan eksplisiet gemaak word. Ons benut die domeinontologieë om beide implisiete data in elke insetdatastel eksplisiet te laat maak as ook teenstrydighede te verifieer en te korrigeer, wat beter data-eksplorasie moontlik maak. Ons demonstreer hierdie benadering deur 'n gevallestudie met wyn data.

List of Publications

[1] Berndt, J. and Fischer, B. and Britz, K. Scaling the ConceptCloud browser to large semi-structured data sets, *HAL 2018*

[2] du Toit, T. and Berndt, J. and Britz, K. and Fischer, B. ConceptCloud 2.0: Visualisation and Exploration of Geolocation-Rich Semi-Structured Data Sets *ceur-ws.org 2019*

Acknowledgements

Thank you to my supervisors, for all the effort, guidance and motivation which was instrumental for the completion of this thesis.

The financial assistance of the Centre for Artificial Intelligence Research (CAIR) is hereby gratefully acknowledged.

Contents

List of Figures

Chapter 1

Introduction

In the current information age, the availability of increasingly large data sets and semi-structured information has become increasingly prevalent [7]. Semi-structured data[5] refers to data which does not conform to the rigid structuring of relational data, but still contains some form of self-describing metadata. Semi-structured data has moved to the forefront of the web data sphere, where the growth of the web has driven the growth of semi-structured data exponentially with regard to availability and size. This is seen with the increased use of both *XML* and *JSON* data, the latter becoming increasingly widespread with the increase in *JavaScript* based web applications.

Semi-structured data can be modelled in graph-like or treelike structures. These structures actively lend themselves towards *data browsing*, as described by Buneman [5]. This browsing or *exploration* allows users to step through the data without prior knowledge of the structure or schema of the data.

Figure 1.1 gives an example of a semi-structured *JSON* [11] data object. We see that the object contains metadata describing its attributes, in this case, giving their titles alongside their values. Data sets typically consist of many such objects; wherein each attribute may contain further subobjects.When presented with many such objects, for instance, a software repository with many thousands of commits such as a *JUnit* [12] repository, one needs an exploration tool to gain a holistic overview of the data set.

Several data exploration and visualisation tools based on concept lattices, specialising in the analysis of unstructured and semi-structured data, exist, and are explored in Section 2.2.

The main flaw with most of these tools is that they scale poorly, either due to visualisation of the entire concept lattice lattice [4, 35], or the context table as a whole. The performance cost of the visualisation of an entire lattice becomes prohibitively expensive. For a tool to be able to scale to analyse increasingly large semi-structured data sets, it must either be highly optimised towards the particular data set or it must avoid paying the performance cost of the visualisation.

This scaling can be achieved by providing an alternate scalable visualisation

```
{
"name": "Stark-Condé: Cabernet Sauvignon ‘‘Three Pines’’ 2007",
"winery": "Stark-Condé",
"location": "Jonkershoek Valley",
"region": "Stellenbosch",
"country": "South Africa",
"varietal": "Cabernet Saubignon",
"vintage": "2007",
"price": "42",
"points": "91",
"reviewer": "Michael Franz",
"reviewyear": "2010",
"review": "Intense and deeply flavorful, this is a dramatic
wine that grabs one’s attention and doesn’t let go.
Perhaps its most impressive aspect is its aromatic complexity,
as notes of dark berries, cassis, smoke, pine needles and
eucalyptus all show themselves quite notably. Full-bodied,
with plenty of tannin and oak, this is a wine for pairing
with assertive foods like grilled lamb."
},
```

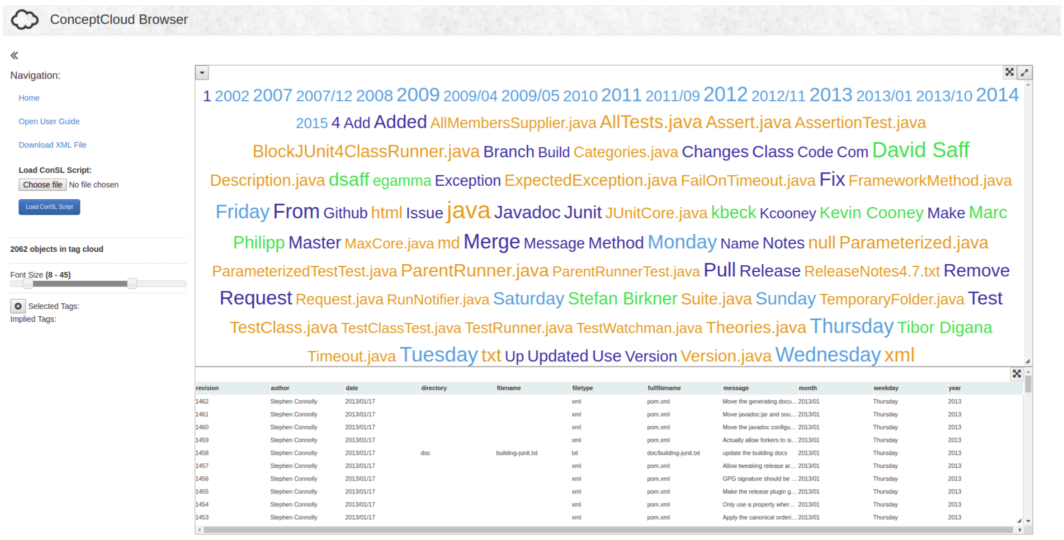
Figure 1.1: Semi-structured JSON object showing wine data.

that can represent the data without having to display it in its entirety.

The tool chosen which represents the alternate visualisation technique is ConceptCloud [17]. ConceptCloud is a visualisation and exploration tool for semi-structured data sets, that makes use of a tag cloud based visualisation to represent an aggregated view of the semi-structured data.

ConceptCloud

ConceptCloud was initially built as an interactive tool for browsing Git and SVN software repositories [16, 18]. These repositories are rich in semi-structured data due to the inherently structured nature of the software version control domain. In summary, items in a software repository are tagged with self-describing metadata by both the author and the version control software itself, providing both highly structured and unstructured metadata. This metadata rich domain proved to be the ideal application domain for ConceptCloud's initial development. ConceptCloud makes use of formal concept analysis [13] to structure semi-structured data hierarchically in a concept lattice. ConceptCloud uses this concept lattice as an underlying navigation structure, and presents the data in the form of an interactive tag cloud, as seen in Figure 1.2,



1 2002 2007 2007/12 2008 2009 2009/04 2009/05 2010 2011 2011/09 2012 2012/11 2013 2013/01 2013/10 2014 2015 4 Add Added AllMembersSupplier.java AllTests.java Assert.java AssertionTest.java BlockJUnit4ClassRunner.java Branch Build Categories.java Changes Class Code Com David Saff Description.java dsaff egamma Exception ExpectedException.java FailOnTimeout.java Fix FrameworkMethod.java Friday From Github html Issue java Javadoc Junit JUnitCore.java kbeck Kcooney Kevin Cooney Make Marc Philipp Master MaxCore.java md Merge Message Method Monday Name Notes null Parameterized.java ParameterizedTestTest.java ParentRunner.java ParentRunnerTest.java Pull Release ReleaseNotes4.7.txt Remove Request Request.java RunNotifier.java Saturday Stefan Birkner Suite.java Sunday TemporaryFolder.java Test TestClass.java TestClassTest.java TestRunner.java TestWatchman.java Theories.java Thursday Tibor Digana Timeout.java Tuesday txt Up Updated Use Version Version.java Wednesday xml

revision	author	date	directory	filename	filetype	fullfilename	message	month	weekday	year
1462	Stephen Connolly	2013/01/17			xml	pom.xml	Move the generating docu…	2013/01	Thursday	2013
1461	Stephen Connolly	2013/01/17			xml	pom.xml	Move javadoc:jar and sou…	2013/01	Thursday	2013
1460	Stephen Connolly	2013/01/17			xml	pom.xml	Move the javadoc configu…	2013/01	Thursday	2013
1459	Stephen Connolly	2013/01/17			xml	pom.xml	Actually allow forkers to si…	2013/01	Thursday	2013
1458	Stephen Connolly	2013/01/17	doc	building-junit.txt	txt	doc/building-junit.txt	update the building docs	2013/01	Thursday	2013
1457	Stephen Connolly	2013/01/17			xml	pom.xml	Allow tweaking release ar…	2013/01	Thursday	2013
1456	Stephen Connolly	2013/01/17			xml	pom.xml	GPG signature should be …	2013/01	Thursday	2013
1455	Stephen Connolly	2013/01/17			xml	pom.xml	Make the release plugin g…	2013/01	Thursday	2013
1454	Stephen Connolly	2013/01/17			xml	pom.xml	Only use a property wher…	2013/01	Thursday	2013
1453	Stephen Connolly	2013/01/17			xml	pom.xml	Apply the canonical orderi…	2013/01	Thursday	2013

Figure 1.2: ConceptCloud tag cloud and context table displaying semistructured data for the *JUnit* GIT repository.

where it is used to visualise repository data for the *JUnit* repository. The interactive tag cloud visualisation allows for selection and deselection of tags which correspond to objects and attributes in the concept lattice, further explained in Sections 2.1 and 2.3, respectively. This selection and deselection of tags corresponds to stepping through the underlying concept lattice by maintaining an updated focus concept, in a process that is referred to as *explorative search*. The explorative search process allows for exploration of a semi-structured data set without constricting the user to predefined search paths or requiring prior knowledge of the underlying structure of the data.

ConceptCloud differs in its visual representation of the data, as it does not attempt to represent the formal concept lattice directly, but rather provides a tag cloud based representation of the items within the context lattice. However, ConceptCloud still allows the user to make use of the formal concept lattice to draw additional insight from the data. ConceptCloud achieves this by representing data as an interactive tag cloud, that allows the users to step through the concept lattice by selection and deselection of tags. ConceptCloud provides a context table linked to the currently selected tag within the Tag Cloud. The context table and tag cloud are dynamically updated as the currently selected tag within the tag cloud is updated, that is the selection of tag(s) changes.

ConceptCloud's application to Large data sets (e.g., the ACM Digital Library, see [10]) has shown limitations in its original client-based architecture, in terms of both scalability, flexibility and extensibility.

1.1 Thesis Goals

The main goal of this thesis is to enrich the explorative search process in the ConceptCloud tool, by (i) improving the tool's scalability and flexibility, (ii) improving the support of visualisation for specific datasets, and (iii) improving ConceptCloud's ability to handle less well-curated semi-structured datasets.

To achieve this these goals, I developed three main additions to ConceptCloud, each contributing towards and required for the end goal of making ConceptCloud a more robust and adaptable software tool, capable of providing the user with a more flexible and intuitive visualisation and data exploration. These improvements yield a more flexible and intuitive visualisation and data exploration than both previous versions of ConceptCloud, and related tools in the same domain.

The first significant addition to ConceptCloud covered is a substantial refactoring and redesign of the underlying architecture. The goals of the architectural updates were to allow for larger semi-structured data sets to be explored, to allow ConceptCloud to become agnostic to the data sets it accepts, and finally to enable the provision of ConceptCloud's back-end explorative search, to a data set as a service to external visualisations or completely separate projects.

The second lays the foundations for the addition of navigational support for specialised geolocation data viewer for ConceptCloud [9]. Substantial optimisations to allow this viewer to represent large data sets and provide additional means of on the fly data aggregation for explorative search. The goal of this is to illustrate the extensibility and robustness of the extended ConceptCloud architecture while showing the feasibility of enriching the explorative search process by adding improved data specific visualisation and data aggregation functionality.

The third and final addition is support for supplementing the input data set with an ontology, allowing for attribute completion within the lattice, leading to improved exploration of the input data. The goal of this is to demonstrate the feasibility of enriching the explorative search process by enriching the underlying data.

1.2 Server-Based Architecture

With the general shift to an Internet Of Things world, the availability of ever-larger data sets has become a norm [7], leading to a growth in size and availability of semi-structured data to be explored by ConceptCloud. As such, ConceptCloud has to adapt to this new demand in scalability to stay relevant.

ConceptCloud was initially adapted to support a much larger data set in [10], wherein specific tailored changes were made to the architecture to support the exploration of the ACM Digital Library. Although effective, the changes made to the architecture where specific to ACM Digital Library data sets and did not translate to other data sets. Thus the goal of the final architecture shift was to adapt the tailored changes made for the Digital Library data set to apply to any semi-structured data set, large or otherwise. This would require the proposed architecture to be configurable to each new data set with minimal effort by the user.

The ConceptCloud tool has grown and been developed over the course of the past seven years. As such, there is a large overhead period for a developer when they wish to extend the system. For non-architectural changes this overhead is especially tedious as the developer only requires the output data from the ConceptCloud tool to build a visualisation, how the data is provided may be of little concern or consequence to their work. I aim to provide an API to developers which allows them access to the unique navigational exploration functionality provided by ConceptCloud as a service, which is based on their input data and provides them with the data corresponding to the underlying concept lattice they are currently focused on. This approach would aid further developers on the project in the creation of further visualisations and extensions based on ConceptCloud's interactive tag based data exploration functionality.
This leads to the first research question: Can a scalable version of ConceptCloud that is generic in the data it accepts be built? We already know we can build a scalable version if it is very specialised but in this thesis that functionality will be migrated to a flexible and configurable implementation whilst still being able to provide this data to external sources.

1.3 ConceptCloud Data Navigation

ConceptCloud provides the user with a default tag cloud visualisation, which, while effective at visualising general data sets, does not take any advantage of the class of data being explored. That is, each data set is provided with a single visualisation. ConceptCloud's default visualisation also only allows for data to be aggregated across the underlying many-valued context table's attribute classes and attribute-values. The default tag cloud visualisation does not provide any way to aggregate data when those values and classes are not present or are incomplete. For example, in Figure 1.2, if I wished to see all of the lattice items which were added to the software repository on a weekend, I would be unable to. While the concepts for days of the week exist in the underlying many-valued context table, there is no explicit concept for weekend or weekday which aggregates these days of the week.

This shortcoming is due to tag cloud based navigation requiring a tag to be present and selectable for it to be used in navigation. In the above example, I would require a weekday and weekend tag, corresponding to the underlying weekday or weekend concepts.

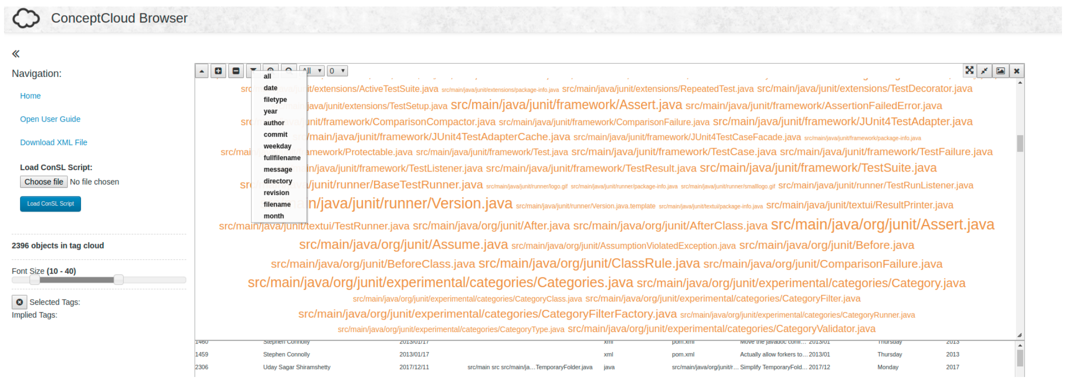


Figure 1.3: ConceptCloud Tag Cloud showing the 'fullfilename' attribute for the *JUnit* repository

Furthermore, consider the ConceptCloud Tag Cloud screen of the *JUnit* software repository, shown in Figure 1.3. If one wanted to select a view of all Java packages used in the project, an attribute which does not explicitly exist, one would have to aggregate across the full filename and directory attributes. While technically possible, this operation is unwieldy and exceedingly slow.

The best approach to resolve this problem would be to provide a specialised viewer, or the ability to implement a specialised viewer, for instance, a file tree to more naturally enable navigation of the data by an attribute that links to a sense of locality.

More generally this problem frequently occurs in both tag cloud and lattice visualisation. That is, the location of items within the lattice and Tag Clouds does not by default correspond to the item's locality attribute in the domain of interest. This makes inferring or exploring along these locality-based bounds increasingly difficult. It is important to note here that the notion of locality does not refer to the position of the tag representation in the tag cloud but rather the position of the concept within the underlying concept lattice itself.

Item or tag positioning in tag cloud visualisation is either random, based on tag size, or alphabetical. Items or concepts within a formal concept lattice, as covered in Section 2.1, are organised into the lattice based on their inclusion of concepts. These concepts don't need to have any locality-based attribute to determine their placement.

This problem can be compounded by an inability to select regional clusters where no such segregating or clustering attribute or concept exists.

It should also be possible to provide a method to aggregate data together, along these locality-based bounds, without changing the underlying data set. Which should be demonstrable with a specialised viewer.

One obvious choice to display this approach is the exploration of location rich data. That is to allow the aggregation of items by their geolocations and use a specialised viewer to show the usefulness of this approach.

An intuitive visualisation to provide for this domain would be an interactive map alongside a tag cloud. This representation of a location is far more useful than a text-based longitude or latitude, or even a place name, as it instantly provides the relative context between two items based on their location.

An interactive specialised viewer of this kind allows for data exploration which can be done from a geographically based context, whilst making use of the underlying data aggregation which I wish to demonstrate.

This leads to the second research question: Is it possible to provide architectural support for specialised visualisations which in turn provide a more intuitive representation of more domain-specific data sets while making use of the flexible aggregation of the underlying data.A map-based visualisation demonstration would allow for data to be aggregated based on its locality and combining multiple relative localities into larger groups, demonstrating the underlying aggregation capabilities while also displaying the architectural support for specialised viewers. This lets us take advantage of the naturally occurring biclusters in the data. The map-based visualisation approach allows us to easily combine these biclusters across geographic bounds, to create new areas of interest without the need for additional attributes separating the data into these areas of interest.

1.4 Data Imputation

ConceptCloud's data exploration is heavily reliant on the quality of the input data. Often with man-made semi-structured data there are missing explicit attributes as they are implicitly present elsewhere. While this is acceptable when a person is working with the data, it is far more difficult to overcome these shortcomings with a lattice-based exploration. This is because without an attribute being explicitly present in an object's context, the object will simply not be included in the part of the lattice the aforementioned attribute is located in. Exploration of this location on the lattice will logically then not include exploration of this object.

An easy example of this comes up in the wine review data set, as shown in Figure 1.4. A person generally knows a Chenin Blanc is a white wine, and that Stellenbosch is in South Africa. It follows that a wine with the attributes Chenin Blanc and Stellenbosch should too have the attributes South Africa and white wine, and thus South African White Wine. But unless this is explicitly stated in the input data ConceptCloud will not include the object in the South African White Wine's portion of the lattice, even though the attribute is implicitly present.

This is the case in Figure 1.4 wherein the wine type attribute is missing, as well as the Country attribute, even though the attributes are stating that the wine is a Chenin Blanc from Stellenbosch. ConceptCloud does not know that a Chenin Blanc is a white wine, nor that Stellenbosch is a location in South Africa.

```
{
"wine_id": "11689",
"greater_then_three": "3 Star Rating Or More",
"greater_then_two_point_five": "2.5 Star Rating Or More",
"greater_then_two": "2.0 Star Rating Or More",
"prod_name": "Mont Destin",
"winery": "Mont Destin",
"range_name": "Mont Destin range",
"brand": "Chenin Blanc",
"winetype": "",
"vintage": "2007",
"name": "Mont Destin Mont Destin range Chenin Blanc 2007",
"cap": "Cork",
"variant_1": "Chenin blanc",
"variant_1_percent": "100",
"other": "NO",
"price": "59",
"margin_rating": "3.5",
"vintage_rating_100": "",
"vintage_rating": "3.0",
"blurb": "06 less beguiling than pvs, still ample charms through
at 14.5% alc, with pineapple & mineral signature.",
"winemaker": "Samantha Bürgin, advised by Bruwer Raats (Jan 2003)",
"viticulturist": "Bertus de Clerk since 2006",
"additional_info": "7ha (cab, cinsaut, grenache, mourvèdre, shiraz,
viongnier) 15t/1000cs 80% red 20% white",
"wine_of_origin_wine": "Stellenbosch",
"wine_of_origin_producer": "Paarl",
"region": "",
"area": "Stellenbosch",
"Country": "",
"alcohol": "13.5",
"blend": "Not Blended"
},
```

Figure 1.4: Example Chenin Blanc wine *JSON* object with missing data

We wish to extend ConceptCloud to accept supplementary data to enable this form of reasoning and attribute completion. The most obvious choices of supplementary data types are taxonomies, thesauri and ontologies.

The taxonomy referred to in the context of this thesis is that of a *hierarchical taxonomy* or *containment hierarchy*. This taxonomy is most easily conceptualised as a Concept Hierarchy, a tree of concepts and subconcepts linked by each child being a subconcept of the parent with an ‘is a’ relation. In our example we have the following Concept Hierarchy:

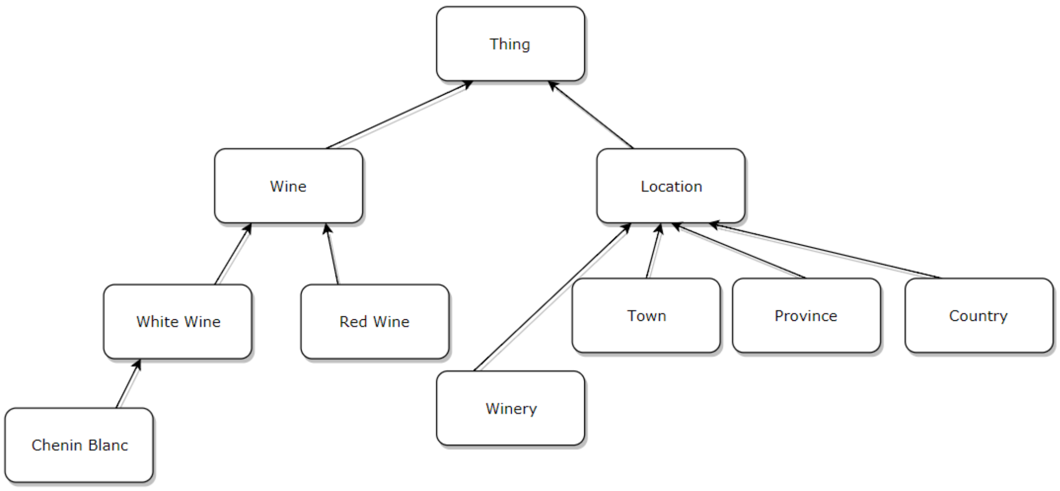


Figure 1.5: Example wine taxonomy

Figure 1.5 shows a summarised taxonomy for the wine data shown in Figure 1.4, each arrow denotes a ‘is a’ relation, that is a Chenin Blanc is a White wine, is a wine, is a thing. In our location subtree, we begin to have some trouble with the lack of expressivity in the ‘is a’ relation. We intuitively know that a wine has a winery and that there is a relation between a country, a winery, a town and a province. That is, in our example shown in Figure 1.4 we see the Wine, ‘Mont Destin, Mont Destin range Chenin Blanc 2007’ has a winery, and has an area Stellenbosch, and we know that Stellenbosch is in the Western Cape, which is in South Africa.

To state that information however, we need more information, that we cannot express in just a taxonomy. A meronomy, that is, a hierarchical structure that makes use of ‘has a’ relations could be used in stating some of the information in our example, as shown in Figure 1.6. A wine has a winery, a winery has a town, a province has a country. In our example this would be ‘Mont Destin, Mont Destin range Chenin Blanc 2007’ has a winery ‘Mont Destin’ which has a town or area ‘Stellenbosch’, which has a province ‘Western Cape’, which has a Country ‘South Africa’. However, it is clear that the ‘has a’ relation in ‘Wine has a Winery’ has a different meaning from ‘has a’ in ‘Province has a Country’, the former being ‘has origin’, and the latter meaning ‘is located in’.

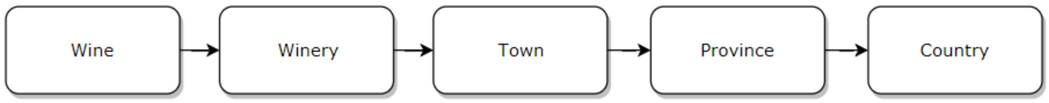


Figure 1.6: Meronomy for wine example

With this example in mind, it is beginning to become apparent we will require a more expressive relationship between words. We need to map the word 'town' to 'area' as in our data set shown in Figure 1.4, we see that Stellenbosch is given as an area, even though we know it is a town. A thesaurus is typically able to bridge this type of gap. Thesauri typically contain hierarchical information in a similar way to taxonomies, that is, information is ordered into some sort of hierarchy. Still, the ordering typically makes use of 'is broader' or 'is narrower' terminology, however, we can view this as a parent-child type relation, as in a taxonomy organised with 'is a' relations. Thesauri differ in that they are able to provide an 'equivalence' expression between concepts.

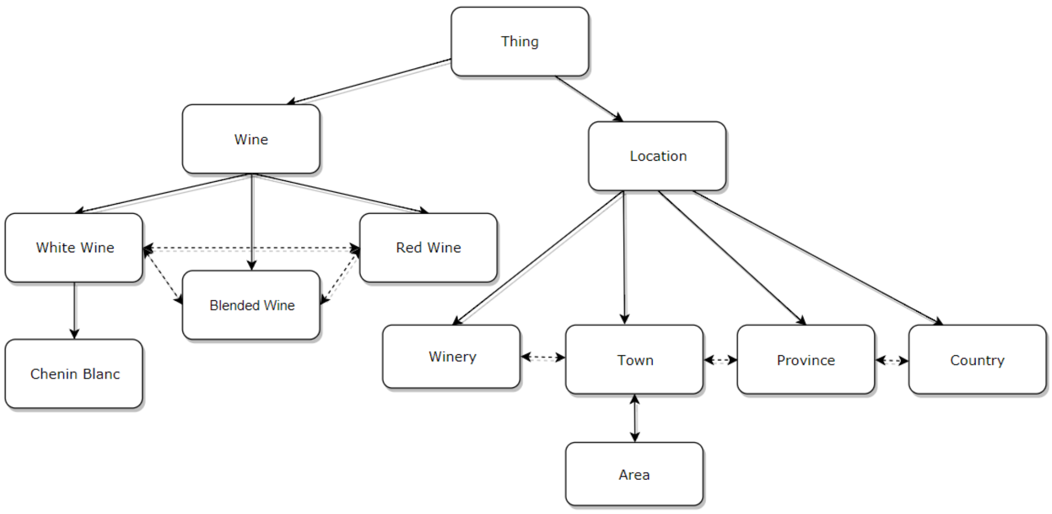


Figure 1.7: Thesaurus for wine example

Figure 1.7 shows the wine example in a thesaurus. The solid down arrows indicate the bottom term being more specific than the term above, while the double-headed solid arrow shows equivalent terms, as is the case with Town and Area. The dotted arrows denote a related to relation. The problem with only a thesaurus for the ConceptCloud application is we wish to mine these relationships for additional information, as such, we will require more specific relationships then 'is related to', as that gives us very little information to work with. It follows then from our wine example that we will at very least require something that has at least the expressivity of both a thesaurus and a meronomy.

This need for expressivity leads us to an ontology. While more heavyweight then our previous examples, an ontology has far more expressive relationships we can use to supplement the existing information. Additionally, ontologies support further reasoning functions for data inference, which can be used to drive the data imputation process, where data imputation refers to the enrichment of the underlying data set, by way of adding in additional data for existing items.

This leads to the third research question: How can an ontology be paired with a reasoner and a ConceptCloud interface in order to enrich the formal concept lattice based data set with entailed and inferred attributes?

Semi-structured data often contains free-text attributes which may provide additional information to be used in the attribute completion process. This information may come from simple key-phrase extraction if the free-text corresponds to a more controlled vocabulary or more advanced data extraction and reasoning techniques covered in Section 4.4. I extract these extra attributes from the free-text to allow for them to be used explicitly in the data exploration architecture.

The data imputation changes are aimed at improving the quality of the data exploration by providing improvements to the underlying data.

1.5 Structure of thesis

This introduction provides the context and problem statement and presents the research goals of this thesis. Chapter 2 provides further background into the formalisms and related work used in this thesis. A brief overview of formal concept analysis, its application to semi-structured data, lattice generation and lattice-based navigation, are given. Related work, other formal concept analysis based data exploration tools, as well as ConceptCloud, are presented by way of use case examples. A brief overview of the natural language techniques used in ConceptCloud as well as their application to an example dataset is shown. Finally, the description logic formalism and reasoner used are covered. The description logic is presented alongside an example ontology.

Chapter 3 covers the architectural changes made, maintenance performed, extensions to the API, and the context for the changes, with a brief description of the current system implementation. A short evaluation of the effect on run-time the architectural changes caused is provided. This is followed by a typical ConceptCloud use case for a large semi-structured data set, wherein a brief investigation is performed to illustrate the general functionality of the ConceptCloud tool.

Finally, architectural support for a specialised map-based visualisation is presented, the changes in search functionality, data representation, the advent of bicluster based navigation in ConceptCloud. A brief data set exploration of Crime Data in South Africa, demonstrating the underlying support for specialised viewers and alternate data aggregation is also included.

Chapter 4 covers the data imputation extensions made to the ConceptCloud tool. The context for the extensions is provided, and the general approach to data extraction from the ontology and lattice are covered. A more specific approach and algorithm are presented, followed by further implementation details and a case study.

Chapter 5 covers the extended evaluations for each extension to ConceptCloud. Results for each evaluation are presented alongside an analysis and discussion.

Chapter 6 covers the conclusions drawn from the work conducted in this thesis as well as recommendations for future work and possible further research highlighted by the work done in this thesis.

1.6 Summary

In this chapter, we covered a brief introduction to ConceptCloud, the formal concept analysis based semi-structured data exploration tool.

We have established the scope of this thesis, that is the three main updates added to ConceptCloud to enable improved data exploration by improving the underlying architecture on which ConceptCloud runs, ConceptCloud's ability to visualise specialised data and the quality of the data driving the exploration.

A structure outlining the contents of the chapters in the thesis was provided alongside a short list of the published works.

Publications

This work has lead to the aforementioned publications, *Scaling the ConceptCloud browser to large semi-structured data sets* [3], based on the work covered in Chapter 3, primarily focused on Sections 2.3-3.4.

ConceptCloud 2.0: Visualisation and Exploration of Geolocation-Rich Semi-Structured Data Sets [9] co-authored with Tiaan Du-Toit is based on work covered in Sections 3.7-3.8.

Chapter 2

Background

This background chapter covers a basic introduction to the core formalisms used in ConceptCloud and this thesis, each section providing an introduction and a brief motivation for the use of the formalism in ConceptCloud.

2.1 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, and the columns denote attributes. The incidence relation I indicates which objects in the table have which attributes.

Definition 1. *A binary 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.*

Based on the data presented in the wine example in Figure 1.4, we could consider identifying the following objects and attributes:

$$\begin{aligned}\mathcal{O} :=& \{\textsf{Mont Destin range Chenin Blanc 2007},\\ & \textsf{Bergsig range Chenin Blanc 2006}\}\\ \mathcal{A} :=& \{\textsf{wine\_id,prod\_name,winery,range\_name,brand,}\\ & \textsf{winetype,vintage,name,cap,variant\_1,variant\_1\_percent,}\\ & \textsf{variant\_2, variant\_2\_percent,other,price,margin\_rating,vintage\_rating\_100,}\\ & \textsf{vintage\_rating,blurb,winemaker,viticulturist,additional\_info,}\\ & \textsf{wine\_of\_origin\_wine,wine\_of\_origin\_producer,region,area,country,}\\ & \textsf{alcohol,blend}\}\end{aligned}$$

However, since a binary formal context has binary attributes, rather then many-valued attributes, for example, an attribute that has values from an entire domain, i.e. strings or integers, I have to flatten each attribute-value pair into a new attribute that each object either could or could not have. In a binary cross table this is indicated by the presence of a cross. A subsection of this flattened context is presented below in Figure 2.1.

wine_id	varietal: Chenin Blanc	vintage: 2006	vintage: 2007	area: Stellenbosch	area: Worcester
Mont Destin range Chenin Blanc 2007	X		X	X	
Bergsig Estate Chenin Blanc 2006	X	X			X

Figure 2.1: Subtable of a binary context showing attributes for the wine objects Mont Destin Range Chenin Blanc 2007 and Bersig Estate Chenin Blanc 2006

The corresponding formal context for the subtable presented in Figure 2.1 is as follows:

$$
\begin{aligned}
\mathcal{O} :=& \{\text{Mont Destin range Chenin Blanc 2007},\\
& \text{Bergsig range Chenin Blanc 2006}\}\\
\mathcal{A} :=& \{\text{varietal:Chenin Blanc, vintage:2006, vintage:2007, area:Stellenbosch,}\\
& \text{area:Worcester}\}\\
\mathcal{I} :=& \{(\text{Mont Destin range Chenin Blanc 2007, varietal:Chenin Blanc}),\\
& (\text{Bergsig range Chenin Blanc 2006, varietal:Chenin Blanc}),\\
& (\text{Mont Destin range Chenin Blanc 2007, vintage:2007}),\\
& (\text{Bergsig range Chenin Blanc 2006,vintage:2006}),\\
& (\text{Mont Destin range Chenin Blanc 2007, area:Stellenbosch}),\\
& (\text{Bergsig range Chenin Blanc 2006,area:Worcester})\\
& \}
\end{aligned}
$$

Concept flattening refers to the process of turning a many valued context, into a binary context. That is, for each possible $a \in \mathcal{A}$, with $v_1 \ldots v_n \in a$, I create new attributes with the form $a_{v_1} \ldots a_{v_n} \in \mathcal{A}$. This process is sometimes referred to as nominal conceptual scaling.

An example of this can be seen in Figure 2.1, where the vintage value pairs were mapped to vintage:2006, vintage:2007. For each possible vintage value, each wine_id object could either have, (as indicated by an X) or not have (as indicated by an empty cell in the binary cross table).

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 $\alpha(O) = \{a \in \mathcal{A} | \forall o \in O : (o, a) \in \mathcal{I}\}$, 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 sets 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.

Taking our flattened example from Figure 2.1 we can see for the objects:

$$\begin{aligned} O := \{&\textsf{Mont Destin range Chenin Blanc 2007}, \\ &\textsf{Bergsig range Chenin Blanc 2006}\} \end{aligned} \tag{2.1}$$

The common attribute is {varietal:Chenin Blanc}, and the common objects for the attribute set $A =$ varietal:Chenin Blanc, are O.

Definition 3. *Let $\mathcal{C}$ be a context. Then $c = \langle O, A\rangle$ 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 $B(\mathcal{C})$.*

Let CB $= \langle O, A\rangle$ as in the example above. Then CB is a concept of the context given in Figure 2.1. Further let CBS $:= \langle A, E\rangle$ where
$A :=$ {Mont Destin range Chenin Blanc 2007} and
$E :=$ {varietal: Chenin Blanc, area: Stellenbosch, vintage:2007}. And let CBW $:= \langle C, D\rangle$ where
$C :=$ {Bergsig range Chenin Blanc 2006} and
$D :=$ {varietal: Chenin Blanc, area: Worcester, vintage:2006}. Then both CBS and CBW are concepts. Concepts are partially ordered by the inclusion of extents such that the extent of a concept includes the extent of all of its subconcepts; dually, the intent of a concept includes the intent of all its superconcepts.

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

Clearly both CBS and CBW are subconcepts of the concept CB. That can be read as Chenin Blancs from Stellenbosch and Chenin Blancs from Worcester are both Chenin Blanc subconcepts.

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 [13]. 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.

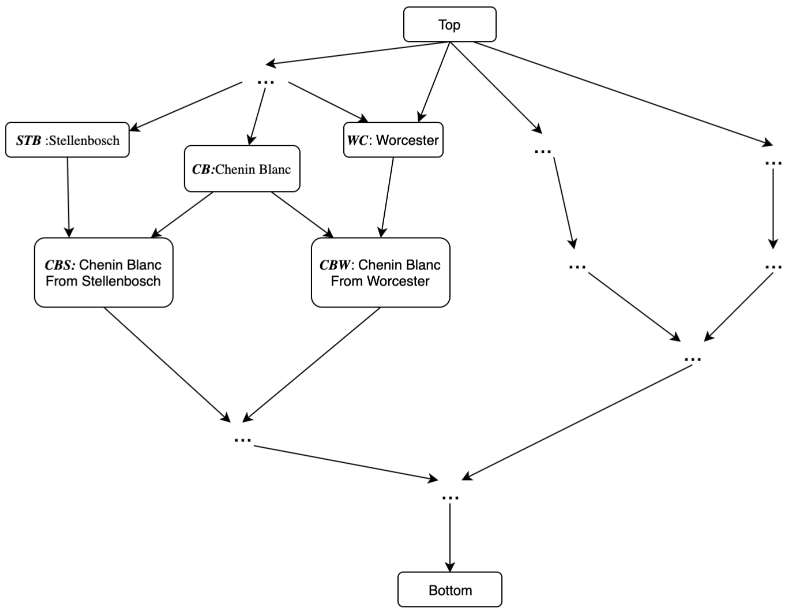

Top
...
STB :Stellenbosch
*CB:*Chenin Blanc
WC: Worcester
CBS: Chenin Blanc From Stellenbosch
CBW: Chenin Blanc From Worcester
Bottom

Figure 2.2: Example concept lattice for the ongoing wine example, showing possible complex concepts CBS and CBW

A possible lattice for the concepts CB, CBS,CBW is shown in Figure 2.2 above. The lattice is left incomplete for ease of illustration. The important thing to note is the lattice position of the Chenin Blanc subconcepts, CBS and CBW, that is, Chenin Blancs from Stellenbosch and Chenin Blancs from Worcester respectively, in relation to the Chenin Blanc Concept CB. Additionally, one can note that the location based subconcepts CBS,CBW are also below corresponding parent location concepts, which in this case, denotes all the objects with attributes Stellenbosch or Worcester, respectively. That is:

$$\mathcal{STB} := \langle \omega(Stellenbosch), Stellenbosch \rangle$$
$$\mathcal{WC} := \langle \omega(Worcester), Worcester \rangle$$

The concepts in the lattice range from the top to bottom concepts. Where the top concept is all objects with attributes shared by all objects (usually the empty set) in the context, and the bottom is all objects that have all the attributes in the context (usually the empty set).

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* $\{\langle A_i, B_i \rangle | i \in I\} \subseteq$

$B(\mathcal{C})$ of concepts are given by:

$$\begin{aligned}\bigwedge_{i\in I}(O_i, A_i) &= (\bigcap_{i\in I} O_i, \alpha(\omega(\bigcup_{i\in I} A_i)))\\ \bigvee_{i\in I}(O_i, A_i) &= (\omega(\alpha(\bigcup_{i\in I} O_i)), \bigcap_{i\in I} A_i)\end{aligned}$$

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, and 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, and 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 [36]. For a detailed introduction to FCA see [13].

2.2 FCA Based Data Exploration Tools

Formal concept lattice based visualisation and data exploration tools have existed for many years. Generally however, all share the same fundamental flaw, they visualise the lattice directly and thus scale poorly. Larger lattices can become exceedingly memory intensive, and progressively less intuitive as they scale in size. These scaling issues are mainly a direct consequence of each software tool trying to maintain the entire concept lattice. This becomes prohibitively resource-intensive for large data sets.

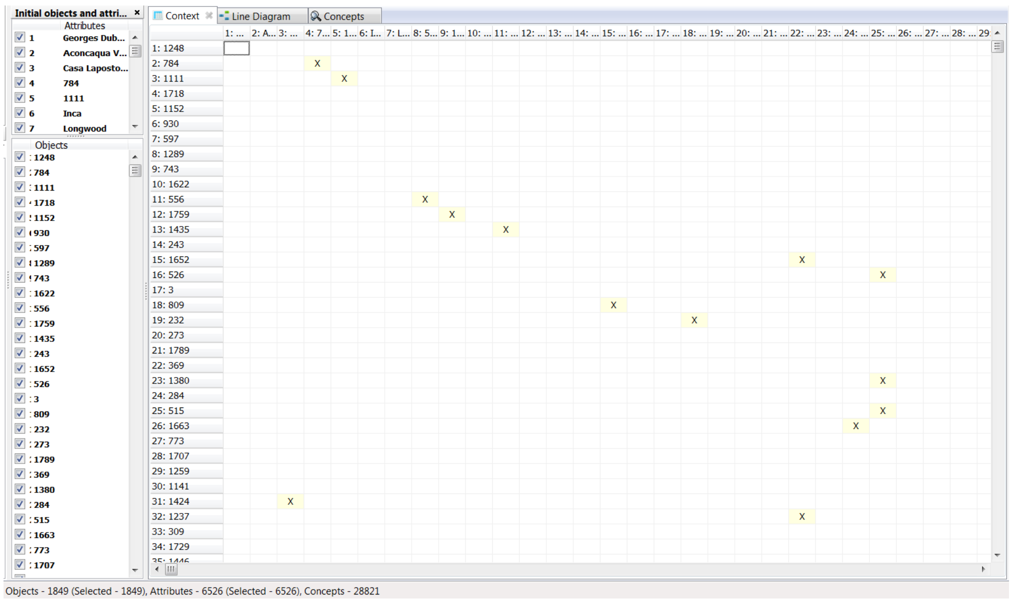
Three data exploration tools are explored with the goal of illustrating their usage on a larger dataset and the scaling issues they encounter.

FCART

The Formal Concept Analysis Research Toolbox (FCART) was designed in 2014, specifically for the analysis of semi-structured, and unstructured data [4]. FCART provides the user with tools for iterative analysis of their data with adjustable data queries and analytic artefacts. It allows the user to read in their data in semi-structured or lattice format and creates a binary context. It then allows for the creation of a concept lattice to visualise the data.

While FCART is able to process large data files, with some delay even their initial versions processed lattices with 20000 concepts, the binary context and visualisations produced do not provide the user with any holistic intuition to the data. This is due to the visualisations of the entire data set becoming too cluttered for the user to be able to discern not only the labels in the visualisation but the links between labelled items. Further exploration of the data requires prior knowledge of the data to form data queries properly. Properly formed data queries, or zooming into small subsets of the data can provide some understanding of the contents of these sub data sets. However it still becomes difficult to link these subsets to each other, which leads to understanding isolated subsets of the data.

For illustration purposes I ran the FCART tool on a subset of our wine data set, with 1849 wine objects, and objects having 66 attributes. When this is inserted into FCART the mapping to a binary cross table results in 6526 binary attributes. This is as each possible value for each of the 66 many-valued (unflattened) attribute classes is mapped to a class value pair representation in the binary cross table. This leads to an attribute explosion for free-text fields, as they could potentially all be unique in value, leading to each object having a unique attribute. This makes using the context table unintuitive and cumbersome, additionally leading to very poor performance.



Attributes	
1	Georges Dub...
2	Aconcaqua V...
3	Casa Laposto...
4	784
5	1111
6	Inca
7	Longwood

Objects: 1248, 784, 1111, 1718, 1152, 930, 597, 1289, 743, 1622, 556, 1759, 1435, 243, 1652, 526, 3, 809, 232, 273, 1789, 369, 1380, 284, 515, 1663, 773, 1707

	1: ...	2: A...	3: ...	4: 7...	5: 1...	6: I...	7: L...	8: 5...	9: 1...	10: ...	11: ...	12: ...	13: ...	14: ...	15: ...	16: ...	17: ...	18: ...	19: ...	20: ...	21: ...	22: ...	23: ...	24: ...	25: ...	26: ...	27: ...	28: ...	29:
1: 1248																													
2: 784				X																									
3: 1111					X																								
4: 1718																													
5: 1152																													
6: 930																													
7: 597																													
8: 1289																													
9: 743																													
10: 1622																													
11: 556								X																					
12: 1759									X																				
13: 1435											X																		
14: 243																													
15: 1652																						X							
16: 526																									X				
17: 3																													
18: 809															X														
19: 232																		X											
20: 273																													
21: 1789																													
22: 369																													
23: 1380																									X				
24: 284																													
25: 515																									X				
26: 1663																								X					
27: 773																													
28: 1707																													
29: 1259																													
30: 1141																													
31: 1424			X																										
32: 1237																						X							
33: 309																													
34: 1729																													
35: 1446																													

Objects - 1849 (Selected - 1849), Attributes - 6526 (Selected - 6526), Concepts - 28821

Figure 2.3: Large FCART context table for a data set of 1849 wine objects

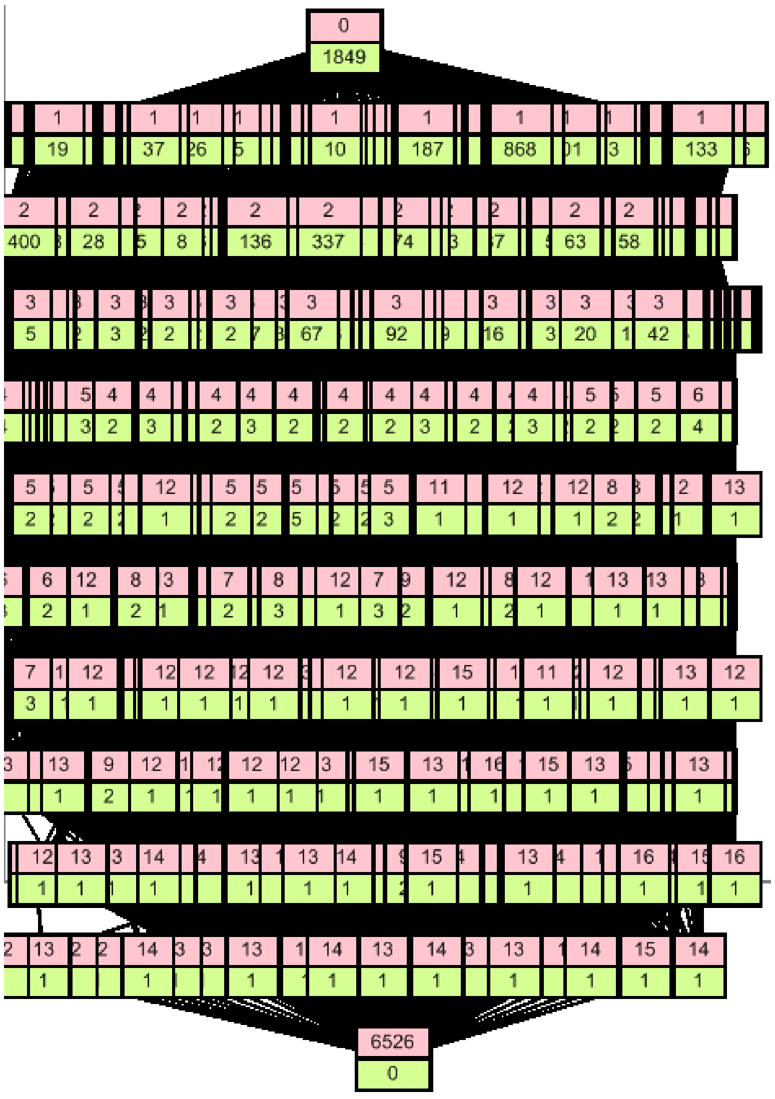


Figure 2.4: Large FCART concept lattice for a data set of 1849 wine objects

As we can see in Figure 2.3 the context table, note the scroll bars in the corners. This context table is relatively sparse and rather large, bearing in mind that this is for objects with roughly 60 many-valued attributes. Figure 2.4 shows the lattice generated for the data set. Note that there are 6526 attributes, and although visualizing a two-dimensional lattice of this size is exceedingly difficult, a lattice is generated and may be queried. It is however incapable of providing any intuition of our data, as the majority of the attribute and object labels in the lattice cannot be properly displayed, and the density of the links between lattice items cause them to display as a black wall from which individual links cannot be discerned.

The only recourse is then to find the attributes of interest in the exceedingly large flattened list, deselect all other attributes and focus only on the selected ones. In this way you can 'zoom' in and out of the selected concepts.

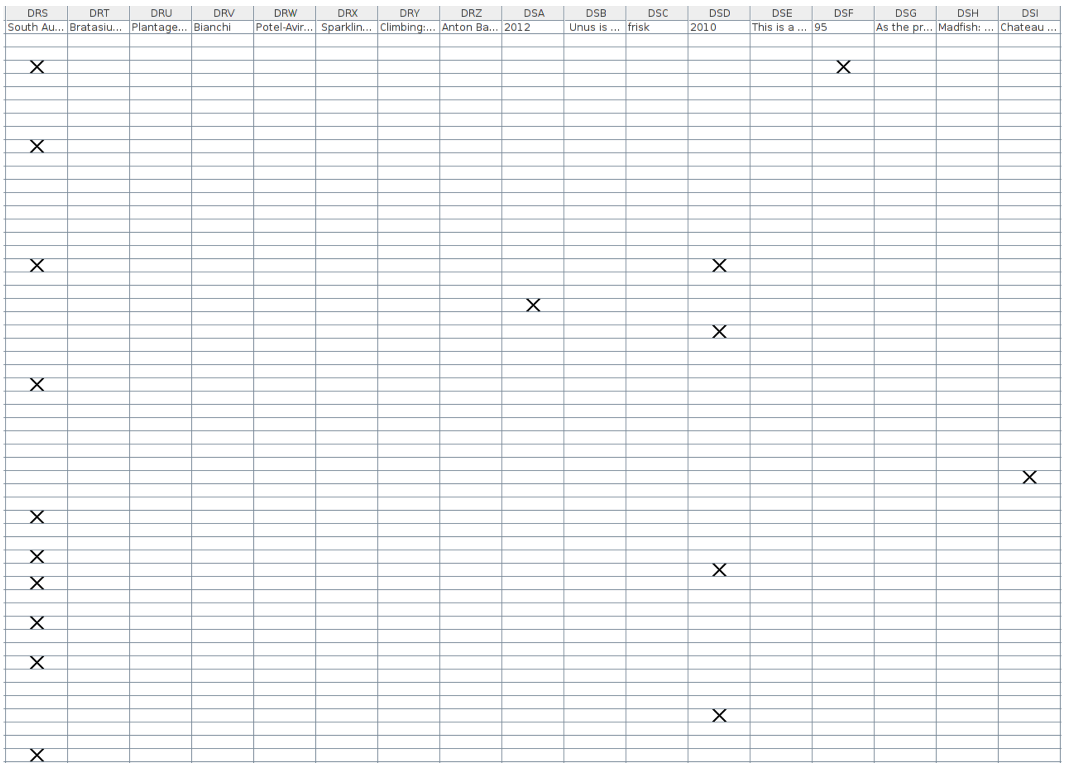
This operation, while valuable, is still fairly difficult to perform as you need to sift through all the flattened attributes to find the binary pairs, such as those in Figure 2.1, of interest.

ConExp

Conexp or Concept Explorer is a concept lattice and context generation tool developed from 2000. It allows the user to explore binary contexts and visualise them in a lattice [35]. It was never made to scale to large numbers of concepts [1].

ConExp was chosen as it provides the ability to generate an implication base. An implication base, of completed data only, would be useful to use to enrich the incomplete data, an operation which is part of the goal of this thesis.

I used the same data set with 1849 wine objects, as with FCART, Conexp was able to visualise the context table but unable to generate a lattice.



DRS	DRT	DRU	DRV	DRW	DRX	DRY	DRZ	DSA	DSB	DSC	DSD	DSE	DSF	DSG	DSH	DSI
South Au...	Bratasiu...	Plantage...	Bianchi	Potel-Avir...	Sparklin...	Climbing:...	Anton Ba...	2012	Unus is ...	frisk	2010	This is a ...	95	As the pr...	Madfish: ...	Chateau ...
×													×			
×																
×											×					
								×								
											×					
×																
																×
×																
×																
											×					
×																
×																
×																
											×					
×																

Figure 2.5: ConExp binary context table for a data set of 1849 wine objects

As we see in Figure 2.5 I have the same visualisation problems with a binary context table as in Figure 2.3. Displaying as a binary context table means I have to display each possible attribute-value combination in the table, leading to attribute explosion, and a poor visual representation of our data.

As previously mentioned, ConExp does provide the ability to compute a Duquenne-Guigues base of implications [25], that is, a minimal base of implications that hold across the context. The implications generated by the ConExp tool are of the following form:

No ⟨Number Of objects ⟩ Premise Conclusion

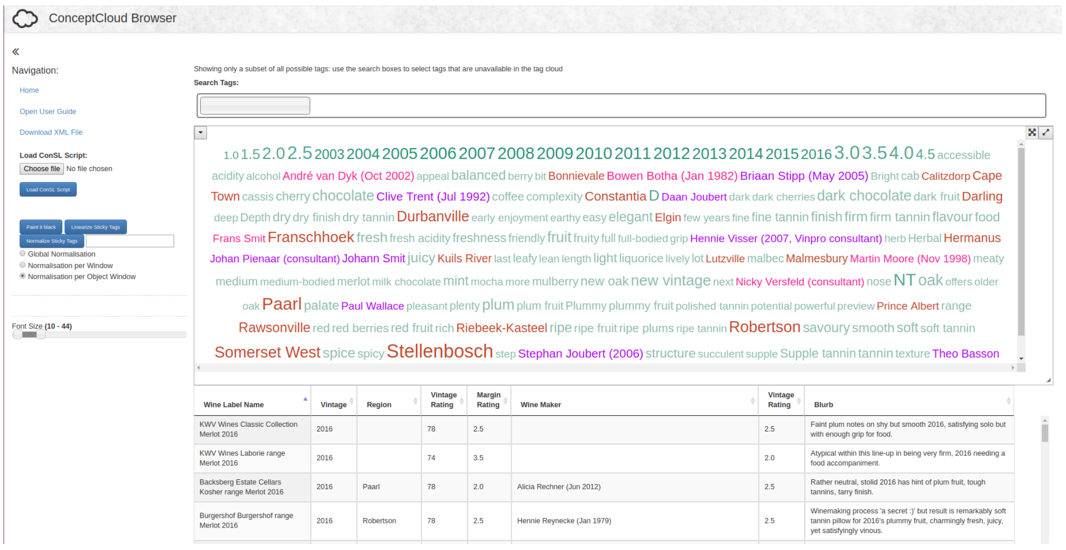
Where 'No' simply denotes the number within the implication base, and the premise and conclusion are sets of attribute names that occur within each.

It is important to note that as ConExp processes only binary contexts, the mapping of a semi-structured data set into a binary context, which maps each possible attribute value to a new attribute which each object may or may not have, also profoundly impacts the performance of the generation of the implication base. Additionally, this operation appears to be single-threaded, which leads to it scaling exponentially in time with the exponential attribute growth in the context.

As such, I was unable to generate an implication base for the wine data set example.

ConceptCloud

ConceptCloud differs in its visual representation of the data but still allows the user to make use of the formal concept lattice to draw additional insight. ConceptCloud achieves this by representing data as an interactive tag cloud, that enables the users to step through the concept lattice by selection and deselection of tags. Once again, I made use of the small wine data set with 1849 wine objects, each with 66 many-valued attributes. ConceptCloud provides us with a tag cloud visualisation and a context table.



Showing only a subset of all possible tags: use the search boxes to select tags that are unavailable in the tag cloud

Search Tags:

1.0 1.5 2.0 2.5 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 3.0 3.5 4.0 4.5 accessible acidity alcohol André van Dyk (Oct 2002) appeal balanced berry bit Bonnievale Bowen Botha (Jan 1982) Briaan Stipp (May 2005) Bright cab Calitzdorp Cape Town cassis cherry chocolate Clive Trent (Jul 1992) coffee complexity Constantia D Daan Joubert dark dark cherries dark chocolate dark fruit Darling deep Depth dry dry finish dry tannin Durbanville early enjoyment earthy easy elegant Elgin few years fine fine tannin finish firm firm tannin flavour food Frans Smit Franschhoek fresh fresh acidity freshness friendly fruit fruity full full-bodied grip Hennie Visser (2007, Vinpro consultant) herb Herbal Hermanus Johan Pienaar (consultant) Johann Smit juicy Kuils River last leafy lean length light liquorice lively lot Lutzville malbec Malmesbury Martin Moore (Nov 1998) meaty medium medium-bodied merlot milk chocolate mint mocha more mulberry new oak new vintage next Nicky Versfeld (consultant) nose NT oak offers older oak Paarl palate Paul Wallace pleasant plenty plum plum fruit Plummy plummy fruit polished tannin potential powerful preview Prince Albert range Rawsonville red red berries red fruit rich Riebeek-Kasteel ripe ripe fruit ripe plums ripe tannin Robertson savoury smooth soft soft tannin Somerset West spice spicy Stellenbosch step Stephan Joubert (2006) structure succulent supple Supple tannin tannin texture Theo Basson

Wine Label Name	Vintage	Region	Vintage Rating	Margin Rating	Wine Maker	Vintage Rating	Blurb
KWV Wines Classic Collection Merlot 2016	2016		78	2.5		2.5	Faint plum notes on shy but smooth 2016, satisfying solo but with enough grip for food.
KWV Wines Laborie range Merlot 2016	2016		74	3.5		2.0	Atypical within this line-up in being very firm, 2016 needing a food accompaniment.
Backsberg Estate Cellars Kosher range Merlot 2016	2016	Paarl	78	2.0	Alicia Rechner (Jun 2012)	2.5	Rather neutral, stolid 2016 has hint of plum fruit, tough tannins, tarry finish.
Burgershof Burgershof range Merlot 2016	2016	Robertson	78	2.5	Hennie Reynecke (Jan 1979)	2.5	Winemaking process 'a secret :)' but result is remarkably soft tannin pillow for 2016's plummy fruit, charmingly fresh, juicy, yet satisfyingly vinous.

Figure 2.6: ConceptCloud tag cloud and many-valued context table for a data set of 1849 wine objects

ConceptCloud makes use of the lattice for explorative search only, as such, it does not generate the entire lattice at any time. ConceptCloud makes use of the *Colibri* java library to generate only segments of the lattice it requires. These sections of the lattice are used and disposed of at each navigation step. As such, it is easily able to step through and visualise the data. Additionally, the Context table ConceptCloud displays, is a multi-valued context, so each attribute may have any particular value, which is more human readable than a binary context, and has the benefit of being far less taxing to display. Figure 2.6 shows the multi-valued context table, where instead of attempting to display each possible attribute-value combination I display only the attribute, and the object's attribute value, in this example, `Attribute:Vintage Value:2016` is displayed instead of `Attribute:Vintage2016`, `Value:  X`.

2.3 ConceptCloud

As mentioned in Sections 1, and 2.2, ConceptCloud [17] is a visualisation and exploration tool for semi-structured data sets, such as software revision control metadata or product reviews. ConceptCloud uses a concept lattice [13] built from the data set as underlying navigation structure but presents the data itself in the form of a tag cloud which concisely summarises the user's current selection in one view and allows further navigation through tag selection and deselection, without constricting them to predefined search paths.

ConceptCloud [17] allows the user to navigate, via tag clouds, through a data set in what is known as an *explorative search*. This type of exploration requires no predefined knowledge of the domain or data set. 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.

Definition 6. *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 focus concept can be further refined by iteratively adding elements to F. When an additional attribute is added to F, I update the focus concept by computing the meet, as per Theorem 1 (see Section 2.1), of the current focus concept c and the concept introduced by the additional attribute. In Section 3.2 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 I select further attributes, I move further down the lattice. If I deselect an attribute, I 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 [10] 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. This was shown to be a strong positive result and is discussed in [18].

ConceptCloud presents the data in the form of a tag cloud, where the frequency of each tag's attribute within the lattice, is taken to denote the tag's 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's size in a tag cloud is based on the frequency of its occurrence within the subselection of the data set, coloured differently based on its category (namely the type value of the attribute or object).

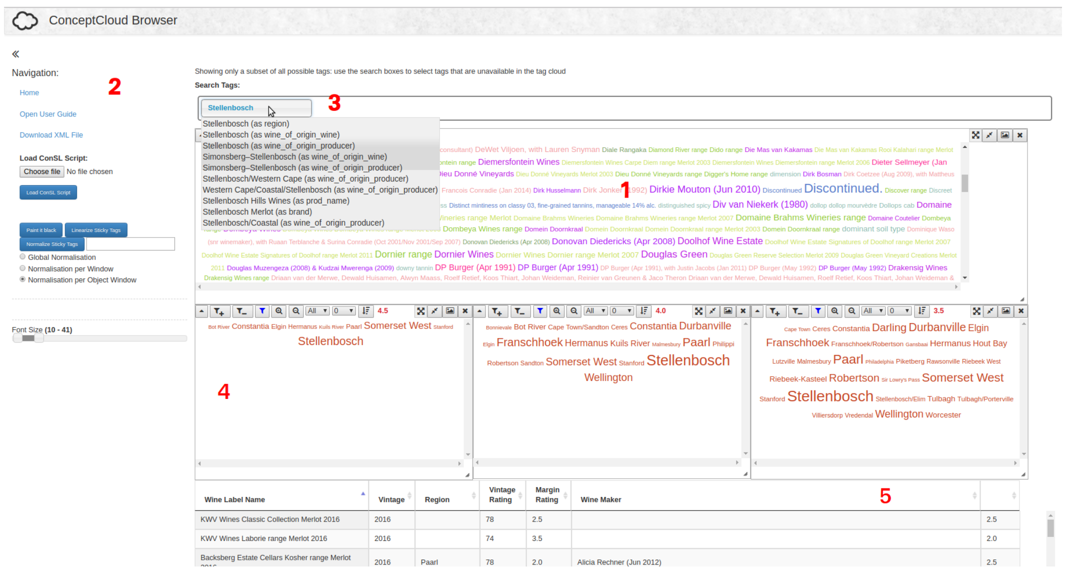
Tag clouds are constructed by taking the extent of the focus concept $c = \langle O, A \rangle$, then for each $o_i \in O$, determining its defining concept c_i (see Definition 5). I then collect all the intents of these defining concepts. These are the attributes I display in the tag cloud. Finally, I add the objects to the tag cloud so that they may be directly selected or searched within the tag cloud.

The initial focus concept has 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 7. *The tag cloud from a concept $c = (O,\ A) \in B(\mathcal{C})$ is defined as $\tau(c) = O \uplus \biguplus_{o \in O} \pi_A(\sigma(o))$.*

By constructing the tags 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.

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
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 2.7: ConeptCloud navigation user interface with numerically labelled components

The navigation user interface consists of the following components shown in Figure 2.7:

- 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 [16] 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.

- *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. It is important to note that the reduction of displayed tags and the search functionality were not part of the original ConceptCloud web application implementation, and were introduced as part of the scalable architecture covered in Chapter 3.

- *Sticky Tag Cloud Viewers* (4) are subwindows 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, e.g., 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 are multitudes of attributes for each object in the context table. Often too many attributes 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.

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 subwindows, which are additional tag cloud viewers with sticky tags. These subwindows 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 the 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 the sticky tag and new focus concept are not disjoint. Otherwise,

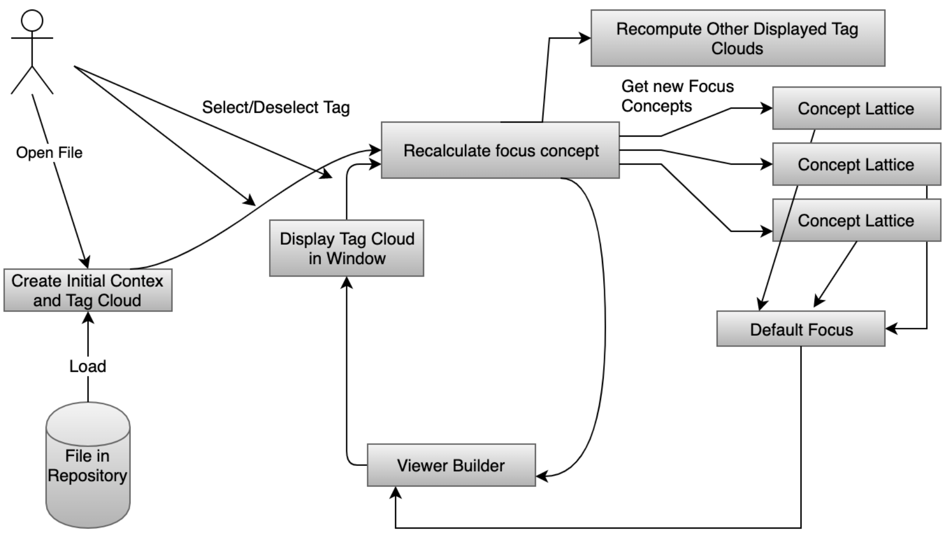


Figure 2.8: Navigation component interaction architecture

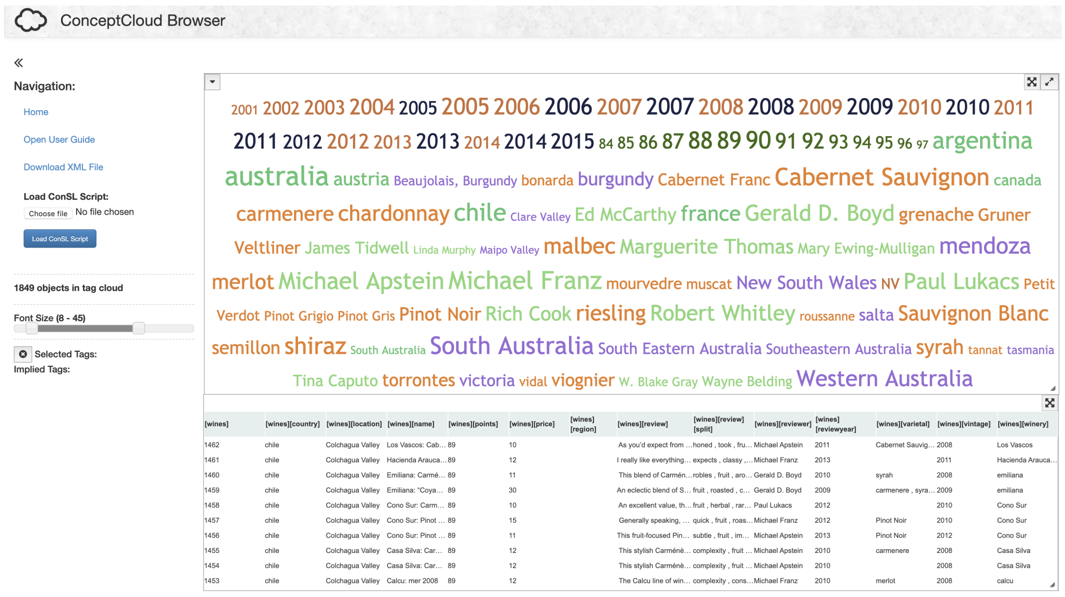
an empty viewer window is displayed. The architecture of the navigation subsystem is outlined in Figure 2.8.

Select / Deselect Tags 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. This update causes each tag cloud window/viewer to request the relevant section of the underlying concept lattice, should it exist, from their controllers. 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 deselection 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 2.3.

Wine Review Usage Example

In Figure 2.9 I show ConceptCloud running on a small wine review dataset of 1849 wine reviews. At this point we are at the point wherein ConceptCloud has created and displayed the initial context table and tag cloud. The explorative search process, further covered in Section 2.3, dictates we need to now select an attribute or tag. ConceptCloud will then update the tag cloud to display the new focus concept.

ConceptCloud does this by recalculating the focus concept in the underlying lattice, and then displaying the related tags, and hiding those tags that



ConceptCloud Browser

«

Navigation:

Home

Open User Guide

Download XML File

Load ConSL Script:

Choose file No file chosen

Load ConSL Script

1849 objects in tag cloud

Font Size (8 - 45)

Selected Tags:

Implied Tags:

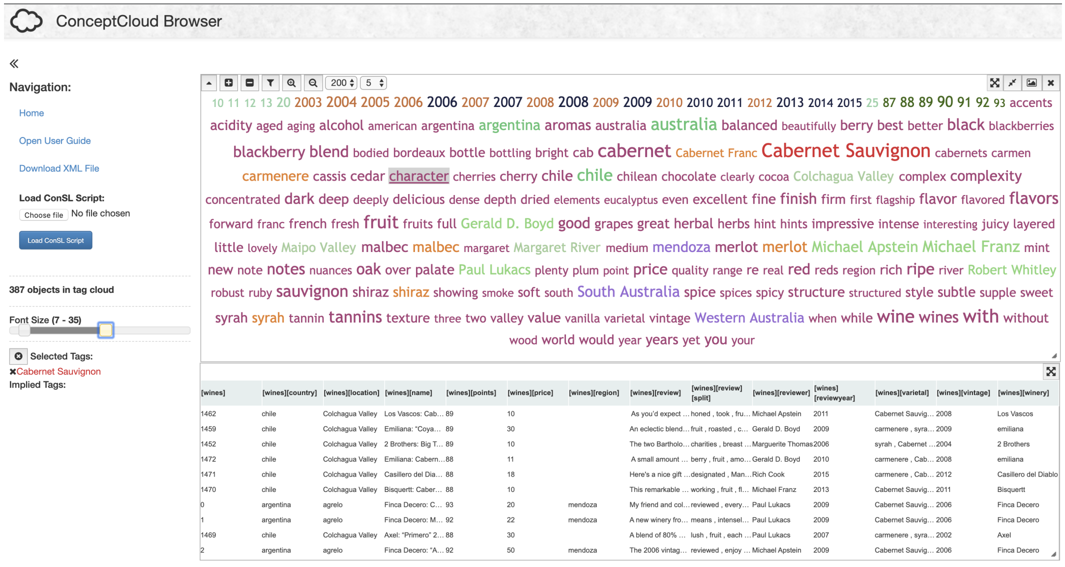
2001 2002 2003 2004 2005 2005 2006 2006 2007 2007 2008 2008 2009 2009 2010 2010 2011 2011 2012 2012 2013 2013 2014 2014 2015 84 85 86 87 88 89 90 91 92 93 94 95 96 97 argentina australia austria Beaujolais, Burgundy bonarda burgundy Cabernet Franc Cabernet Sauvignon canada carmenere chardonnay chile Clare Valley Ed McCarthy france Gerald D. Boyd grenache Gruner Veltliner James Tidwell Linda Murphy Maipo Valley malbec Marguerite Thomas Mary Ewing-Mulligan mendoza merlot Michael Apstein Michael Franz mourvedre muscat New South Wales NV Paul Lukacs Petit Verdot Pinot Grigio Pinot Gris Pinot Noir Rich Cook riesling Robert Whitley roussanne salta Sauvignon Blanc semillon shiraz South Australia South Australia South Eastern Australia Southeastern Australia syrah tannat tasmania Tina Caputo torrontes victoria vidal viognier W. Blake Gray Wayne Belding Western Australia

[wines]	[wines][country]	[wines][location]	[wines][name]	[wines][points]	[wines][price]	[wines][region]	[wines][review]	[wines][review][split]	[wines][reviewer]	[wines][reviewyear]	[wines][varietal]	[wines][vintage]	[wines][winery]
1462	chile	Colchagua Valley	Los Vascos: Cab…	89	10		As you'd expect from …	honed , took , fru…	Michael Apstein	2011	Cabernet Sauvig…	2008	Los Vascos
1461	chile	Colchagua Valley	Hacienda Arauca…	89	12		I really like everything…	expects , classy ,…	Michael Franz	2013		2011	Hacienda Arauca…
1460	chile	Colchagua Valley	Emiliana: Carmé…	89	11		This blend of Carmén…	robles , fruit , aro…	Gerald D. Boyd	2010	syrah	2008	emiliana
1459	chile	Colchagua Valley	Emiliana: "Coya…	89	30		An eclectic blend of S…	fruit , roasted , c…	Gerald D. Boyd	2009	carmenere , syra…	2009	emiliana
1458	chile	Colchagua Valley	Cono Sur: Carm…	89	10		An excellent value, th…	fruit , herbal , rar…	Paul Lukacs	2012		2010	Cono Sur
1457	chile	Colchagua Valley	Cono Sur: Pinot …	89	15		Generally speaking, …	quick , fruit , roas…	Michael Franz	2012	Pinot Noir	2010	Cono Sur
1456	chile	Colchagua Valley	Cono Sur: Pinot …	89	11		This fruit-focused Pin…	subtle , fruit , im…	Michael Apstein	2013	Pinot Noir	2012	Cono Sur
1455	chile	Colchagua Valley	Casa Silva: Car…	89	12		This stylish Carménè…	complexity , fruit …	Michael Apstein	2010	carmenere	2008	Casa Silva
1454	chile	Colchagua Valley	Casa Silva: Car…	89	12		This stylish Carménè…	complexity , fruit …	Michael Apstein	2010		2008	Casa Silva
1453	chile	Colchagua Valley	Calcu: mer 2008	89	12		The Calcu line of win…	complexity , cons…	Michael Franz	2010	merlot	2008	calcu

Figure 2.9: Example home screen of original web application ConceptCloud implementation for a reduced size wine review dataset.

are no longer relevant to the focus concept.

This corresponds to Figure 2.10, wherein I selected the Cabernet Sauvignon varietal tag.



ConceptCloud Browser

«

Navigation:

Home

Open User Guide

Download XML File

Load ConSL Script:

Choose file No file chosen

Load ConSL Script

387 objects in tag cloud

Font Size (7 - 35)

Selected Tags:

×Cabernet Sauvignon

Implied Tags:

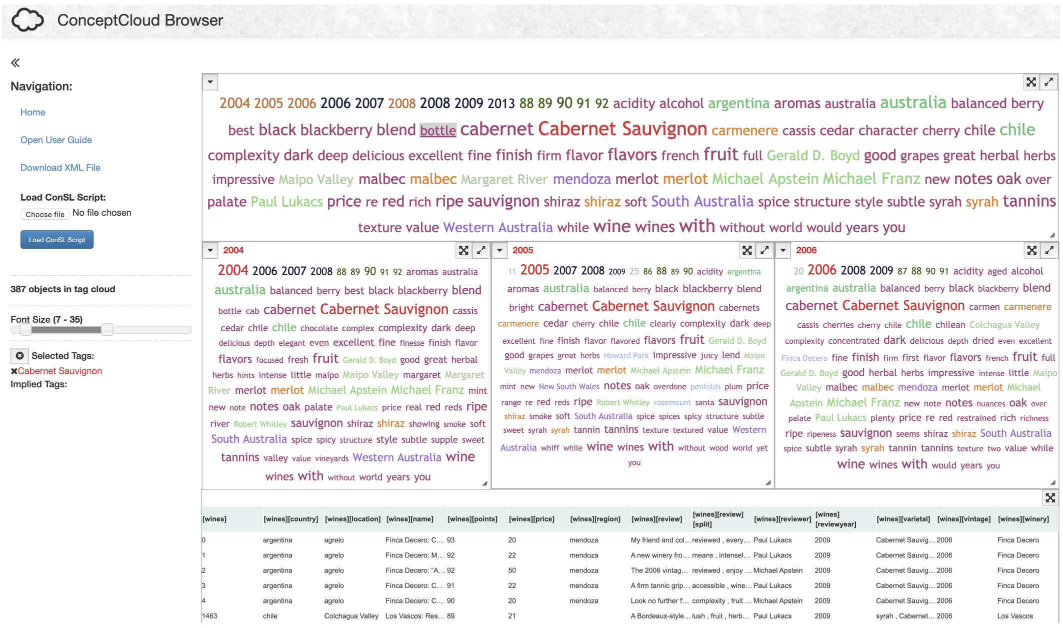
200 5

10 11 12 13 20 2003 2004 2005 2006 2006 2007 2007 2008 2008 2009 2009 2010 2010 2011 2012 2013 2014 2015 25 87 88 89 90 91 92 93 accents acidity aged aging alcohol american argentina argentina aromas australia australia balanced beautifully berry best better black blackberries blackberry blend bodied bordeaux bottle bottling bright cab cabernet Cabernet Franc Cabernet Sauvignon cabernets carmen carmenere cassis cedar character cherries cherry chile chile chilean chocolate clearly cocoa Colchagua Valley complex complexity concentrated dark deep deeply delicious dense depth dried elements eucalyptus even excellent fine finish firm first flagship flavor flavored flavors forward franc french fresh fruit fruits full Gerald D. Boyd good grapes great herbal herbs hint hints impressive intense interesting juicy layered little lovely Maipo Valley malbec malbec margaret Margaret River medium mendoza merlot merlot Michael Apstein Michael Franz mint new note notes nuances oak over palate Paul Lukacs plenty plum point price quality range re real red reds region rich ripe river Robert Whitley robust ruby sauvignon shiraz shiraz showing smoke soft south South Australia spice spices spicy structure structured style subtle supple sweet syrah syrah tannin tannins texture three two valley value vanilla varietal vintage Western Australia when while wine wines with without wood world would year years yet you your

[wines]	[wines][country]	[wines][location]	[wines][name]	[wines][points]	[wines][price]	[wines][region]	[wines][review]	[wines][review][split]	[wines][reviewer]	[wines][reviewyear]	[wines][varietal]	[wines][vintage]	[wines][winery]
1462	chile	Colchagua Valley	Los Vascos: Cab…	89	10		As you'd expect …	honed , took , fru…	Michael Apstein	2011	Cabernet Sauvig…	2008	Los Vascos
1459	chile	Colchagua Valley	Emiliana: "Coya…	89	30		An eclectic blend…	fruit , roasted , c…	Gerald D. Boyd	2009	carmenere , syra…	2009	emiliana
1452	chile	Colchagua Valley	2 Brothers: Big T…	89	10		The two Bartholo…	charities , breast …	Marguerite Thomas	2006	syrah , Cabernet …	2004	2 Brothers
1472	chile	Colchagua Valley	Emiliana: Cabern…	88	11		A small amount …	berry , fruit , amo…	Gerald D. Boyd	2010	carmenere , Cab…	2008	emiliana
1471	chile	Colchagua Valley	Casillero del Dia…	88	18		Here's a nice gift …	designated , Man…	Rich Cook	2015	carmenere , Cab…	2012	Casillero del Diablo
1470	chile	Colchagua Valley	Bisquertt: Caber…	88	10		This remarkable …	working , fruit , fl…	Michael Franz	2013	Cabernet Sauvig…	2011	Bisquertt
0	argentina	agrelo	Finca Decero: C…	93	20	mendoza	My friend and col…	reviewed , every…	Paul Lukacs	2009	Cabernet Sauvig…	2006	Finca Decero
1	argentina	agrelo	Finca Decero: M…	92	22	mendoza	A new winery fro…	means , intensel…	Paul Lukacs	2009	Cabernet Sauvig…	2006	Finca Decero
1469	chile	Colchagua Valley	Axel: "Primero" 2…	88	30		A blend of 80% …	lush , fruit , each …	Paul Lukacs	2007	carmenere , syra…	2002	Axel
2	argentina	agrelo	Finca Decero: "A…	92	50	mendoza	The 2006 vintag…	reviewed , enjoy …	Michael Apstein	2009	Cabernet Sauvig…	2006	Finca Decero

Figure 2.10: Example focussed concept tag cloud of original web application ConceptCloud implementation for a reduced size wine review dataset.

ConceptCloud is also able to visualise multiple points in the lattice at any time. It does so by allowing the user to open multiple tag clouds, each with their own focus concepts. These tag clouds are all linked to the focus concept of the main tag cloud viewer, that is, while each may have a separate focus concept, all those focus concepts are all subconcepts of the focus concept in the main tag cloud viewer. We see this in Figure 2.11, the main focus concept is that of {`varietal: Cabernet Sauvignon`}, and the focus concepts in the subsequent tag clouds are {`varietal:Cabernet Sauvignon`∩`vintage:2004`}, {`varietal: Cabernet Sauvignon` ∩ `vintage: 2005`} and {`varietal: Cabernet Sauvignon` ∩ `vintage:2006`} respectively. Each of which is a child concept of {`varietal:Cabernet Sauvignon`}. Here the concepts of each tag cloud are subconcepts of the main tag cloud's focus concept as per Definition 7.



2004 2005 2006 2006 2007 2008 2008 2009 2013 88 89 90 91 92 acidity alcohol argentina aromas australia australia balanced berry best black blackberry blend bottle cabernet Cabernet Sauvignon carmenere cassis cedar character cherry chile chile complexity dark deep delicious excellent fine finish firm flavor flavors french fruit full Gerald D. Boyd good grapes great herbal herbs impressive Maipo Valley malbec malbec Margaret River mendoza merlot merlot Michael Apstein Michael Franz new notes oak over palate Paul Lukacs price re red rich ripe sauvignon shiraz shiraz soft South Australia spice structure style subtle syrah syrah tannins texture value Western Australia while wine wines with without world would years you

2004

2004 2006 2007 2008 88 89 90 91 92 aromas australia australia balanced berry best black blackberry blend bottle cab cabernet Cabernet Sauvignon cassis cedar chile chile chocolate complex complexity dark deep delicious depth elegant even excellent fine finesse finish flavor flavors focused fresh fruit Gerald D. Boyd good great herbal herbs hints intense little maipo Maipo Valley margaret Margaret River merlot merlot Michael Apstein Michael Franz mint new note notes oak palate Paul Lukacs price real red reds ripe river Robert Whitley sauvignon shiraz shiraz showing smoke soft South Australia spice spicy structure style subtle supple sweet tannins valley value vineyards Western Australia wine wines with without world years you

2005

11 2005 2007 2008 2009 25 86 88 89 90 acidity argentina aromas australia balanced berry black blackberry blend bright cabernet Cabernet Sauvignon cabernets carmenere cedar cherry chile chile clearly complexity dark deep excellent fine finish flavor flavored flavors fruit Gerald D. Boyd good grapes great herbs Howard Park impressive juicy lend Maipo Valley mendoza merlot merlot Michael Apstein Michael Franz mint new New South Wales notes oak overdone penfolds plum price range re red reds ripe Robert Whitley rosemount santa sauvignon shiraz smoke soft South Australia spice spices spicy structure subtle sweet syrah syrah tannin tannins texture textured value Western Australia whiff while wine wines with without wood world yet you

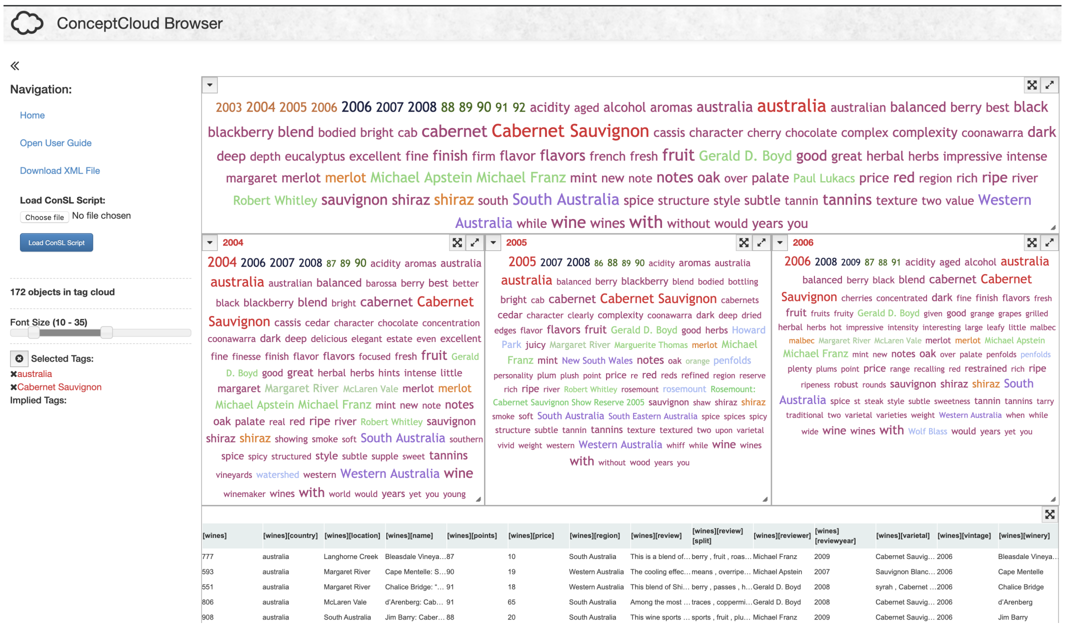
2006

20 2006 2008 2009 87 88 90 91 acidity aged alcohol argentina australia balanced berry black blackberry blend cabernet Cabernet Sauvignon carmen carmenere cassis cherries cherry chile chile chilean Colchagua Valley complexity concentrated dark delicious depth dried even excellent Finca Decero fine finish firm first flavor flavors french fruit full Gerald D. Boyd good herbal herbs impressive intense little Maipo Valley malbec malbec mendoza merlot merlot Michael Apstein Michael Franz new note notes nuances oak over palate Paul Lukacs plenty price re red restrained rich richness ripe ripeness sauvignon seems shiraz shiraz South Australia spice subtle syrah syrah tannin tannins texture two value while wine wines with would years you

[wines]	[wines][country]	[wines][location]	[wines][name]	[wines][points]	[wines][price]	[wines][region]	[wines][review]	[wines][review][split]	[wines][reviewer]	[wines][reviewyear]	[wines][varietal]	[wines][vintage]	[wines][winery]
0	argentina	agrelo	Finca Decero: C…	93	20	mendoza	My friend and col…	reviewed , every…	Paul Lukacs	2009	Cabernet Sauvig…	2006	Finca Decero
1	argentina	agrelo	Finca Decero: M…	92	22	mendoza	A new winery fro…	means , intensel…	Paul Lukacs	2009	Cabernet Sauvig…	2006	Finca Decero
2	argentina	agrelo	Finca Decero: "A…	92	50	mendoza	The 2006 vintag…	reviewed , enjoy …	Michael Apstein	2009	Cabernet Sauvig…	2006	Finca Decero
3	argentina	agrelo	Finca Decero: C…	91	22	mendoza	A firm tannic grip…	accessible , wine…	Paul Lukacs	2009	Cabernet Sauvig…	2006	Finca Decero
4	argentina	agrelo	Finca Decero: C…	90	20	mendoza	Look no further f…	complexity , fruit …	Michael Apstein	2009	Cabernet Sauvig…	2006	Finca Decero
1463	chile	Colchagua Valley	Los Vascos: Res…	89	21		A Bordeaux-style…	lush , fruit , herb…	Paul Lukacs	2009	syrah , Cabernet…	2006	Los Vascos

Figure 2.11: Example focused concept tag clouds of original web application ConceptCloud implementation for a reduced size wine review dataset.

The main tag cloud viewer can therefore be used to drive the explorative search process, as further covered in Section 2.3. By updating its focus concept, the main tag cloud viewer will force all the subsequent tag cloud viewer subwindows to update their focus concepts to remain subconcepts of the main tag cloud viewer's focus concept. We see this in Figure 2.12. The main window focus concept has been updated to display the concept defined by the set {`varietal:Cabernet Sauvignon`, `location:australia`}. This in turn forced the three vintage windows to have to include `location:australia` and `varietal:Cabernet Sauvignon` to remain subconcepts of the main window focus concept. That is, as per Definition 6, the sub viewer's focus concepts shown in the windows with stickied vintage: 2004, 2005, 2006, tags, each have a set of currently selected attributes F, each of these sets F, have been updated to include the binary concept `location:australia`, the extent for these F-sets is then changed, and each subconcept is recalculated.


ConceptCloud Browser
«
Navigation:
Home
Open User Guide
Download XML File
Load ConSL Script:
Choose file No file chosen
Load ConSL Script
172 objects in tag cloud
Font Size (10 - 35)
Selected Tags:
✖australia
✖Cabernet Sauvignon
Implied Tags:
2003 2004 2005 2006 2006 2007 2008 88 89 90 91 92 acidity aged alcohol aromas australia australia australian balanced berry best black blackberry blend bodied bright cab cabernet Cabernet Sauvignon cassis character cherry chocolate complex complexity coonawarra dark deep depth eucalyptus excellent fine finish firm flavor flavors french fresh fruit Gerald D. Boyd good great herbal herbs impressive intense margaret merlot merlot Michael Apstein Michael Franz mint new note notes oak over palate Paul Lukacs price red region rich ripe river Robert Whitley sauvignon shiraz shiraz south South Australia spice structure style subtle tannin tannins texture two value Western Australia while wine wines with without would years you
2004
2004 2006 2007 2008 87 89 90 acidity aromas australia australia australian balanced barossa berry best better black blackberry blend bright cabernet Cabernet Sauvignon cassis cedar character chocolate concentration coonawarra dark deep delicious elegant estate even excellent fine finesse finish flavor flavors focused fresh fruit Gerald D. Boyd good great herbal herbs hints intense little margaret Margaret River McLaren Vale merlot merlot Michael Apstein Michael Franz mint new note notes oak palate real red ripe river Robert Whitley sauvignon shiraz shiraz showing smoke soft South Australia southern spice spicy structured style subtle supple sweet tannins vineyards watershed western Western Australia wine winemaker wines with world would years yet you young
2005
2005 2007 2008 86 88 89 90 acidity aromas australia australia balanced berry blackberry blend bodied bottling bright cab cabernet Cabernet Sauvignon cabernets cedar character clearly complexity coonawarra dark deep dried edges flavor flavors fruit Gerald D. Boyd good herbs Howard Park juicy Margaret River Marguerite Thomas merlot Michael Franz mint New South Wales notes oak orange penfolds personality plum plush point price re red reds refined region reserve rich ripe river Robert Whitley rosemount rosemount Rosemount: Cabernet Sauvignon Show Reserve 2005 sauvignon shaw shiraz shiraz smoke soft South Australia South Eastern Australia spice spices spicy structure subtle tannin tannins texture textured two upon varietal vivid weight western Western Australia whiff while wine wines with without wood years you
2006
2006 2008 2009 87 88 91 acidity aged alcohol australia balanced berry black blend cabernet Cabernet Sauvignon cherries concentrated dark fine finish flavors fresh fruit fruits fruity Gerald D. Boyd given good grange grapes grilled herbal herbs hot impressive intensity interesting large leafy little malbec malbec Margaret River McLaren Vale merlot merlot Michael Apstein Michael Franz mint new notes oak over palate penfolds penfolds plenty plums point price range recalling red restrained rich ripe ripeness robust rounds sauvignon shiraz shiraz South Australia spice st steak style subtle sweetness tannin tannins tarry traditional two varietal varieties weight Western Australia when while wide wine wines with Wolf Blass would years yet you

[wines]	[wines][country]	[wines][location]	[wines][name]	[wines][points]	[wines][price]	[wines][region]	[wines][review]	[wines][review][split]	[wines][reviewer]	[wines][reviewyear]	[wines][varietal]	[wines][vintage]	[wines][winery]
777	australia	Langhorne Creek	Bleasdale Vineya…	87	10	South Australia	This is a blend of…	berry , fruit , roas…	Michael Franz	2009	Cabernet Sauvig…	2006	Bleasdale Vineya…
593	australia	Margaret River	Cape Mentelle: S…	90	19	Western Australia	The cooling effec…	means , overripe…	Michael Apstein	2007	Sauvignon Blanc…	2006	Cape Mentelle
551	australia	Margaret River	Chalice Bridge: "…	91	18	Western Australia	This blend of Shi…	berry , passes , h…	Gerald D. Boyd	2008	syrah , Cabernet …	2006	Chalice Bridge
806	australia	McLaren Vale	d'Arenberg: Cab…	91	65	South Australia	Among the most …	traces , coppermi…	Gerald D. Boyd	2008	Cabernet Sauvig…	2006	d'Arenberg
908	australia	South Australia	Jim Barry: Caber…	88	20	South Australia	This wine sports …	sports , fruit , plu…	Michael Franz	2009	Cabernet Sauvig…	2006	Jim Barry

Figure 2.12: Multiple example focused concept tag clouds of original web application ConceptCloud implementation for a reduced size wine review dataset.

2.4 Natural Language Processing

Semi-structured data very often have large free-text components. These components often contain insights into the data we wish to take advantage of, but cannot do with formal means as we lack the specification for what type of information may be contained in the data.

The field of natural language processing provides mechanisms to extract information from such free-text. Maintaining the semantic meaning of the text whilst removing ambiguity are the main goals of the natural language processing we wish to perform.

ConceptCloud makes use of the Stanford CoreNLP library to perform its natural language processing tasks [26]. These include the following:

Tokenization Split the text into words, compound words are broken up into their parts, isn't becomes is n't for example, that is $T(isn't) = \{is, n't\}$. Each token is split based on its context.

Sentence Splitting Break down the input free-text into sentences to be processed.

Named Entity Recognition Annotates known named entities in a body of text.

Key-Phrase Extraction Break text down into separate phrases.

Stemming Reduce derived or inflected words back down to a root or stem form regardless of whether the stem itself is valid.

Lemmatization Group words together to be analysed as a single item, normalises each word to a valid root based on its dictionary form.

```
{
"wine_id": "5731",
"prod_name": "Anura Vineyards",
"brand": "Merlot",
"vintage": "2004",
"wine_label_name": "Anura Vineyards Merlot 2004",
"cap": "Cork",
"margin_rating": 3.5
"vintage_rating_100": "null",
"vintage_rating": 3.5,
"blurb": "Bottled version of sullen sample 04 (***1/2) shows a
leaner, edgier style with sour plum & herbaceous nuances.
Taut tannins make it better suited as a food wine. 15 mths Fr
oak. Pvsly tasted was fragrant 02. This, above, Simonsberg-Paarl
WO.",
"winemaker": "Tymen Bouma & Carla van der Mescht (Jan 2002)",
"viticulturist": "Hannes Kloppers (Oct 1997)",
"additional_info": "Est 1990, 1stBB 2001, Closed Easter Fri/Mon,
Dec 25/26 & Jan 1, Fee R15 (cheese & wine), Lilly Pad Restaurant
(see Eat out section), Tours by appt, Farm Produce sold, 118 ha
(cab, malbec, merlot, mourvèdre, sangiovese, shiraz), 600 tons
total 450 tons / 35000 cs own label 80% red 20% white",
"estate": "0",
"bottling": "1",
"name": "Paarl",
"area": "null",
"details": "null",
"further_remarks": "null",
"blend": "Blended"
},
```

Figure 2.13: Merlot *JSON* object with a free-text wine review / blurb

```
{
 "Blurb keyphrases": "sullen sample , sour plum , sour ,
 plum , herbaceous , herbaceous naunces , sour plum &
 herbaceous nuances , Taut tannin , fragrant , Bottled
 version , leaner , edgier style , food wine ,
 Simonsberg-Paarl WO , Simonsberg , Paarl , WO"
}
```

Figure 2.14: Results of key-phrase extraction performed on the blurb of Figure 2.13

The primary use case for NLP in ConceptCloud is to pre-process free-text attributes within semi-structured data sets. An example of one such free-text attribute is the blurb attribute of the Merlot wine *JSON* object shown in Figure 2.13.

In our example, tokenization would split each item in the blurb into separate tokens or word items. In the example presented in Figure 2.13, the final sentence would be tokenized into the following comma-space delimited sequence: `This , above , Simonsberg Paarl WO`. Note here the hyphenated words are split. This is a configuration choice specifically made for this data set. The sentence splitting component applied to the same example splits on full stops and quoted full stops only. This process would then split all quoted sentences away from the sentence in which they reside. This operation becomes important when determining the subject for each sentence.

Key-phrase extraction of on the blurb yields the following phrases.

It is important to note here the key-phrase extraction will contain phrases that are subphrases of other extracted phrases, this may seem redundant but becomes useful when matching on extracted phrases. This can however lead to inconsistencies or false positives. As with this example, the difference between sour plum and sour lead to very different interpretations of the wine's flavour profile.

I cover this topic and the problems it introduces in both Chapters 3 and 4.

2.5 Description Logic And Ontologies

Ontologies in mathematics and computer science refer to a repository of domain information, most often describing some collection of entities, relations, functions, axioms and instances [24]. Ontologies allow for formal reasoning to be performed over a domain and on data from the domain. This reasoning capability makes them ideal to supplement the existing data in ConceptCloud.

Description Logics

Description Logics (DLs) are a family of knowledge representation formalisms used in the modelling of ontologies. Description Logics vary in expressivity from that of very inexpressive, to near the expressivity of first order logic. Description logics are in fact usually a fragment of first order predicate logic, many being fragments of first order two variable logic, where the description logic is expanded to include counting quantifiers [24].

Description logics are generally focused on decidable fragments, unlike general first order logic, wherein logical inference is undecidable. Decidability is almost regarded as a prerequisite to calling a formalism a description logic [30].

As logical formalisms, description logics are equipped with formal semantics precisely specifying the meanings of description logic ontologies, allowing for logical deduction to infer additional information from statements in an ontology.

In ConceptCloud I wish to take advantage of this inferential capability to turn implicit information contained in the textual attributes within our concept lattice and ontology, into explicit textual attributes within the formal concept lattice. The basis of the implementation relies on the OWL2 API [27], which has a strong relation to the $\mathcal{SROIQ}$ description logic [20].

In this section, I cover the basics of description logics and their relation to OWL, followed by a brief introduction to the HermiT reasoner used in ConceptCloud.

$\mathcal{SROIQ}$ Description Logic Introduction

Description logics provide us with the tools to model the relationships between sets of objects, and between objects within our data sets, or domains of interest.

DLs are languages used to describe concepts, roles and individuals. The meaning of concepts are sets of individuals, roles are the binary relations between these individuals, and individual names are single individuals in the domain.

To avoid confusion the following description logic terms will henceforth be referred to by their OWL syntax: DL concepts will be referred to as *class descriptions*, DL roles will be referred to as *object properties*, when presented in conjunction with their FCA counterparts. Otherwise, the prefix DL or FCA will be used before each term.

We use the description language $\mathcal{SROIQ}$ [20], as it shares many commonalities with OWL2 [27], which is used extensively in the integration of ontology based data imputation into ConceptCloud, as explained in Chapter 4.

The $\mathcal{SROIQ}$ DL concept language consists of the following fundamental components:

- A set $\mathcal{N_I}$ of individual names which correspond to first order logic (FOL) constants, these are all names used to denote individuals in the domain which we describe. In our wine example from Figure 1.1 the winery 'Stark-Condé', or the country 'South Africa' are both examples of DL individual names.

- A set $\mathcal{N_C}$ of DL concept names, these are types, classes or categories of entities in the domain. In the previous wine example, 'winery' and 'country' are both examples of DL concept names (atomic class descriptions).

- A set $\mathcal{N_R}$ of role names (object properties), that denote the relationships between individuals in the domain. An example relationship may be:

  $$\mathsf{(Stark\text{-}Condé,South\ Africa):locatedIn}$$

A $\mathcal{SROIQ}$ ontology consists of sets of axioms expressed in the language of the description logic that are separated into three main groups:

TBox The TBox or Terminological box consists of all axioms describing the relationship between DL concepts (class descriptions).

ABox The ABox or Assertion box consists of all assertions describing named individuals (OWL object instances).

RBox The RBox or Role box consists of all axioms describing the properties of roles and the relations to other roles (OWL object property axioms).

RBox Axioms

A $\mathcal{SROIQ}$ RBox specifies the relationships and interdependencies between DL roles in the knowledge base.

Given a set of $\mathcal{N_R}$ of DL role names, a role is either of the form r or r^- or u, where u is the universal role and r^- denotes an inverse role, that is $Inv(r) := r^-$ and $Inv(r^-) := r$. Where $(x, y) \in Inv(r) \Leftrightarrow (y, x) \in r$

Given a set R of roles, and $r, s \in \mathsf{R}$, a role inclusion axiom (RIA), or role composition axiom, is a statement of the form $r_1 \circ \ldots r_n \sqsubseteq r$ where $r_1 \ldots r_n, r \in \mathsf{R}$. As a special case of role composition, where $n = 1$ we have simple role inclusions $r \sqsubseteq s$.

In our example wine domain, we may wish to express relationships between a wine farm, its location and the wine it makes, as shown in Figure 2.15.

With this example in mind we would have the following:

$$madeBy \circ LocatedIn \sqsubseteq madeAt$$
$$isLocationOf \circ makes \sqsubseteq madeAt^-$$

A finite set of role inclusion axioms is known as a role hierarchy. In a role hierarchy one can distinguish between complex and simple roles, that is to distinguish between roles formed by a role chain or role inclusion axioms when

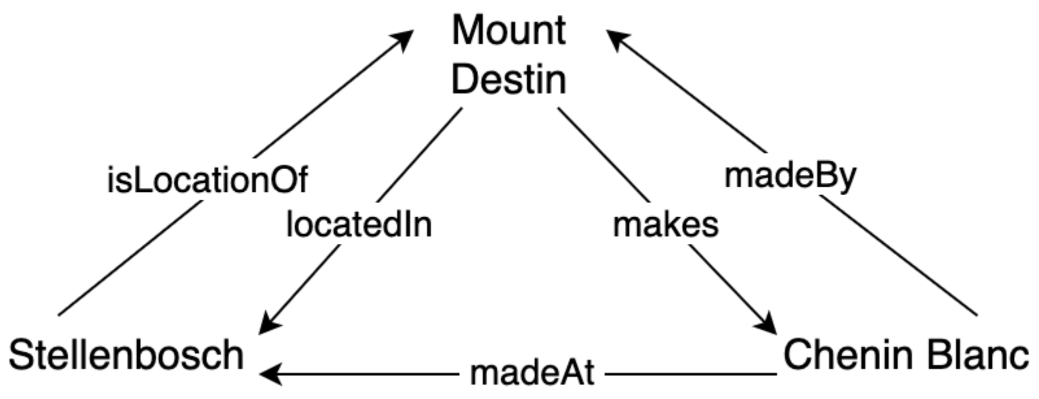


Figure 2.15: Example relationships between a winery, wine and a town based on the ongoing wine example

$n > 1$, and those formed by simple role inclusions. Complex, or non-simple roles are defined as follows:

1. Each role r occurring on the right in a role inclusion axiom $r_1 \circ \ldots r_n \sqsubseteq r$ where $n > 1$ is non-simple.
2. Each role r occurring in a simple role inclusion $s \sqsubseteq r$ with a non-simple s is non-simple.
3. If r is non-simple then its inverse role $Inv(r) := r^-$ is also non-simple.
4. No other role is non-simple.

For a set of non-simple roles R^n of our role hierarchy R, we define all simple roles R^s as the following: $R^s := R \setminus R^n$.

Given our example roles we have the following role hierarchy:

$$\begin{aligned}
\mathsf{isRegionOf} &\sqsubseteq \mathsf{isLocationOf} \\
\mathsf{isWineryOf} &\sqsubseteq \mathsf{isLocationOf} \\
\mathsf{isLocationOf} &\sqsubseteq \mathsf{locatedIn}^- \\
\mathsf{makes} &\sqsubseteq \mathsf{isMadeBy}^- \\
\mathsf{isRegionOf} \circ \mathsf{makes} &\sqsubseteq \mathsf{regionMakes} \\
\mathsf{isWineryOf} \circ \mathsf{makes} &\sqsubseteq \mathsf{wineryMakes} \\
\mathsf{isLocationOf} \circ \mathsf{makes} &\sqsubseteq \mathsf{locationMakes} \\
\mathsf{locatedIn} \circ \mathsf{locatedIn} &\sqsubseteq \mathsf{locatedIn}
\end{aligned}$$

By condition 1 of the definition of non-simple rules we have that the roles, $\mathsf{regionMakes}$, $\mathsf{wineryMakes}$, $\mathsf{locationMakes}$ are non-simple. By 1 we have that the role $\mathsf{locatedIn}$ is non-simple as we have the axiom $\mathsf{locatedIn} \circ \mathsf{locatedIn} \sqsubseteq$

locatedIn, this in turn gives us that the inverse LocatedIn^{-} is non-simple by 3. As none of the other roles expressed in the role hierarchy are built off of these non-simple roles, or their inverses, they must be simple by 2,3 and 4 above.

For $\mathcal{SROIQ}$ to maintain decidability role hierarchies are restricted to regular role hierarchies. A role hierarchy is regular if there exists a strict partial ordering $\prec$ on non-simple roles in R such that $s \prec r$ if and only if $s^{-} \prec r$, and each role inclusion axiom is of the following form:

1. $r \circ r \sqsubseteq r$
2. $Inv(r) \sqsubseteq r$
3. $s_1 \circ \ldots s_n \sqsubseteq r$
4. $r \circ s_1 \circ \ldots s_n \sqsubseteq r$
5. $s_1 \circ \ldots s_n \circ r \sqsubseteq r$

Here $r \in N_R$ is a non-inverse role name r, and $s_i \prec r$ for $i = 1 \ldots n$ when s_i is non-simple.

A $\mathcal{SROIQ}$ RBox is a union of a DL role-hierarchy (OWL object property hierarchy) and a finite set of DL role characteristics. A role characteristic is a statement of the form:

1. $Ref(r)$ role r is reflexive
2. $Dis(s, s')$ roles s, s' are disjoint
3. $Asy(s)$ role s is asymmetric

Where s, s' are simple roles and r may be simple or non-simple. A $\mathcal{SROIQ}$ RBox is regular if its role hierarchy is regular.

TBox Axioms Given a $\mathcal{SROIQ}$ RBox, $\mathcal{R}$, DL-concepts are defined as follows:

1. Each concept name $C \in \mathcal{N}_\mathcal{C}$ is itself a concept
2. $\top$ (Top) is a concept, used to quantify all objects in our domain.
3. $\bot$ (Bottom), is a concept of which no object in the domain may be an instance
4. Each Individual name $Ind_1 \ldots Ind_n \in \mathcal{N}_I$ is itself a concept known as a nominal concept
5. If A and B are concepts then so too is their intersection or conjuction $A \sqcap D$
6. If A and B are concepts then so too is their union or disjunction $A \sqcup D$
7. If A is a concept then so too is its negation $\neg A$
8. For a role r and a concept A the existential quantification $\exists r.C$ is a concept

9. For a role r and a concept A the universal quantification $\forall r.C$ is a concept
10. For a simple role r, a natural number n, and a concept A we have the following concepts:

- $\exists r.Self$ self restriction
- $\leq n\ r.C$ At-least restriction
- $\geq n\ r.C$ At-most restriction

At-least and at-most restrictions are known as cardinality restrictions or qualified number restrictions.

Let C be the set of all DL concepts, then general concept inclusion axioms (GCIs) are axioms of the form $A \sqsubseteq B$ for DL concepts $A, B \in \mathsf{C}$. GCIs are often referred to as subsumption or is-a relations, that is, in the GCI $A \sqsubseteq B$ A is subsumed by B or A is-a B. These relations may be chained to form a subsumption hierarchy. This links back to the thesaurus notion in Figure 1.5, where we have $\mathsf{Chenin\ Blanc} \sqsubseteq WhiteWine, WhiteWine \sqsubseteq Wine, Wine \sqsubseteq Thing$. That is, a Chenin Blanc is a White Wine, which is a Wine which is a Thing.

A $\mathcal{SROIQ}$ TBox is a finite set of concept inclusion axioms.

ABox An ABox or assertion box is made up of individual assertions, statements which pertain to named individuals, unlike the TBox which may pertain to all individuals.

For individuals $i, j \in \mathcal{N_I}$, C $\in \mathcal{N}_C$, $r \in \mathcal{N}_R$, individual assertions have the following forms:

1. $C(i)$ is a concept assertion, i is an instance of concept C
2. $r(i, j)$ is a role assertion, i is linked to j by role r.
3. $\neg r(i, j)$ negated role assertion
4. $i \approx j$ individual equality i is j, that is the names for i and j correspond to the same individual
5. $i \not\approx j$ individual inequality

A $\mathcal{SROIQ}$ ABox is a finite set of individual assertions. We call it existentially reduced if it only contains roles and concepts that correspond to role and concept names respectively. That is each concept $C \in \mathcal{N_C}$ and each role $r \in \mathcal{N_R}$.

A $\mathcal{SROIQ}$ knowledge base $\mathcal{KB}$ is the union of a regular Rbox $\mathcal{R}$, a TBox $\mathcal{T}$ and an ABox for $\mathcal{R}$, $\mathcal{A}$. Elements of the $\mathcal{KB}$ are referred to as axioms. We denote the individual names, concept names and role names of the $\mathcal{KB}$ by $\mathcal{N}_C(\mathcal{KB})$, $\mathcal{N}_I(\mathcal{KB})$ and $\mathcal{N}_R(\mathcal{KB})$ respectively.

For the sake of illustration I will develop a simple Wine Ontology as we continue. The following will be some foundational axioms for it.

TBox Axioms

$\mathcal{T} = \{$

1. $\mathsf{RedWine} \equiv \mathsf{Wine} \sqcap \exists\ \mathsf{hasColour}.\{\mathsf{red}\}$
2. $\mathsf{WhiteWine} \equiv \mathsf{Wine} \sqcap \exists\ \mathsf{hasColour}.\{\mathsf{white}\}$
3. $\mathsf{RoseWine} \equiv \mathsf{Wine} \sqcap \exists\ \mathsf{hasColour}.\{\mathsf{rose}\}$
4. $\mathsf{RedWine} \sqcup \mathsf{WhiteWine} \sqcup \mathsf{RoseWine} \sqsubseteq \mathsf{Wine}$
5. $\mathsf{Winery} \sqsubseteq \exists\ \mathsf{locatedIn}.\mathsf{Location}$
6. $\mathsf{Location} \sqsubseteq \exists\ \mathsf{locatedIn}.\mathsf{Region}$
7. $\mathsf{Region} \sqsubseteq \exists\ \mathsf{locatedIn}.\mathsf{Country}$
8. $\mathsf{Winery} \sqcup \mathsf{Region} \sqcup \mathsf{Country} \sqcup \mathsf{Location} \sqsubseteq \mathsf{Place}$
9. $\mathsf{Wine} \sqcap \exists \mathsf{locatedIn}.\{\mathsf{SouthAfrica}\} \sqsubseteq \mathsf{SouthAfricanWine}$
10. $\mathsf{SouthAfricanWine} \sqsubseteq \exists\ \mathsf{locationMakes}.\{\mathsf{SouthAfrica}\} \sqcap \mathsf{Wine}$
11. $\top \sqsubseteq \mathsf{Wine} \sqcup \mathsf{Colour} \sqcup \mathsf{Place}$

$\}$

Our first 3 axioms state that the concepts named RedWine, WhiteWine and RoseWine all correspond to Wine concepts which have the role hasColour with the instance of a corresponding Colour concept. Our fourth axiom states that RedWines, WhiteWines and RoseWines are all subconcepts of a Wine, we can read this as a RedWine is a Wine, a WhiteWine is a Wine etc.

The fifth to seventh axioms state that a Winery is a located in an existing location Location, a location is located in an existing Region and a Region is located in an existing Country respectively. The ninth axiom states that wineries, regions, countries and locations are all places. Axiom 10 states that a wine located in the individual SouthAfrica is a South African wine. The eleventh axiom states that a SouthAfricanWine is a Wine that is made at an existing location in SouthAfrica. The final axiom states that everything in our ontology is either a Wine, a Colour or a Place.

ABox Axioms

$\mathcal{A} = \{$

1. petersMerlot:RedWine
2. samsSauvignonBlanc:WhiteWine
3. rosiesRose:RoseWine
4. mattsMerlot:RedWine
5. South_Africa:Country
6. $\mathsf{mattsMerlot} \approx \mathsf{petersMerlot}$
7. white:Colour, red:Colour, rose:Colour
8. $\mathsf{red} \not\approx \mathsf{white}$, $\mathsf{white} \not\approx \mathsf{rose}$, $\mathsf{rose} \not\approx \mathsf{red}$

}

Our first four ABox statements are concept assertions, which assert that the individuals, named in brackets are instances of the preceding concepts, i.e Peter's Merlot is a red wine. The sixth axiom states that the named individuals, 'mattsMerlot' and 'petersMerlot' refer to the same individual. We can think of this as Matt and Peter having the same Merlot wine. Our seventh axiom states that red, white, rose are all instances of the Colour concept. Our final axiom states that the Colour individuals red, white and rose are disjoint.

RBox Axioms Given our previous role hierarchy we may have the following RBox:

$\mathcal{R} = \{$

1. $\mathsf{WineryMakes}^{-} \equiv \mathsf{madeAtWinery}$
2. $\mathsf{makes} \equiv \mathsf{isMadeBy}^{-}$

3.$\mathsf{locatedIn} \circ \mathsf{locatedIn} \sqsubseteq \mathsf{locatedIn}$

$\}$

The first role axiom states that the inverse of a wine being made a winery is a winery making a wine. The third role axiom states that the role locatedIn is transitive. It is important to note that our ontology usually does not fully specify a particular situation it is meant to describe, our intended meaning of the ontology is only derivable from it's ontological axiom, and not it's identifier names. In our above example, even though we have Matt's Merlot as an individual wine, there is no guarantee an individual Matt does (or does not) exist.

Description logic semantics generally consider all possible situations wherein our ontological axioms hold, we keep unspecified information open, this is known as the open-world assumption.

Description Logic Semantics

Formal Description Logic semantics revolves around the central idea of providing a semantic counterpart for consequence relation which determines whether a particular axiom follows, or is entailed by, a set of axioms. Our notion of semantics is defined in a model-theoretic way, that is by defining the set theoretic notion of an interpretation of the axioms in an ontology.

Interpretation

Formally, an interpretation $\mathcal{I}$ provides:

- A domain, a non empty set $\Delta^{\mathcal{I}}$ representing the entirety of all individuals or things in the universe about which the axioms are being expressed.

- An interpretation function $\cdot^{\mathcal{I}}$ which provides a mapping between the domain and the axioms by providing:

  A corresponding individual in the interpretation for each individual a in the ontology, $\forall a \in \mathcal{N}_i : \cdot(a) = a^{\mathcal{I}} : a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$

  A corresponding set of domain elements in the interpretation for each concept name in the ontology. That is $\forall A \in N_C : \cdot(A) = A^{\mathcal{I}} : A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$

  A corresponding set of ordered pairs of domain elements for each role name in the ontology. $\forall r \in N_R : \cdot(r) = r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

DL Complex Concept and Role semantics

Recall that a complex concept $\mathcal{C}$ was defined as a concept built up from one or more atomic concepts, eg., $\mathcal{C} : A \sqcap B$, $\mathcal{C} := \neg A$ or $\mathcal{C} := \exists r.A$.

An example of this is for the complex concept RedWine, where we have built a new concept from the atomic concept wine, and the atomic concept defined by all colour objects having the instance value red as colour. That is RedWine $\equiv$ Wine $\sqcap \exists$hasColour.$\{$red$\}$.

In Figure 2.16, we present an extended table of $\mathcal{SROIQ}$ semantics and syntax.
As the interpretation $\mathcal{I}$ fixes the meanings of all the entities in the ontology we can say with certainty whether each axiom in the ontology holds in the interpretation $\mathcal{I}$. If an axiom α holds in $\mathcal{I}$ we say that α is satisfied in $\mathcal{I}$ or $\mathcal{I}$ models α that is $\mathcal{I} \models \alpha$.

More specifically an axiom holds in $\mathcal{I}$ if the following is true [30]:

Definition 8.

1. *A role inclusion or role composition axiom* RAxiom*:=* $r_1 \circ \dots r_n \sqsubseteq r$*,* RAxiom *is true in the interpretation* $\mathcal{I}$ *if it has a direct* r *path that traverses these roles* $r_1, \dots r_n$ *in order, that is* $r_1^{\mathcal{I}} \circ \dots \circ r_n^{\mathcal{I}} \subseteq r^{I}$*.*
   RAxiom *holds true in* $\mathcal{I}$ *if for all* $\delta_1 \dots \delta_n \in \Delta^{\mathcal{I}}$ *where*
   $\langle \delta_1, \delta_2 \rangle \in r^{\mathcal{I}}, \dots, \langle \delta_{n-1}, \delta_n \rangle \in r_n^{\mathcal{I}}$ *holds, that* $\langle \delta_1, \delta_2 \rangle \in r^{\mathcal{I}}$ *is also satisfied.*

2. *A subsumption axiom* $A \sqsubseteq B$ *is satisfied by an interpretation* $\mathcal{I}$ *if every instance of* A *is also an instance of* B*, that is* $A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$*.*

3. *For concept assertion axiom* C(a) *to hold true in* $\mathcal{I}$ *each individual with the name* a *must be an instance of* C*, that is that* $a^{\mathcal{I}} \in C^{\mathcal{I}}$*.*

	Syntax	Semantics
Atomic Concept	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
Concept Negation	$\neg A$	$\Delta^{\mathcal{I}} \setminus A^{\mathcal{I}}$
Bottom Concept	$\bot$	$\emptyset$
Top Concept	$\top$	$\Delta^{\mathcal{I}}$
Concept Disjunction	$A \sqcup B$	$A^{\mathcal{I}} \cup B^{\mathcal{I}}$
Concept Conjunction	$A \sqcap B$	$A^{\mathcal{I}} \cap B^{\mathcal{I}}$
Universal Quantification	$\forall R.A$	$\{x \Vert \forall y(x,y) \notin R^{\mathcal{I}}$ or $y \in A^{\mathcal{I}}\}$
Existential Quantification	$\exists R.A$	$\{x \Vert \exists y(x,y) \in R^{\mathcal{I}}$ and $y \in A^{\mathcal{I}}\}$
Atomic Role	R_A	$R_A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
Inverse Role	R^-	$\{(y,x) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \Vert (x,y) \in R^{\mathcal{I}}\}$
Universal Role	U	$\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
Nominals	$\{n_1, \ldots, n_m\}$	$\{n_1^{\mathcal{I}}, \ldots, n_m^{\mathcal{I}}\}$
At least restriction	$\geq mS.A$	$\{x \Vert \#\{y : (x,y) \in S^{\mathcal{I}}$ and $y \in A^{\mathcal{I}}\} \geq m$
At most restriction	$\leq mS.A$	$\{x \Vert \#\{y : (x,y) \in S^{\mathcal{I}}$ and $y \in A^{\mathcal{I}}\} \leq m$
Local Reflexivity	$\exists S.Self$	$\{x \Vert (x,x) \in R^{\mathcal{I}}\}$

Figure 2.16: $\mathcal{SROIQ}$ syntax and semantics

4. *A role assertion axiom* r(a,b)*, holds true in an interpretation* $\mathcal{I}$ *if the individuals* a *and* b *are connected in* $\mathcal{I}$ *by* r*, that is* $\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \in r^{I}$*.*

5. *A role disjointness axiom* Dis(r_1, r_2) *holds true in an interpretation* $\mathcal{I}$ *if every pair of domain individuals* $\delta_1, \delta_2 \in \Delta^{\mathcal{I}}$ *which are connected by* r_1 *are not connected by* r_2*, that is that the roles* r_1, r_2 *are mutually exclusive and* $r_1^{\mathcal{I}} \cap r_2^{\mathcal{I}} = \emptyset$*.*

6. *And equality statement axiom* a $\approx$ b *holds true in an interpretation* $\mathcal{I}$ *if the individuals* a,b *refer to the same individual in the domain,* $a^{\mathcal{I}} = b^{\mathcal{I}}$*.*

7. $\mathcal{I}$ *is a model of* $\neg r$(a,b) *if it is not a model of* r(a,b)*.*

8. $\mathcal{I}$ *is a model of* a $\not\approx$ b *if it not a model of* a $\approx$ b*.*

For an ontology $\mathcal{O}$ if each axiom $a \in \mathcal{O}$ $\mathcal{I} \models \alpha$ then $\mathcal{I} \models \mathcal{O}$, then $\mathcal{I}$ is a model of $\mathcal{O}$. We say that $\mathcal{O}$ is consistent if it has at least one model.

We say that α is a consequence of $\mathcal{O}$ if it holds in every model of $\mathcal{O}$. That is $\mathcal{O}$ entails α, $\mathcal{O} \models \alpha$.

An axiom that holds in all possible interpretations or situations that satisfy the ontology, is referred to as a logical consequence.

As we add axioms to the ontology we tighten the constraints on any interpretations of the ontology, leading to fewer models, the converse holds true, where there are fewer models of an ontology there are more axioms that hold in all models, and more logical consequences follow from the ontology.

We represent our interpretation of the axioms and domain outlined previously in Figure 2.17.

Given the above interpretation, we see that our TBox axiom, 4. RedWine $\sqcup$ WhiteWine $\sqcup$ RoseWine $\sqsubseteq$ Wine, holds, each wine concept is a subconcept of the wine concept. We see that as in our ABox we have that each of, Peter's Merlot, Matt's Merlot, Rosies Rose are red and rose wines respectively. We have that South Africa is a country and our colours are all distinct individuals.

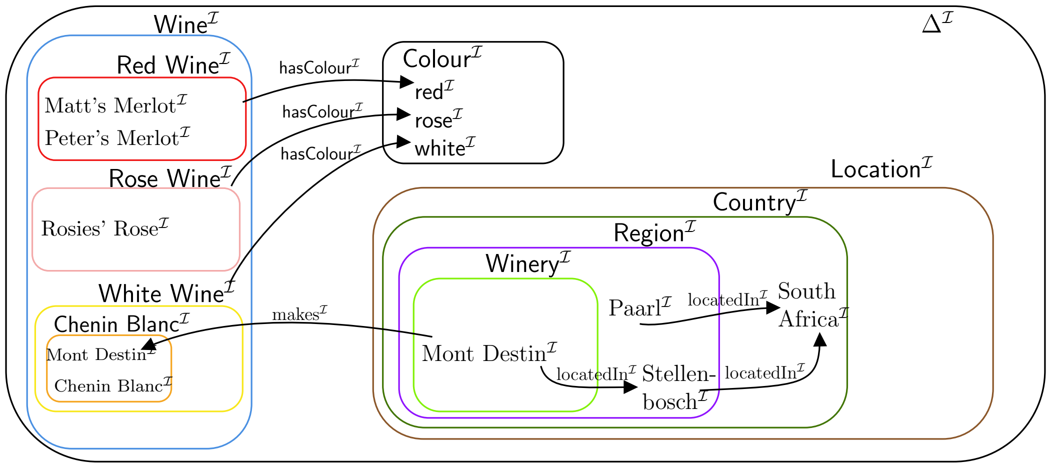

$\text{Wine}^{\mathcal{I}}$
$\Delta^{\mathcal{I}}$
$\text{Red Wine}^{\mathcal{I}}$
$\text{Matt's Merlot}^{\mathcal{I}}$
$\text{Peter's Merlot}^{\mathcal{I}}$
$\text{hasColour}^{\mathcal{I}}$
$\text{hasColour}^{\mathcal{I}}$
$\text{hasColour}^{\mathcal{I}}$
$\text{Colour}^{\mathcal{I}}$
$\text{red}^{\mathcal{I}}$
$\text{rose}^{\mathcal{I}}$
$\text{white}^{\mathcal{I}}$
$\text{Rose Wine}^{\mathcal{I}}$
$\text{Rosies' Rose}^{\mathcal{I}}$
$\text{Location}^{\mathcal{I}}$
$\text{Country}^{\mathcal{I}}$
$\text{Region}^{\mathcal{I}}$
$\text{Winery}^{\mathcal{I}}$
$\text{White Wine}^{\mathcal{I}}$
$\text{Chenin Blanc}^{\mathcal{I}}$
$\text{makes}^{\mathcal{I}}$
$\text{Mont Destin}^{\mathcal{I}}$
$\text{Chenin Blanc}^{\mathcal{I}}$
$\text{Paarl}^{\mathcal{I}}$
$\text{locatedIn}^{\mathcal{I}}$
$\text{South Africa}^{\mathcal{I}}$
$\text{Mont Destin}^{\mathcal{I}}$
$\text{locatedIn}^{\mathcal{I}}$
$\text{Stellenbosch}^{\mathcal{I}}$
$\text{locatedIn}^{\mathcal{I}}$

Figure 2.17: Possible interpretation $\mathcal{I}$ of the domain information presented in Chapter 2.

$\mathcal{SROIQ}$ and OWL

OWL follows closely to the description logic $\mathcal{SROIQ}$ for which entailment of axioms is decidable, with structural restrictions, as shown above in Figure 2.16.

$\mathcal{SROIQ}$ is an extension of $\mathcal{ALC}$ [24] adding in transitive roles ($\mathcal{S}$), complex role axioms ($\mathcal{R}$), nominals ($\mathcal{O}$), inverse roles ($\mathcal{I}$), and qualified number restrictions ($\mathcal{Q}$) [20].

OWL is a knowledge representation language formalised by W3C, I make use of an OWL 2 based API to interact with our ontology. We require that our ontology's axioms are able to be expressed in $\mathcal{SROIQ}$, ensuring that our entailment is decidable and our reasoning terminates.

If in our original wine example we had Rose $\equiv$ White$\sqcap$Red we could express the same in OWL as EquivalentClasses(Rose, ObjectIntersectionOf(White,Red)).

In most cases, there are corresponding OWL operator names for each description logic syntax component.

In Chapter 4 we describe an algorithm for the enrichment of semi-structured datasets focussed around mapping row items from the context table as described in Section 2.1 to OWL individuals, with a corresponding mapping between FCA attributes and OWL object property assertion axioms.

HermiT

In Section 2.5, we discussed semantics and semantic reasoning, and how we would link axioms together to infer new information from our ontology.

Automatic semantic reasoners have existed for many years. Informally semantic reasoners are software engines that are able to read in sets of axioms and infer logical consequences from them.

When selecting a reasoner to integrate into ConceptCloud the chief deciding factors were to find one which will be easy to integrate, that is, is implemented in Java, the same language as ConceptCloud and conforms to the OWL API. If possible, it should too be able to process multiple different ontologies of varying expressivity, to allow ConceptCloud to process many differing ontologies to supplement the existing input semi-structured data.

The HermiT reasoner conforms to each of these requirements, it implements the *OWL2 java API* and has been shown to be able to process many ontologies of varying expressivity [32]. HermiT is one of the few reasoners that is able to perform both object and data property classification—reasoning tasks, which are required by ConceptCloud, as explained in Chapter 4, Section 4.4.

2.6 Summary

In this chapter, I covered a basic introduction to the basis of concepts used in this thesis. Each section focussed mainly on establishing the theory required to understand the techniques used in the development of the ConceptCloud tool.

Section 2.1 covers an introduction to Formal Concept analysis and its' use in establishing the formal concept lattice driven explorative search process used in ConceptCloud. This explorative search process provides the core functionality for ConceptCloud, enabling users to explore their data sets without prior knowledge of the internal structure of the data, while still allowing the user to gain knowledge contained in the underlying structure.

Section 2.2 covered related work in the field of data exploration tools, as well as introducing the ConceptCloud tool. The tools, ConExp, FCArt and ConceptCloud are presented, and the strengths and shortcomings of each were given. The shortcomings of ConceptCloud and their resolutions form the basis of this thesis.

Section 2.4 covers the natural language processing techniques used in ConceptCloud; these are further expanded on in Chapter 4.

We then cover an introduction to description logics and their applications in Section 2.5. I introduce the basics of description logic syntax and semantics and lead into an introduction to OWL, which forms the basis of our technical implementation of formal reasoning into ConceptCloud. The extended implementation detail is covered in Chapter 4.

Chapter 3

Server Based Architecture

As discussed previously, semi-structured datasets have become progressively larger, while incorporating more and more domain-specific information. An example of this is the rise of geolocation rich data.

Thus arises the problem of ConceptCloud needing both to scale to accommodate these larger datasets and provide support for specialised visualisations of domain specific data.

This chapter discusses the re-factorisation of the initial ConceptCloud code base and additions, improvements and expansions made to the software.

3.1 Overview

ConceptCloud began as an interactive browser-based tool for *Git* and *SVN* repositories [16, 18]. ConceptCloud 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 [10]) has shown the limitations of its original client-based architecture. I 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 chapter, I describe the new architecture and interface and show that it yields 10x performance improvements. Specifically, I present the original architecture of the ConceptCloud System, the limitations of the original implementation, changes made, and preliminary experimental results over a wine review data set.

Shift In Application Domain

The initial implementation of the ConceptCloud system was geared towards exploration of the metadata of software repositories [17, 18]. 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 data sets [10], 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.

Further use of the ConceptCloud tool drove it toward being used on ever-growing semi-structured datasets, such as academic paper libraries, academic conference data, and finally to the domain I am mostly working on, wine review information. As such, the ConceptCloud tool needed a shift in its architecture to better accommodate the larger data sets, as the previous versions, in which the entire data set was represented on the client side in the tag-cloud windows, was becoming far too memory intensive and no longer practically feasible for the larger, and in some senses more general data sets. The first change in this architecture was done by Gillian Greene for the *ACM* paper library data set [10]. The main shift was moving to have only a limited subset of the tags displayed in the client's front end window and search functionality assisted by a database cache for access to the non-displayed items in the data set. In this way, the user is still able to navigate through the larger overall lattice with an explorative search as before, but the user now does so by viewing a much smaller subset of the data in the front end.

This version of ConceptCloud was explicitly written for the *ACM* publication database. At a high level, this approach was generalised and then the general implementation was used for the wine data domain.

Description of the Generalised ConceptCloud Architecture

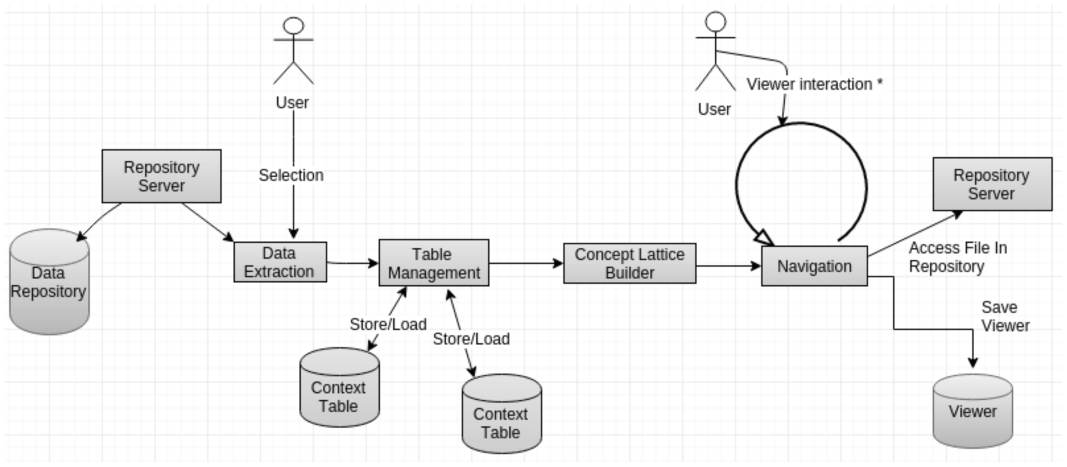


Figure 3.1: Initial ConceptCloud System architecture as implemented by Gillian Greene in "Interactive tag cloud visual-ization of software version control repositories"[18].

The original ConceptCloud architecture was as in Figure 3.1.

Data Extraction Responsible for the retrieval of data from the data repository. This component stems from the original ConceptCloud implementation, which was focused on the analysis of software repositories, this component would fetch and process the data from a software repository. It was later adapted to accept data in the form of semi-structured data files. To accomplish data extraction, the user must provide subsystem with some specifications for the extracted data, most commonly a document type definition or *DTD* file alongside the semi-structured input data.

Table Management Responsible for the creation of a context table from the extracted data, as well as loading an existing context table either from the file-system or the data extracted from the repository.

Concept Lattice Builder Responsible for the creation of a concept lattice from the Context table.

Navigation Responsible for the creation of Tag-Cloud viewers. These are the Tag Clouds through which the user interacts with the underlying concept lattice and exploratively searches through the dataset.

As the initial ConceptCloud system was stable and achieved its goals, it made sense to retain as much of the existing architecture and implementation as possible. However, it was still necessary to update some of the components to work with the different underlying architecture goals, more significant separation of the data in the client, allowing for larger data sets to more easily be worked with, as well as separating the data stored in the live application to memory and into a SQL database, which allowed for a clearer caching strategy and more responsiveness in the search functionality.

Initial Maintenance

In 2017 it was announced that the primary build tool of the Play! framework, the Typesafe Activator Project, was being deprecated and moved to maintenance mode. Following this, the Play! framework project moved away from using the Activator Build Tool, as in its earlier versions, to using Simple Build Tool, SBT. This alongside the updates in the Play! framework from version 2.2, to 2.6 required a substantial refactor of the ConceptCloud codebase.

This included changes to support Google Guice dependency injection. This required the shift of many of the older static Java classes to a dynamic object orientated model, which caused a knock-on effect of classes needing to be refactored and updated. Additionally, many of the dependencies had to be revised and updated where possible.

3.2 Scalable ConceptCloud Architecture

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 2.3. For this to function correctly, a caching structure and separation of the data in memory was required. For this, I implemented a *PostgreSQL* database, in which the database tables are generated based on 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. This allows for generally responsive tag cloud creation and rendering.

The architectural changes presented are a generalisation of the architecture used in [10], a highly specialised version of ConceptCloud used to visualise the the *ACM* Digital Library. This dataset, at the time of writing, has over 4 million records. The scalable ConceptCloud Architecture presented is not specialised to any specific dataset and may be used for any well formed *JSON* dataset, or *CSV* 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.

The largest architecture change is that the system takes a pre-input dataset in the form of a properly formatted *JSON* file, or *CSV* file for very large inputs. This differs from the original architecture in that each client using the application will be looking at the same dataset, as apposed to each client uploading a different smaller dataset to work with each time they access the server. First time execution of the new ConceptCloud system will perform key phrase extraction on fields in the dataset specified by the configuration file, this is only the very first execution from the very first client, on this initial run the system will also generate and populate the required postgres databases and database tables.

Note the changes to the data extraction modules, shown in Figures 3.1 and 3.2, the data is purely encapsulated in a properly formed *JSON* or *CSV* file. The Data extraction modules then use the file to generate the caching databases for the system. Additionally the context table is built in memory, page by page based on the currently focused concept, and not output to a file.

The *CSV* approach allows for batched database insertion and has a far smaller memory footprint. This is as for very large *JSON* input the validators have to open the entire file into memory, to ensure the syntax is correct and the braces are all matched. For a *CSV* file only the header and the current line of input need to be held in memory for validation, allowing for batched input into the database and a far lower memory footprint in the data validation phase.

Description of Updated System Architecture

In conjunction with the descriptions given above an adjusted architecture diagram showing the component interaction for a normal project execution is presented in Figure 3.2.

As reflected in Figure 3.2, the main architectural changes are the changing of the data-source from a software repository to a *JSON* or *CSV* semi-structured input, the addition of the project execution scripts and the caching database.

Fundamentally the functionality of the ConceptCloud tool remain the same, the user selects a semi-structured data source, uploads it to ConceptCloud. The uploaded data is then extracted and the initial lattice and context tables are prepared, the tag clouds are presented to the user. The user then enters the navigation loop.

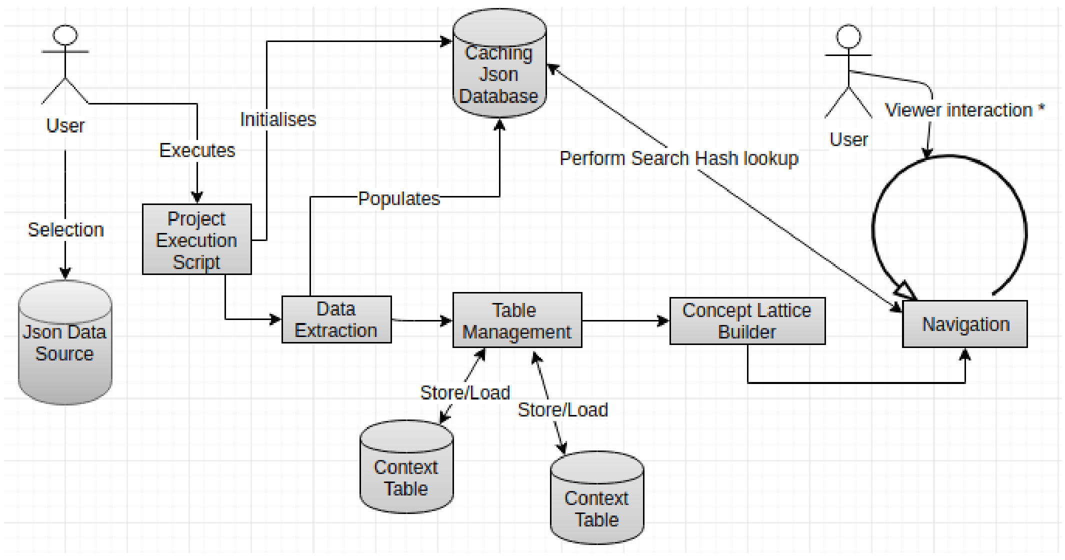


Figure 3.2: Updated ConceptCloud system architecture including the addition of a fixed data source and caching database.

Data Source The user uploaded data source, either a *JSON* or *CSV* data source. Now a single data source for all clients, as apposed to earlier ConceptCloud implementations that had multiple data-sources.

Project Execution Script The project execution script houses the initial first run configuration for the project, the location of the data source (if a *CSV* datasource is selected, batch input will always be performed), the fields in which key-phrase extraction should be performed and the columns to be used in the displayed context table.

Data Extraction Responsible for the retrieval of data from the Data repository. The extraction system now gets all of the structure required to generate the caching database and extract the data from the input data itself. Additionally this component performs the keyphrase extraction and any require NLP operations on the data before passing the data through to the table management and lattice builder operations.

Table Management Responsible for the creation of a context table from the extracted data, as well as loading an existing context table from the file-system. The context table displayed is limited to the context for the context table objects relevant to the 5000 displayed tags.

Concept Lattice Builder Responsible for the creation of a concept lattice from the Context table. Only generates the relevant subsections of the lattice as required by the navigation parameters determined by the navigation component.

Navigation Responsible for the creation of the Tag-Cloud viewers. These are the Tag Clouds through which the user interacts with the underlying concept lattice and exploratively searches through the dataset. The tags are now limited to the top 5000 results related to the user selected tags.

Scalable Navigation Architecture

The core idea of the scalable architecture was to limit the cost of displaying all the tags within the dataset, while still providing access to all of the data where required.

I imposed a tag limit of 5000 displayed tags. This is then at worst the linear cost of the creation and display of 5000 tags, no matter the size of the underlying dataset. To access the underlying dataset an autocomplete search function was added. This auto completes the input and suggests the tag's category in which to search, this adds further granularity to the search results and narrows the size of the search result. In turn this lowers the cost of displaying the result and allows us to lower the frequency of having to select the top 5000 tags from result sets greater then 5000.

For example, in a free text rich dataset, such as that of the wine reviews dataset, further covered in Section 3.6, when key phrase extraction is performed often the extracted phrases may be identical to the values of a particular attribute category, for example "Merlot" is both a varietal and a phrase, it becomes useful to segment the two, as in Figure 3.3. This in turn allows for each search result to be logically equivalent to a tag, which allows for a natural integration with the existing navigation architecture.

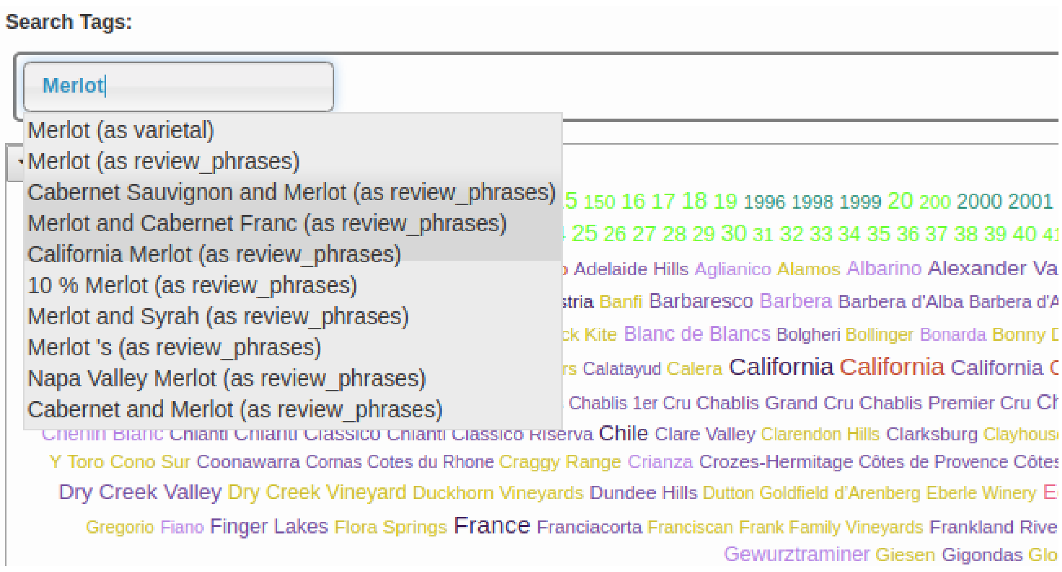

Search Tags:
Merlot
Merlot (as varietal)
Merlot (as review_phrases)
Cabernet Sauvignon and Merlot (as review_phrases)
Merlot and Cabernet Franc (as review_phrases)
California Merlot (as review_phrases)
10 % Merlot (as review_phrases)
Merlot and Syrah (as review_phrases)
Merlot 's (as review_phrases)
Napa Valley Merlot (as review_phrases)
Cabernet and Merlot (as review_phrases)

Figure 3.3: ConceptCloud Auto-complete functionality for 'Merlot' search input

By making the search results equivalent to a tag I can allow the selection of a search result and the selection of a tag to be identical in their resulting output. This equivalence is displayed in the updated navigation architecture in Figure 3.4. The user selecting a tag from search or directly from the tag cloud leads to the same event loop. The focus concept of the main tag cloud viewer is recalculated and updated, then each of the subviewers have their tag clouds recalculated. This is done by recalculating each sub viewer's focus concept, and then calculating the subsections of the lattice to which each focus concept corresponds. These lattice subsections are then used to update the existing tag cloud values. This data then goes through the viewer builder which generates each tag cloud for display and exploration.

The user can then search, or interact with, the resulting tags, which would cause the system to step through the event loop again as described above.

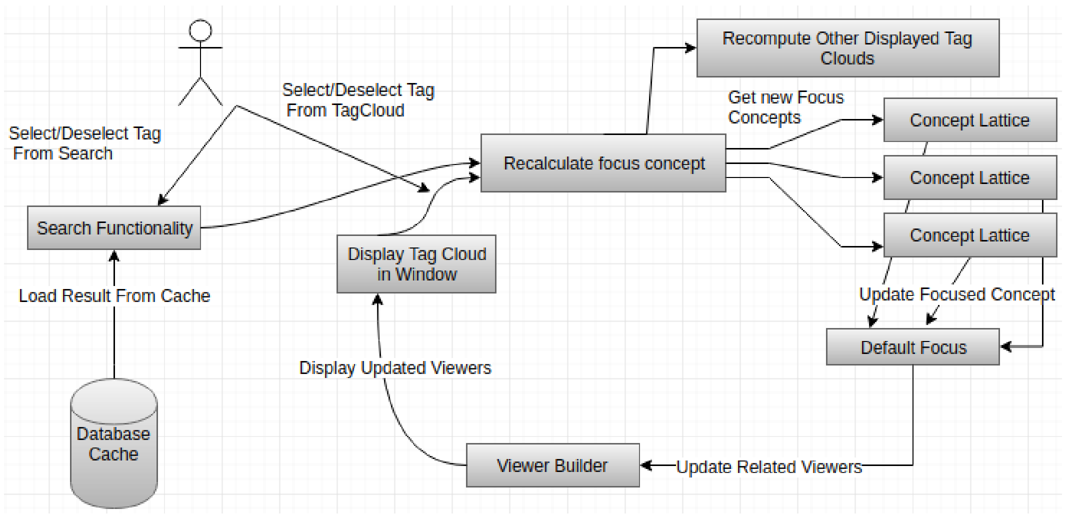


Figure 3.4: Updated navigation component flow diagram showing search functionality and Database cache interaction with a ConceptCloud user

3.3 System Extensibility

The updated ConceptCloud system was written in such a way as to be as generic as possible, in this way it should be relatively simple to swap in different datasets. This is achieved by having the system generate as many of the resources for the dataset to be used as possible. This includes the java ebean [6] model for the caching database.

Extented API

The external facing ConceptCloud API allows the developer to de-couple the explorative search process as described in Section 2.3 from the ConceptCloud API. This is achieved by providing functionality that provides the user with the data traditionally represented within a tag cloud, as well as providing the data corresponding to the process of selecting and de-selecting tags.

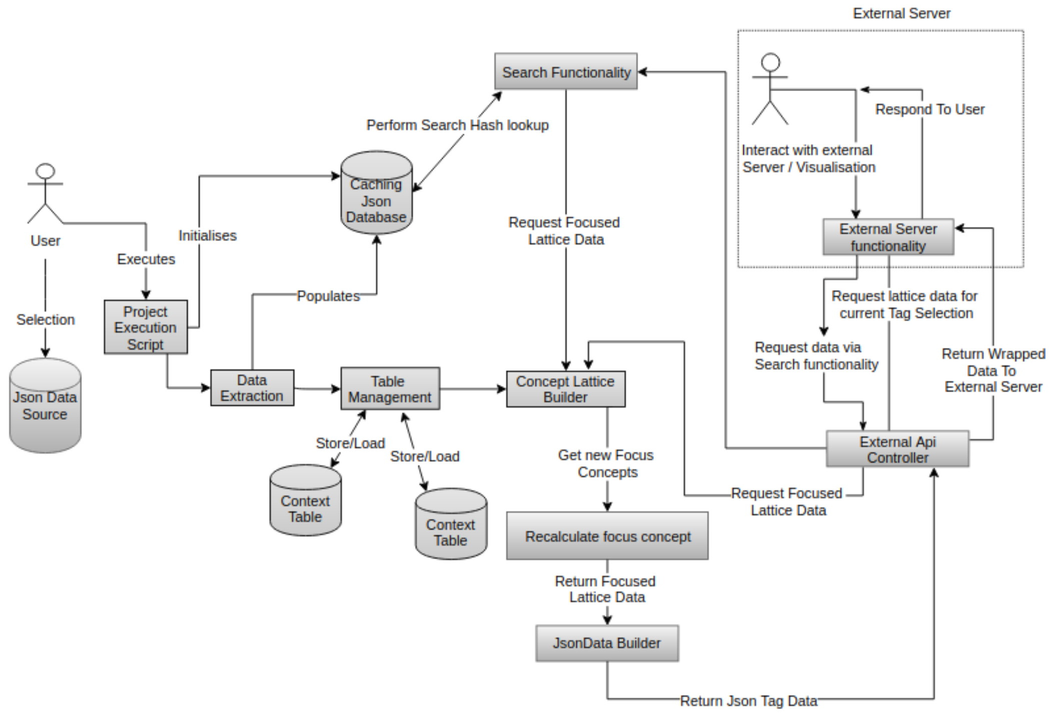


Figure 3.5: ConceptCloud external API architecture showing the external API controller's interaction with the rest of ConceptCloud's components.

Figure 3.5 shows the updated architecture for the external facing API, it shares most of the components with the architecture shown in the updated ConceptCloud architecture shown in Figure 3.2. The main functional difference is that instead of the lattice being provided to a front end tag cloud builder, it is translated into *JSON* format and wrapped in *CORS* headers and sent to the requesting server. The program flow for navigation, as seen in Figure 3.6, is from a server component is functionally identical as before shown in Figure 2.8, the server receives either a blank request for the top of the lattice, as before, and responds with data corresponding to the most relevant tags in the top of the lattice within the tag limit size. Or the server receives a request with a selected tag hash and a list of already selected tags, in the order in which they were selected, and responds with the tags representing the relevant portion of the lattice.

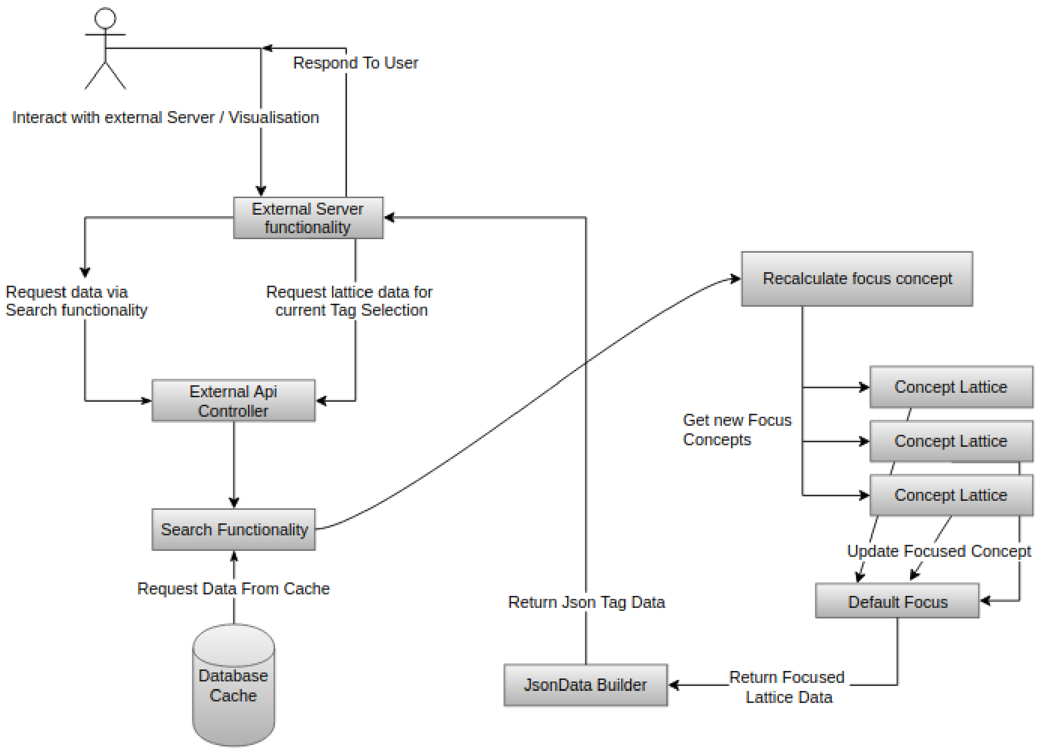


Figure 3.6: ConceptCloud external API's interaction with ConceptCloud's navigation components.

The user interacts with the external server or service, that server then requests data from the ConceptCloud external facing API, each tag has a hash which is required to identify the selected item in the lattice, the extended API hands this request to a lattice builder which then updates the focus concept of the lattice, builds the new section of the lattice and sends the resulting data to the *JSON* Data Builder, this databuilder then serialises the Tag data into tag information and frequency of occurance in the resulting lattice into *JSON* format. This data is then wrapped in CORS headers and sent back to the ConceptCloud external API and then the requesting server.

3.4 Data Presentation

As stated in Section 2.3, ConceptCloud represents data as interactive tags. Each of these tag i is scaled by minimum and maximum font sizes f_{min} and f_{max} in relation to its count t_i in relation to the minimum and maximum counts in the context table, t_{min} and t_{max}, yielding:

$$size(i) = \left[\frac{(f_{max} - f_{min}) \times (t_i - t_{min})}{t_{max} - t_{min}} \right] + f_{min} - 1$$

This was originally created to ensure that tags with lower counts remain visible. But for larger datasets where there is a much larger variance between the minimum and maximum values this tends to squash the larger tags down and does not capture the magnitude of the variance. Functionality was added to re-scale all tags currently displayed in the Tag viewers, by different forms of either linearisation or normalisation.

Linearisation

I removed the maximum size and added a variable scaling factor, I kept a minimum fontsize ensuring nothing totally disappears. This provides the user with a simple linear scaling purely scales by the font scale and the tag's frequency t_i.

$$size(i) = fontscale * t_i + 1$$

Normalisation

The normalisation functionality is split into global normalisation or per window normalisation, it should be made clear that the domain of these scaling functions is only on the displayed tags. This scaling is performed on the initial tag size.

Global Normalisation Each tag is scaled by its count t_i normalised against either its occurrence in all tag clouds displayed $t_i^{occurrenceG}$, or by the counts of all tags displayed, t^{gc} scaled by a flat scaling parameter $FlatScale$.

$$size * (i) = \left[size(i) \times \frac{t_i}{t_i^{occurrenceG}} \right] + 5$$

$$size * (i) = \left[FlatScale \times \frac{t_i}{t_i^{gc}} \right] + 5$$

Per Window Normalisation Each tag is scaled by its count t_i normalised against the counts of all tags in its local tag cloud, t_i^{CL} scaled by a flat scaling parameter, $FlatScale$.

$$size(i) = \left[FlatScale \times \frac{t_i}{t_i^{CL}} \right] + 5$$

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

Figure 3.7: Table showing measured response times of old and new Concept-Cloud architectures for listed user actions.

3.5 Experiments

To show the difference in the architectures, I 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 keyphrase 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: The user actions carried out involved the following; changing the category filter to varietal, 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 I am no longer rendering the entire dataset, but only the top 5000 tags and a far smaller table view.

Even when rendering fewer 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.

3.6 Wine Review Data Case Study

In this section a case study on wine reviews is conducted. The purpose of this case study is to illustrate the functionality and robustness of the implemented architecture while providing an insight into a typical ConceptCloud use case.

Data

The dataset used was a set of 16306 Wine Reviews for 5 star wines, (points ranging from 82 to 100), from the years 2005 to 2016. The vintages reviewed were from 1995-2014. The wines come from 22 different countries or states. Each review has 14 attributes. This dataset was selected as it did not run on the previous architecture, a short investigation into the data illustrating the new ConceptCloud Architecture is conducted below.

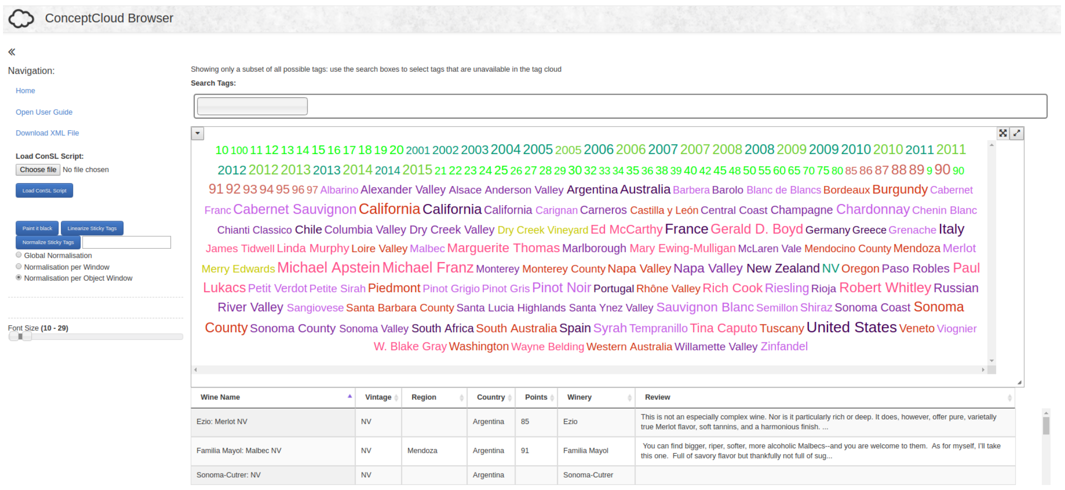
We wish to see if with no prior knowledge of the dataset, is it possible to find common trends just by an explorative search, and if so do these trends provide further insight into the data, or provide us with questions for further study.

Main View

In Figure 3.8 the default Wine Review Data Main Tag Cloud view is shown, and we immediately note the issue discussed in Section 3.4, we are unable to discern the differences in size between each tag, if we really focus we can see that Italy, France and the United States are the highest contributing countries in our dataset. But discerning any difference between the these is not possible by a simple eye test. Hovering over each we see their counts, 2144, 2579 and 5335 respectively.

In Figure 3.9 the linearised Wine Review Data Main Tag Cloud view is shown, the linearisation causes the major trends in the data to become more apparent, we note that the United States tag is roughly twice the size of that of the Italy and France tags, as we expect from the linearisation function. We note too that the bulk of our wine reviews scored in the range of 87-94 points.

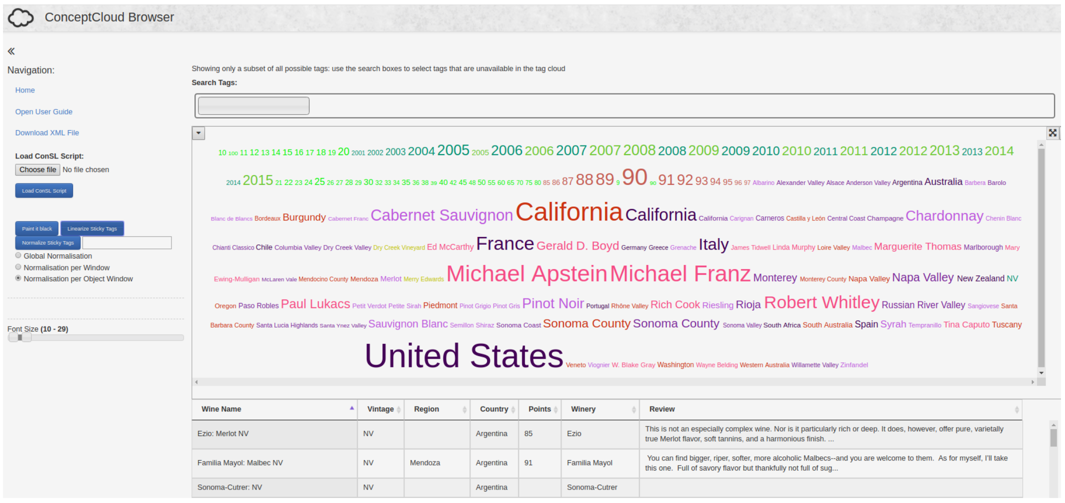
ConceptCloud allows for the isolation of specific concepts and the visualisation of extra tag clouds for them. In Figure 3.10 I break down the points range 87-94 into 8 distinct viewers, each viewer represents a section of the lattice where the focus concept is the selected red points tag, these views have been linearised to help us note any trends.



Showing only a subset of all possible tags: use the search boxes to select tags that are unavailable in the tag cloud

Wine Name	Vintage	Region	Country	Points	Winery	Review
Ezio: Merlot NV	NV		Argentina	85	Ezio	This is not an especially complex wine. Nor is it particularly rich or deep. It does, however, offer pure, varietally true Merlot flavor, soft tannins, and a harmonious finish. ...
Familia Mayol: Malbec NV	NV	Mendoza	Argentina	91	Familia Mayol	You can find bigger, riper, softer, more alcoholic Malbecs--and you are welcome to them. As for myself, I'll take this one. Full of savory flavor but thankfully not full of sug...
Sonoma-Cutrer: NV	NV		Argentina		Sonoma-Cutrer	

Figure 3.8: ConceptCloud tag cloud showing Wine review with no linearisation



Showing only a subset of all possible tags: use the search boxes to select tags that are unavailable in the tag cloud

Wine Name	Vintage	Region	Country	Points	Winery	Review
Ezio: Merlot NV	NV		Argentina	85	Ezio	This is not an especially complex wine. Nor is it particularly rich or deep. It does, however, offer pure, varietally true Merlot flavor, soft tannins, and a harmonious finish. ...
Familia Mayol: Malbec NV	NV	Mendoza	Argentina	91	Familia Mayol	You can find bigger, riper, softer, more alcoholic Malbecs--and you are welcome to them. As for myself, I'll take this one. Full of savory flavor but thankfully not full of sug...
Sonoma-Cutrer: NV	NV		Argentina		Sonoma-Cutrer	

Figure 3.9: ConceptCloud tag cloud showing linearised Wine review data

In our linearised views we see that the highest density are 88,89,90,91 points. Amongst these we see that United States is the largest in each. The two main reviewers in each are Michael Apstein and Michael Franz.

In our linearised Views if I isolate vintage and varietal we are easily able to see, as per Figure 3.11 that Cabernet Sauvignon, Chardonnay and Pinot Noir are the most common varietals across our points break down, it is somewhat more difficult however to isolate how each performed over the years. The ConceptCloud Scaling architecture does however allow us to extend the views to the entire points range, 82 - 100, and isolate a varietal. I do this for Cabernet Sauvignon, Pinot Noir and Chardonnay in Figures 3.12,3.13,3.14 respectively.

Each of these Figures 3.12,3.13,3.14, have been normalised as per the normalisation discussed in Section 3.4. This normalisation adjusts the size of the tags in each window relative to the total count for each window. This allows

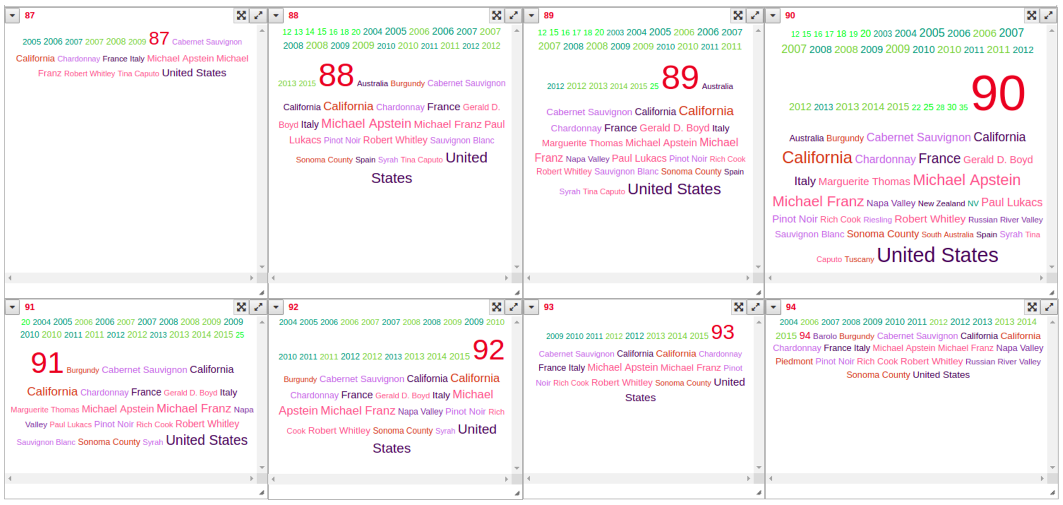


Figure 3.10: ConceptCloud stickied 'Points' viewers for points values 87-94

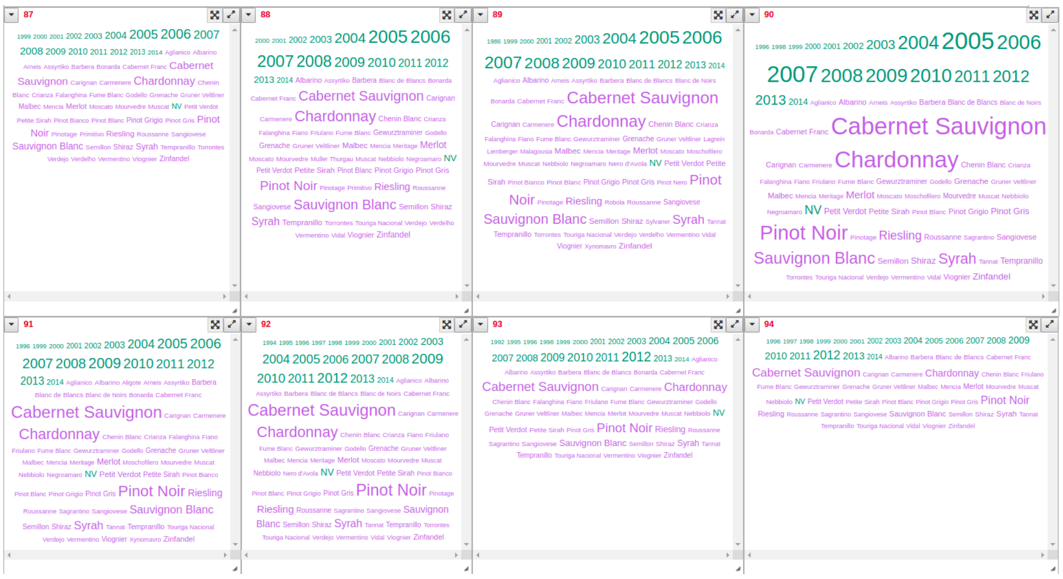


Figure 3.11: ConceptCloud stickied 'Points' viewers for points values 87-94 filtered by vintage and varietal.

us to more easily see the larger year tags even for years with very few reviews.

In Figure 3.12 We can see an almost linear trend in vintage to review score, which is what we would expect as the dataset's review years were from 2005-2016. That is that as the points score increases, the vintage year grows, even in our normalised views, which adjust the sizes to avoid a bias to higher volume years. One would expect a linear improvement in wine score as time progresses as the wine makers release better vintages, which is what we see, interestingly the cellaring time, that is the difference between the year the wine was reviewed and its vintage begins to decrease. Figure 3.13 shows there

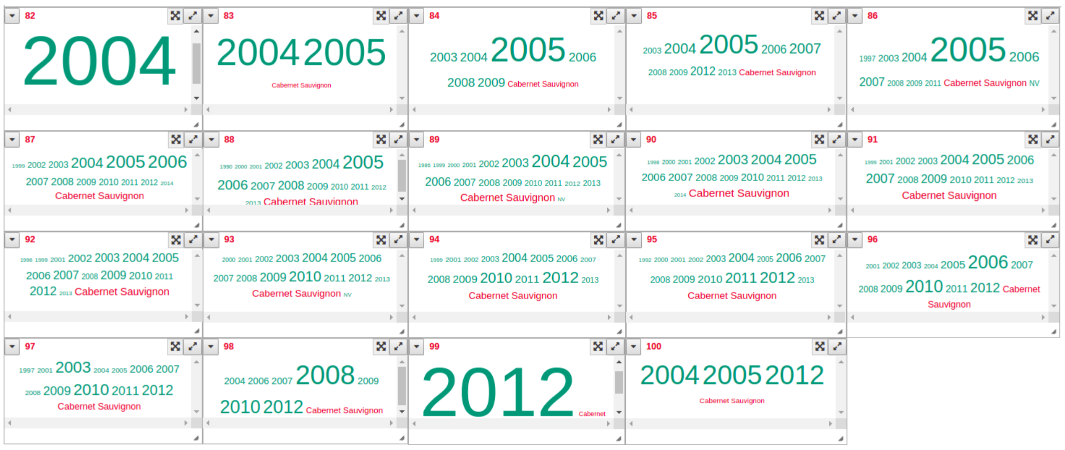


Figure 3.12: Normalised per window vintage and varietal viewers across points for cabernet sauvignon.

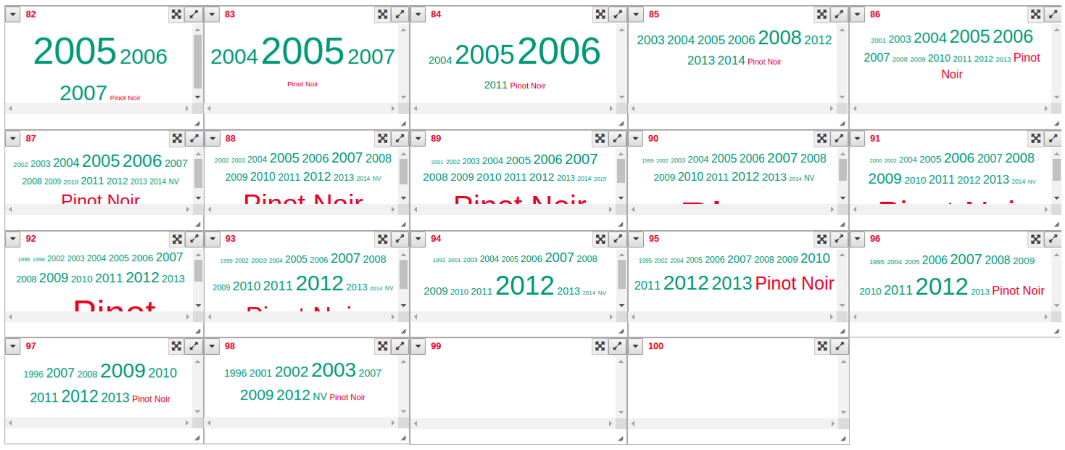


Figure 3.13: Normalised per window vintage and varietal viewers across points for Pinot Noir.

is progress toward a similar linear trend with regards to Pinot Noir scores over time, but the bulk of the scores are still from the pre 2009 period. This may be due to Pinot Noir having a higher cellar time, but this requires further investigation.

Figure 3.14 has a similar trend to the Pinot Noir, growth towards a newer vintage at higher point thresholds, but the ranges from 82-92, and 97 and up are dominated by the pre 2009 vintages.

This leads to the question of, as the point values increase, does the cellar time increase?

To further investigate this I retrieved our data and calculated the cellar time for each wine across the review points range. I then average the cellar times per point value, to see if there is a linear trend as we suspect from our explorative search.

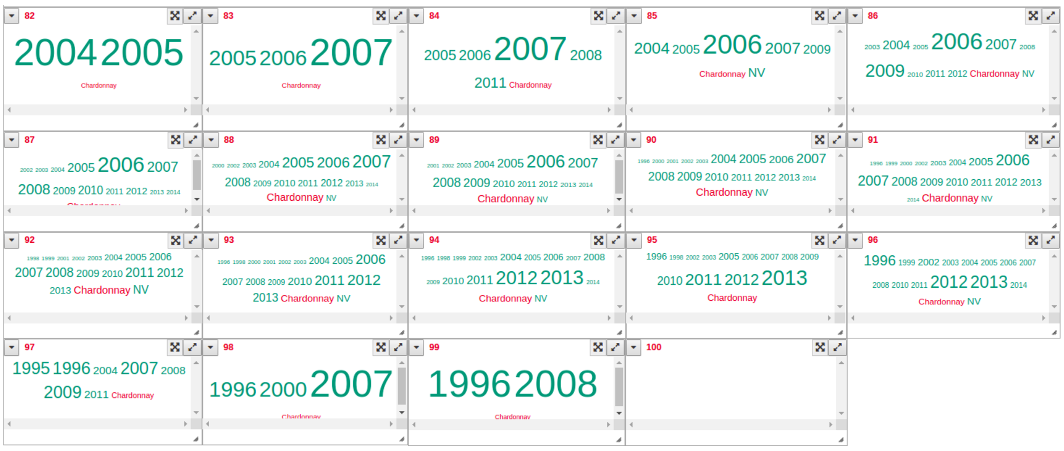


Figure 3.14: Normalised per window vintage and varietal viewers across points for Chardonnay.

In Table 3.1 the average cellar time, where cellar time is the time between the vintage and the review year, is shown against the point distribution of the dataset, for Cabernet Sauvignon, Pinot Noir and Chardonnay respectively. Table 3.2 shows the frequencies across across the point distribution for Cabernet Sauvignon, Pinot Noir and Chardonnay.

We note a linear increase in the cellar time across all three wines as the point values increase, which corresponds to our ConceptCloud screens in Figures 3.12; 3.13; 3.14. This points the notion that in this data there is a relationship between the increased cellar time and the increased review score that may warrant further investigation.

A further point of investigation may be to perform this comparison, not for isoloated varietals but for varietals grouped into red and white wines respectively. Though this would require some corpus of domain specific information to perform the classification into red or white wine varietals.

Future Work

I 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.

The planned future work relating to system usability is to dockerize the ConceptCloud project, and to automate the remaining manual steps described in Section A.1. This would allow users with little to no programming experience to be able to use the project on their datasets.

The approach here should be to build a docker container for the ConceptCloud project, allowing the required dependencies to be bundled in with the ConceptCloud system, this then effectively removes the need for the user to

Poi-nts	Cabernet Sauvignon	Pinot Noir	Char-donnay
82	2	2	1.5
83	2.5	1.75	1
84	2.22	1.92	1.29
85	2.86	1.8	1.75
86	2.9	2.2	1.9
87	2.64	2.14	1.9
88	3.06	2.32	2
89	3.28	2.28	2.06
90	3.38	2.46	2.25
91	3.39	2.53	2.4
92	3.74	2.69	2.58
93	3.58	2.62	3.04
94	3.7	2.69	2.68
95	3.81	2.72	3
96	3.69	2.98	4.37
97	3.67	3.18	6.18
98	3.73	3.33	7.5
99	2	x	6
100	3	x	x

Table 3.1: Average Cellar Time in years against Points for Cabernet Sauvignon, Pinot Noir and Chardonnay

Poi-nts	Cabernet Sauvignon	Pinot Noir	Char-donnay
82	1	4	2
83	2	4	4
84	9	12	7
85	21	9	16
86	30	40	34
87	104	85	116
88	175	158	196
89	241	181	226
90	404	309	359
91	240	209	191
92	233	227	182
93	152	149	116
94	132	114	81
95	120	89	63
96	55	42	26
97	36	17	11
98	15	12	4
99	1	0	2
100	3	0	0

Table 3.2: Frequency against points for Cabernet Sauvignon, Pinot Noir and Chardonnay

install anything apart from docker and its dependencies. Within the docker build script part of the setup described in Section A.1 can be performed. The remaining part should then be handled with a shell script, merely taking in the name of the *JSON* file in which the dataset sits.

Conclusion

In this subsection I have shown that it is possible to make use of Concept-Cloud to gain insights into a dataset with no prior exposure to it, by way of explorative search. Additionally I was able to use a far larger dataset then previously possible.

The investigation did however highlight the need to integrate a domain specific corpus of information, to aid in adding complex concepts, such as the cellar time attribute to our lattice, or to separate varietals into white and red wines.

A more in depth study could be conducted to compare the performance of the various countries across our points range.

3.7 Maps Extension

In order to showcase the extensibility of ConceptCloud and its ability to process different data sets, the base architecture was modified to include an interactive Map interface. The addition of the map viewer itself was done in conjunction with another masters student Tiaan Du Toit, thesis forthcoming. The core of the work covered in this thesis is the back end support for the map based viewer and the data aggregation.

The main motivation behind this specific extension is the emergence of semi-structured data sets that incorporate some geolocation aspect, for example, the location of the winery for wine reviews, or the accident location for traffic data. With the rapid increase in always-on, embedded GPS devices, such geolocation aspects are becoming more prevalent. This abundance presents the opportunity to perform more widespread knowledge discovery across further domains by exploring the geolocation aspect of the data, but also demands a different analytic approach [22] – clearly, textually displaying latitude and longitude in a tag cloud is not optimal. Historically, maps have most commonly been used to visualise the geolocation aspects of such data sets, my first challenge was thus *how do I support a specialised map visualisation in a tag cloud based data exploration tool?*

Although maps are immediately useful for visualising the geolocation aspect of data, this geolocation aspect further allows for a new way to aggregate data, that is along a locality that is not explicit. *Can I aggregate data across non-explicit boundary points?*

Another aspect of these semi-structured data sets with geolocation aspects, is that the speed at which this data is being generated is increasing, resulting in larger and larger sets. This fact requires that the visualisation and exploration tool be highly scalable in order to process extremely large data sets.

With maps providing a time-tested method of exploring the geolocation aspect of data, and tag clouds providing an effective method for facilitating knowledge discovery and data visualisation for semi-structured data, the overall goal of this section is therefore to merge these two proven methods into an integrated and scalable package.

Biclusters

When working with geolocation rich large semi-structured data sets there often exist biclusters formed from the geolocation aspect of data. Ordinarily a bicluster is defined to be a pair (A,B) of inclusion-maximal sets of objects and attributes such that almost all objects in A have almost all attributes in B. This technique has been implemented and applied successfully to mine numeric data sets[21].

A bicluster of tags in a numeric context would be all tags whose many valued context tables' numeric attribute falls within a certain tolerance. I.e all wines whose grapes grow between 18-32 degrees Celsius.

A bicluster of geolocation data would in this case have a forced inclusion of the geolocation attribute on all objects in A, as well as the numerical values of the geolocation attribute all being within some tolerance, which would correspond to each object being in a similar location on a map. I.e all wines from a particular region on a map, or all wines from areas near the coast on a map.

I explore and group together these naturally occurring biclusters and mine them to view common trends where possible, as described in Section 3.8. These biclusters are used to generate a new tag cloud which can be mined further.

Disjunctive selection

In order to expand on the geolocation exploration and maintain a single focus IR navigation algorithm [19], Boolean disjunctive selection is utilised. This techniques involves modifying the underlying context table of the lattice by making use of an OR operator in the query sent to the server and generating a new temporary concept lattice of only the selected objects from the data set. These are either from objects in the bicluster, or objects picked by the user, or further, entire biclusters selected by the user. In Figure 3.16 I present a new running example based on crimes, as these datasets were both very large and had readily available geolocation aspects to be explored.

	Mossel Bay	Stellen-bosch	Paarl	Ash-ton	Ars-on	Kidnap-ping	Select-ed
1			x		x		
2	x					x	x
3		x			x		
4				x		x	
5				x	x		x

Figure 3.15: Context table showing South African crimes with objects selected

Computing the Boolean OR of two or more objects involve assigning a new "meta"-tag to the selected objects, and in doing so, generating a new temporary lattice on the fly. This new lattice consisting of the merged objects becomes the focus of the new tag cloud window, and the user is free to explore the desired objects without introducing concept broadening [19].

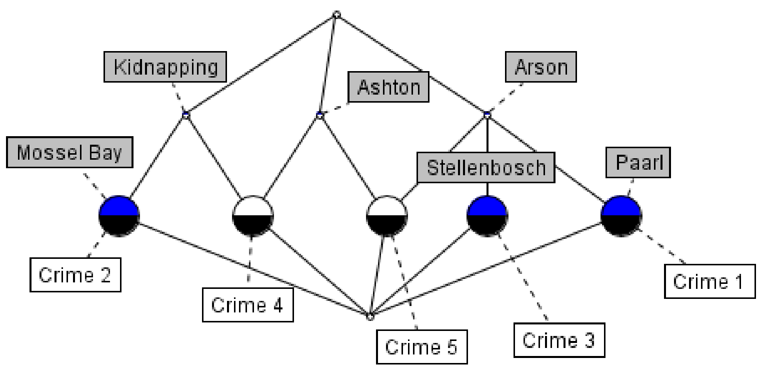


Figure 3.16: Formal concept Lattice without any objects selected for formal context items in Figure 3.15.

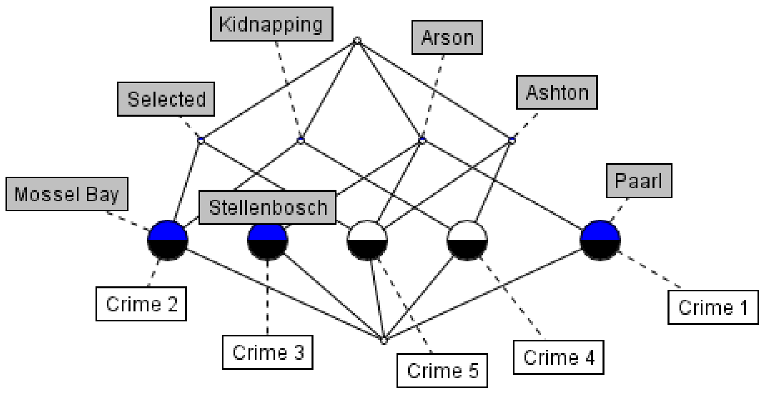


Figure 3.17: Formal Concept Lattice with objects selected

Without disjunctive selection, the join of the focus and the attribute concept of "Kidnapping" would return the top concept in the lattice and our new lattice would contain attributes of other types (such as "Arson"), as can be seen in Figure 3.16. By using the boolean OR operation I am able to retrieve all crimes that are only of the "Kidnapping" OR "Arson" types along with the attributes these crimes possess. This allows the user maximum control in selecting only the formal objects which the user wishes to further explore.

3.8 Map-Viewer Implementation

Map Visualisation

Geolocation data has most intuitively been displayed on maps for hundreds of years, making a map based visualisation the obvious choice to display geolocation data. A map based viewer allows the user to most easily distinguish the differences in geolocation values by providing them with an intuitive representation of the spacial dimensions of the data.

Given that ConceptCloud organizes data into context tables of objects and attributes, our map representation would then be the map based visualisation of the set of all objects in $\mathcal{O}$ that have geolocation attributes. This corresponds to the notion of a formal concept. For a focus concept $c := \langle O, A \rangle$ we have that for a subconcept of the focus concept $\mathcal{M} := \langle X, Y \rangle$ where X is the set of all objects which contain attributes with geolocation data, and Y is the set of all attributes for all objects in X.

Technical Implementation

The maps viewer implementation makes use of the Google Maps JavaScript API to render a fully interactive map and generate markers based on the objects' geolocation attribute, where available. The markers then function the same as a tag in the tag cloud and corresponds to an exploratory search as described in [18].

The software allows for two-way filtering between the word cloud and map window. The user may start exploration of the concept lattice by either selecting an attribute in the word cloud which will update the cloud as described above as well as update the map window to display only the objects with that selected (focus) attribute. Alternatively, the user may select an individual marker which will drill down the concept lattice to that specific formal object and update the word cloud with only that object and its attributes, as well as removing all other markers from the map.

The map and marker objects generated by the Google Maps API [8] are populated by a single specialised ConceptCloud server call. The ConceptCloud tool allows the user to pre-configure the data attributes they wish to appear in each map pin object, allowing for a far smaller, and therefore more responsive server call to create each marker. The ConceptCloud server returns a set of objects, containing at least an identifier and the geolocation, from which a Google Maps marker object is created and populated with the pre-configured attributes of that object.

When working with larger data sets or data sets that contain multiple objects in close proximity, an issue arises where densely populated regions on the map become difficult to navigate. To maintain a functional user interface,

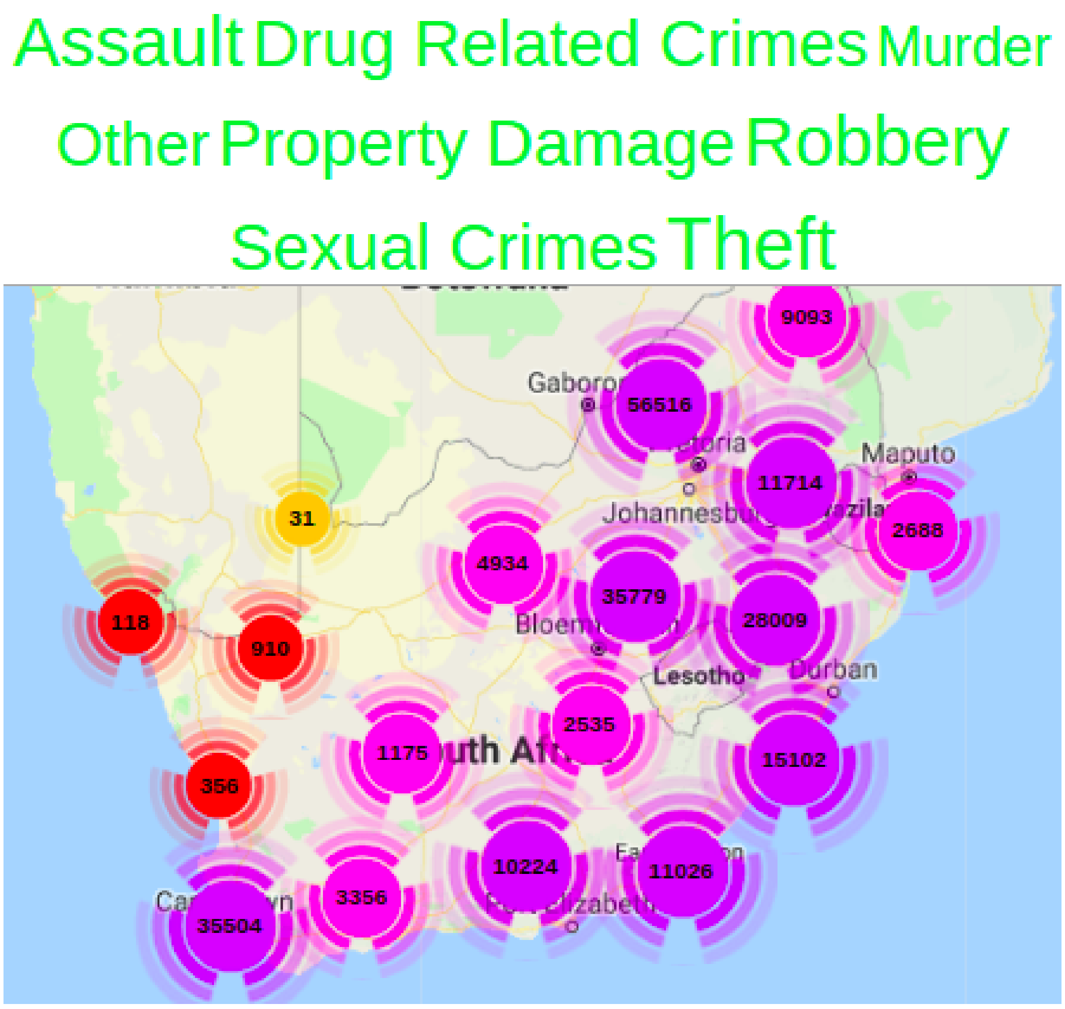


Figure 3.18: ConceptCloud map viewer for crime data

the generated markers are clustered automatically within a shifting geolocational tolerance, based on the current zoom level of the map. This clustering is handled by an external JavaScript library [29] that automatically clusters the markers and displays a count of the markers in each cluster.

The location of the rendered cluster on the map is the location of the last marker added to that cluster. Smaller clusters are also clustered into bigger clusters at higher zoom levels. The user may make use of these clusters to generate a new tag cloud that can then be further explored.

This bicluster corresponds to a smaller lattice created from the main lattice, with its own focus concept derived from the main focus concept, that can be navigated through independently of the main lattice.

The user is able to create tag clouds to visualise the data contained in a selection of a single, or multiple biclusters.

Architecture changes

ConceptCloud makes use of a Postgresql database back end. PostGIS is an extension for Postgres that converts standard latitude and longitude into specialised data types using a specific SRID, either geographic or geometric, depending on whether the user is dealing with 2D or 3D shapes, in order to execute more advanced geographically optimised queries.

The extension allows the user to index these new geometric data types and group them according to their real world locations, allowing for faster querying/look up when making use of PostGIS query methods/functions.

In order to make full use of the PostGIS data transformation and indexing/clustering I use the `ST_D_within` method which takes 3 parameters, namely a latitude, a longitude and a radius/distance measured in SRID. The latitude and longitude are then converted to a geometry data type.

Making use of the location of the Google Maps viewer window as a starting point for our server call, initially we pass the centre coordinates of the map viewer to the server, together with the zoom level of the viewer. We make a server call and return only pins that are visible to the user. The size of the visible map in the Google Map viewer depends on the actual size of the window. Using the dimensions of this viewer, we then calculate the visible radius in km/2. We then use this distance together with the zoom level and centre coordinates of the map viewer to get map pins that are visible to the user. We keep track of movement in the map viewer and do a server call whenever the map is idle to fetch newly visible pins.

Result of these changes

These changes enabled the ConceptCloud system to now process massive datasets, while providing support for a developer to be able to implement an explorative search visualiser of their own. This functionality is demonstrated with the maps viewer in a case study below. Finally the architecture changes and separation of the explorative search functionality allows for viewers to be implemented on mobile devices.

Case Study

Data set

The semi-structured data set used in our investigation involves the official SAPS type [34] and count of the various crimes reported to every police station in South Africa for the year of 2006. This data set contains over 2.2 million unique crime incidents, each containing the type of crime, the station at which the crime was reported and finally, the year in which the crime was reported. For privacy reasons the geolocation of each crime is the latitude and longitude of the police station at which it was reported.

This large data set also allowed me to test the robustness of the software tool as well as its scalability. In my testing it became apparent that the Google Maps API [8] is a limiting factor, since it adds marker objects to the map individually before being clustered by the third-party library. This process is highly memory intensive, and the web browser becomes unresponsive when exceeding 250 000 markers, although with the map viewer closed the rest of the ConceptCloud tool ran no differently to the previous case study in this chapter. Thus it was decided that the investigation would be conducted on a random subset of the total data set, limited to a representative 10%, or 229 070 crime objects only.

Exploration

I make use of the most powerful exploratory functionality of the software tool by selecting multiple naturally occurring, automatically generated biclusters. By right-clicking a cluster on the map, a new tag cloud window is created containing only the attributes of the objects in that cluster. This new word cloud is then used for further exploration. The size of the generated clusters can be modified by changing the zoom level of the map. I make use of these biclusters to investigate a case study that would be difficult and unintuitive to accomplish using traditional exploratory tools.

A prevalent issue found in geolocation data sets is the limiting effect of pre-imposed borders on the data. These borders are typically political or geographical. Often the data does not correspond directly to these borders. Crime data is a good example of this, where crimes are not restricted by political borders but perhaps by natural or socio-economic borders instead.

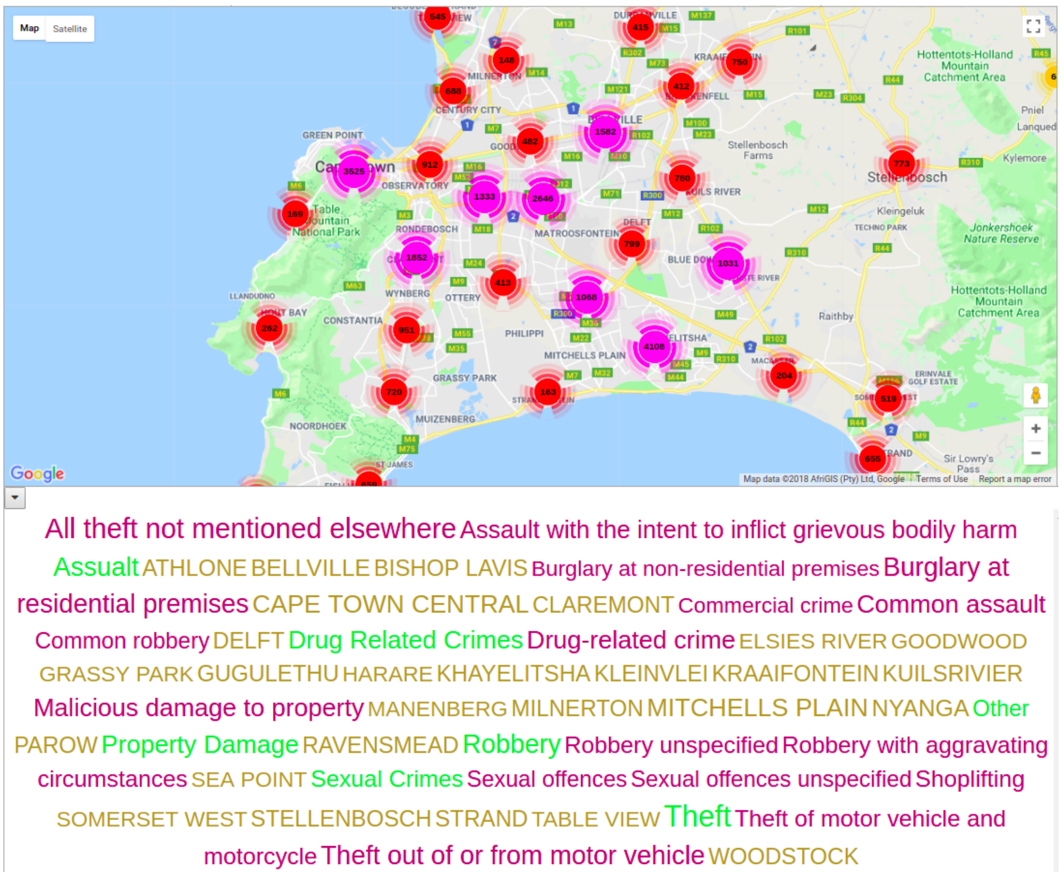
Generally any quantitative or statistical data collected is automatically confined to these borders. Any query on the data involving unique intersections of these borders become very complex, but the improvements made to the ConceptCloud architecture provide a solution to this problem. By allowing the user to freely select biclusters to explore, they are not restricted by these boundaries.

In order to demonstrate this functionality, I investigate the crimes reported at police stations in and surrounding the two biggest cities in South Africa, namely Cape Town and Johannesburg. The data was aggregated by selecting all the biclusters in the greater surrounding area, which was then used to generate a new bicluster based, that is a new lattice with only the tags occurring in the biclusters was generated and used to power the tag cloud's explorative search, which in turn was used to drive the investigation in the case study.

Results

Results are presented in two screenshots of the tag cloud viewer as well as the map viewer in ConceptCloud used to demonstrate the flexibility of the additions to ConceptCloud.

The following two screenshots show the crime biclusters for the regions around each of the biggest cities in South Africa, namely Cape Town, Johannesburg (I include Pretoria in the Johannesburg region). I purposefully select crime biclusters that do not fall strictly within the municipal or political borders of these regions, but rather select them based on proximity. I determine the three highest occurring crime categories and crime types for each region. I exclude "All theft not mentioned elsewhere" from our crime types comparison due to the vague nature, and resulting inflated incident count, of this type.

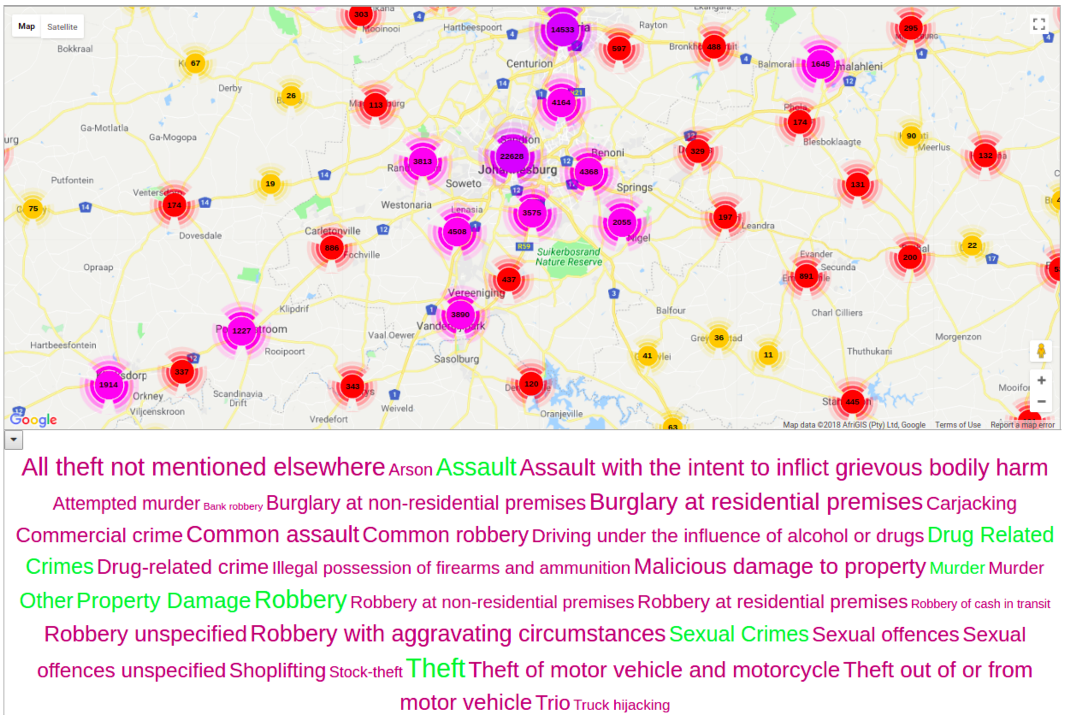


All theft not mentioned elsewhere Assault with the intent to inflict grievous bodily harm Assault ATHLONE BELLVILLE BISHOP LAVIS Burglary at non-residential premises Burglary at residential premises CAPE TOWN CENTRAL CLAREMONT Commercial crime Common assault Common robbery DELFT Drug Related Crimes Drug-related crime ELSIES RIVER GOODWOOD GRASSY PARK GUGULETHU HARARE KHAYELITSHA KLEINVLEI KRAAIFONTEIN KUILSRIVIER Malicious damage to property MANENBERG MILNERTON MITCHELLS PLAIN NYANGA Other PAROW Property Damage RAVENSMEAD Robbery Robbery unspecified Robbery with aggravating circumstances SEA POINT Sexual Crimes Sexual offences Sexual offences unspecified Shoplifting SOMERSET WEST STELLENBOSCH STRAND TABLE VIEW Theft Theft of motor vehicle and motorcycle Theft out of or from motor vehicle WOODSTOCK

Figure 3.19: ConceptCloud map viewer and tag cloud showing the Greater Cape Town area bicluster.

Cape Town Bicluster: The biclusters selected for Cape Town contains 29099 individual crime incidents. From the newly generated tag cloud viewer generated from these biclusters, I determine that the "Theft", "Assault" and "Robbery" crime categories occur most frequently in the selected objects. From Figure 3.19 I also determine that, "Theft from a motor vehicle" is the most common reported crime in this region (2930 crime objects), followed by

”Burglary at a residential premises” (2851 crime objects) and lastly ”Common Assault” (2354 crime objects). If required, I can explore these crimes further and drill down to a single object.



All theft not mentioned elsewhere Arson Assault Assault with the intent to inflict grievous bodily harm Attempted murder Bank robbery Burglary at non-residential premises Burglary at residential premises Carjacking Commercial crime Common assault Common robbery Driving under the influence of alcohol or drugs Drug Related Crimes Drug-related crime Illegal possession of firearms and ammunition Malicious damage to property Murder Murder Other Property Damage Robbery Robbery at non-residential premises Robbery at residential premises Robbery of cash in transit Robbery unspecified Robbery with aggravating circumstances Sexual Crimes Sexual offences Sexual offences unspecified Shoplifting Stock-theft Theft Theft of motor vehicle and motorcycle Theft out of or from motor vehicle Trio Truck hijacking

Figure 3.20: ConceptCloud map viewer and tag cloud showing the Greater Johannesburg area bicluster.

Johannesburg Bicluster: Next I investigate biclusters in the greater Johannesburg region. Our selection contains 75 969 individual crime incidents, and once again the ”Theft”, ”Assault” and ”Robbery” crime categories occur most frequently in our new tag cloud viewer shown in Figure 3.20. I also determine that ”Burglary at a residential premises” (87 271 crime objects) occurs most frequently in this region, followed by ”Common Assault” (6675 crime objects), and ”Assault with the intent to inflict grievous bodily harm” (6643 crime objects).

Discussion

National Trends

Cape Town

I observed that Cape Town has a far higher average incidence of Drug Related Crimes than the rest of the country. This was hypothesised to be due to depressed socio-economic issues in the Cape flats and the increased influence of the use of "Tik" in the area at the time[14]

Johannesburg

The Johannesburg bicluster was observed to be very similar to the National Averages. Theft, Robbery and Assault were the major crime categories committed, and by drilling deeper, I found that the most frequent crime type in this region to be Burglary at a residential premises. This corresponds to a report from 2006 that suggested that this was due to a strike of security guards in Gauteng, and an increase in the police countermeasures to other forms of robbery. [28]

Conclusions

By exploring the semi-structured crime data set using ConceptCloud's tag viewer, and exploiting the geolocation aspects of that data in the integrated map viewer, I demonstrate the flexible nature of ConceptCloud. It allowed for the an entirely new visualisation while still maintaining the core explorative search functionality, and allowed the viewers to interact seemlessly with each other. ConceptCloud also allowed me to explore the data using a method not easily achieved in any other software tool, by ignoring explicit borders and aggregating the data based on its map viewer displayed locality.

Although I have proven the functionality of the ConceptCloud software, a systematic usability study is still required to determine the intuitiveness and ease of use of the software , similar to the methods used in [10].

As stated in Section 3.8, when making use of data sets exceeding 250 000 objects, the browser did become unresponsive. After some investigation it was determined that that the Google Maps API [8] is a limiting factor, as it is very resource intensive. The ConceptCloud software itself proved to be highly scalable, processing the full 2.5 million object data set without any error or decrease in performance.

Future Work

Further studies can be conducted by making use of the ConceptCloud software tool in conjunction with other geographically rich data sets such as winery or traffic data. This will allow researchers and domain professionals to gain a deeper understanding of their data sets and mine it in a unique way.

In addition, some improvements can be made to the software tool in the form of front end marker caching in order to improve performance on larger data sets. These improvements would include using a webcaching system to improve responsiveness for larger datasets. This will allow for a quick retrieval of data for large datasets.

Another aspect of this study that can be expanded upon is the preprocessing of the data before it is entered into the ConceptCloud software. By making use of a formal domain ontology, certain attributes of the semi-structured data set can be made more coherent. In addition to this, attributes that implicitly follow from the explicitly stated attributes of an object can be inferred [2, 33].

Chapter 4

Ontology Driven Data Imputation

Semi-structured data is often sampled from human input data, which may often contain implicit information which will be useful in driving data exploration if made explicit.

We wish to take advantage of domain-ontologies to both allow implicit data in each input data set to be made explicit and verify and correct inconsistencies allowing for better data exploration.

4.1 Introduction

ConceptCloud is a semi-structured data exploration tool; as such, the benefits gained by the exploration of the data used are primarily determined by the quality of the input data.

As an example, recall Figure 1.4, from Section 1.4, and a similar example shown below in Figure 4.1: Here I am once again working with the wine dataset, and looking at Chenin Blanc Wines from Stellenbosch. I intuitively know that I am looking at a subset of South African white wines, I know this because of the implicit knowledge that a Chenin Blanc is a white wine, and Stellenbosch is in South Africa. Suppose I wished to see this subset of wines compared to the greater set of South African White Wines, that is, all wine objects having a region, area or location attribute South Africa, and a varietal Chenin Blanc. Unless the data items in my subset explicitly contain the attribute South African White Wine, I would not be able to visualise this in the ConceptCloud tool.

Suppose no FCA concept exists for this grouping. The lattice will not contain a subset of the overall dataset with the shared implicit attribute, South African White Wines, as no such concept exists explicitly. This shortcoming in many input datasets gives rise to the requirement to allow the user to

```
{
    "name":"Mont Destin Mont Destin range Chenin Blanc 2007",
    "winery":"Mont Destin",
    "location": "Stellenbosch",
    "region": "",
    "country": "",
    "varietal": "Chenin Blanc",
    "vintage": "2007",
    "price": "12",
    "points": "70",
    "reviewer": "Michael Apstein",
    "reviewyear": "2008",
    "review": "06 less beguiling than pvs, still ample charms
    through 14.5% alc, with pineapple & mineral signature."
}
```

Figure 4.1: Example Chenin Blanc wine *JSON* object with missing data

enable ConceptCloud to supplement their dataset with extra knowledge to make navigation by these implicit properties possible.

4.2 General Approach

The act of making implicit properties or information explicit has been primarily covered in both formal logic, and knowledge engineering [31]. As explained in Section 1.4, an ontology in the same domain of information as our input dataset therefore serves as a natural source of the explicit properties for our data.

My general approach is to allow the user to input a domain ontology, which I will use to complete the formal context as far as possible.

Implicit Information

Implicit information refers to information that follows from the explicit information available in domain ontologies and data sources, and which is accessible through logical consequence, or entailment. In ConceptCloud, I break this down into two categories, namely controlled vocabulary information and free-text information. Controlled vocabulary information refers to information contained within fixed attributes with controlled values. For example, given a wine, it will have a varietal, this varietal can only have a value from a well defined and highly constrained set of terms, for example, the wine presented below in Figure 4.2.

```
{
 "name": "Stark-Condé: Cabernet Sauvignon ‘‘Three Pines" 2007",
 "winery": "Stark-Condé",
 "location": "Jonkershoek Valley",
 "region": null,
 "country": "South Africa",
 "varietal": "Cabernet Sauvignon",
 "vintage": "2007",
 "price": "42",
 "points": "91",
 "reviewer": "Michael Franz",
 "reviewyear": "2010",
 "review": "Intense and deeply flavorful, this is a dramatic
 wine that grabs one’s attention and doesn’t let go. Perhaps
 its most impressive aspect is its aromatic complexity, as
 notes of dark berries, cassis, smoke, pine needles and
 eucalyptus all show themselves quite notably. Full-bodied,
 with plenty of tannin and oak, this is a wine for pairing
 with assertive foods like grilled lamb."
}
```

Figure 4.2: Example Cabernet Sauvignon *JSON* object

The values of these items will often lead to the exposure of further implicit information. For example, given a Cabernet Sauvignon varietal, I know the wine must then be a red wine, as the varietal concept value ‘Cabernet Sauvignon’ exposes the wine colour concept value ‘red’. Here the depth of the implicit information available is entirely determined by the constraints of the knowledge base in use. If in our knowledge base we have that all Red wines in our domain are from the Cape Winelands region, then we must have that our wine with the Cabernet Sauvignon varietal must too have the Cape Winelands region. Alternatively that the region our wine is from must be contained in the Cape Winelands region, (if of course, all wines are from a region in our knowledge base).

Free-text refers to information contained in a non controlled vocabulary, which is presented as a free-text attribute within our formal context. Here, the implicit information must first be extracted into a controlled vocabulary before our algorithm can process it.

For example, refer to the review attribute of Figure 4.2, the text present here is uncontrolled and does not conform to any particular fixed vocabulary. In fact, in the case of wine reviews, the text is intentionally obfuscated. The added ambiguity is introduced to avoid repetition between reviews.

There is still implicit information present here, with the wine described as intense and deeply flavourful, I know this refers to the taste, but there is never an explicit reference to taste as such. So extracting these implied subject properties can be difficult.

Somewhat easier to extract is implicit information with explicit subjects, here I know precisely what the descriptors are tied to, this process is more straightforward when I have a matching free-text subject and FCA attribute pair. For example, the phrase "aromatic complexity, as notes of dark berries, cassis, smoke, pine needles and eucalyptus" all refer to the aroma of the wine. This phrase can then be extracted and added to the aroma attribute of the wine, provided you have natural language processing that is powerful enough to link each of the aroma attributes to the aroma subject. The phrase would ideally match a many-valued formal context table attribute named aroma.

4.3 Vocabulary Mapping Between FCA and DL

To access the implicit information available from the ontology, I will need to access the statements made explicit by logical consequence.

To achieve this implicit information access, these entailments must first be calculated with an automatic reasoning process. The HermiT reasoner makes this process available to us [32].

However, for the reasoner to calculate the entailments of our FCA attributes, I must first map them to either $\mathcal{A}$Box assertions or $\mathcal{T}$Box class descriptions.

The most intuitively simple and straight forward version of this mapping problem is to make the implicit information contained in the controlled vocabulary. Where I have a match between an FCA attribute, and a corresponding description logic concept assertion, class description or object property assertion axiom describing the FCA attribute within the domain of the knowledge base.

Most often this corresponds to matching a particular FCA attribute to a DL concept, class description or object property assertion, recall Section 2.5, here there is a clear link between the FCA attributes, and the assertions or class descriptions. The operation linking the FCA attribute to the corresponding DL vocabulary member or symbol is relatively simple. There is a direct match between the controlled vocabulary terms used in both the class name or class description, contained within the aforementioned DL vocabulary member or symbol, and the FCA attribute.

As such, mapping controlled vocabulary FCA attributes to existing DL vocabulary, and mapping the corresponding logical consequences is the first goal for our data imputation operation. The second is to match controlled vocabulary FCA attributes to DL vocabulary members (class descriptions and $\mathcal{A}$Box assertions) where there is an indirect link between the two. That is, the terminology in either the DL vocabulary member or FCA attribute is a member of a controlled vocabulary wherein a list of synonyms exist and can be used to link the two. However, this may lead to inconsistencies as there is no longer a direct match between the axiom and attribute. It would be far safer to add an explicit equivalence axiom to the ontology, backed and verified by domain knowledge.

Finally, the mapping requires mining the free text, or uncontrolled vocabulary of a many-valued context table attribute, such as a review and linking to DL vocabulary members within the input ontology. This mapping has an inherent reliance on the natural language processing available. This is covered in Section 4.7.

Specific Approach

My specific approach to ontology data imputation is to mirror the formal context attribute data into the $\mathcal{A}$Box of the input ontology. Data is mapped into the ontology; we perform inference with our reasoner and extract the logical consequences back into our formal context.

Given our input ontology, I map the DL-concepts from the class hierarchy, the available object property assertion axioms and data properties to FCA attributes, and their potential values. That is, from the class hierarchy, a subclass with no children is a potential value, and its parent a potential FCA attribute. We then use natural language processing, namely the Stanford Core NLP library, to check all attributes in our formal context table against an N-gram of potential attributes in the ontology class hierarchy. From this, we create a mapping between the formal context table attributes and the ontology classes and data types.

Suppose there is a free-text attribute in the formal context table. In that case, I analyse it with key-phrase extraction guided by our potential attributes and values to continue expanding our map between formal context table attributes and ontology vocabulary members.

I then use this mapping to create new individuals with the aforementioned object property, class and data property assertions. Next, I add the new individuals to the ontology. In this way, the first part of the data reflection from many-valued FCA context table to ontology has been completed. Following this, we use the reasoner to calculate our inferred assertions for each individual and map each back to the corresponding many-valued context table object entry. We take care to only add data properties to these individuals that already exist in the ontology, as such only the $\mathcal{A}$Box of the ontology is being expanded. This precaution is taken to maintain consistency across the ontology.

After each individual is added to the ontology, all implicit properties are calculated, and then an expanded ontology is produced.

This expanded ontology is then mined for each individual. Any attributes in the ontology not present in the formal context table are added to the individual in the formal context table. In this way, I add FCA attributes which follow from our ontology to each corresponding FCA object in the formal context.

4.4 Algorithm

The basic overview of the algorithm can be seen on an architectural level in Figure 4.9. Based on the user-specified configuration parameters, the system will perform reasoning on the input data.

The data imputation controller mainly handles the reasoning process. This controller reads in the data set and formal context table data, as well as the ontology vocabulary (inclusive of all individuals and their related $\mathcal{A}$Box assertions). It then makes use of the NLP controller to mine and perform key-phrase extraction on the formal context table data. The data imputation controller then uses this NLP data to link FCA attributes to the corresponding ontology vocabulary, that is class assertions and descriptions, object property assertion axioms and data property assertion axioms within the domain ontology.

This mapping happens for each object in the formal context. The controller then adds each applicable ontology vocabulary item to the object, which then is added as an individual in the formal context table. These axioms representing the new individual and its attributes are then added to the ontology and verified for consistency by the reasoner. If an individual cannot be added to the formal context table, the reasoner provides a justification for the failure and logs it. This full failure would occur when none of the $\mathcal{A}$Box axioms can be added to the individual as they each fail the reasoner consistency checks.

Once each object in the formal context table has been added to the ontology with its corresponding $\mathcal{A}$Box axioms, the reasoner computes the inferred ontology. The inferred ontology is then mined against each object in the formal context table, the data imputation controller then adds the extra applicable attributes to the formal context table, and the data repository.

Once this data enrichment process is complete, the user is able to perform navigation as before, as described in Section 2.3.

The algorithm is described in detail below:

Algorithm

The algorithm comprises of three main phases, attribute and axiom mapping, ontology enrichment and inference and ontology mining and lattice completion.

Attribute And Axiom Mapping

The attribute and axiom mapping algorithm computes a list of target attributes and values from the formal context table data. These are then mapped to OWL classes and individuals, respectively. For an item in the context table, each of its existing attributes and attribute values is checked against the list of OWL object property assertion axioms. That is, for some context item X with the many-valued context pair `varietal:Chenin Blanc` the algorithm will search for all axioms associated with Chenin Blanc. The first set of axioms it checks against are object property assertion axioms. In our example it would find the axiom $X, CheninBlanc$:hasVarietal. It then saves this target assertion in a list against the context table item. The algorithm makes use of a template-based approach, that is for the set of many-valued context pairs, a mapping to object property assertion axioms will be used. As an item is present in the context table, I know that we should be working with is-a or has-a relations. For example for many-valued context pair `location:Paarl` the algorithm would match `location:Paarl` to the object property assertion axiom $X, Paarl$:hasLocation.

As a further step, the system will create a secondary list for items for which no individual exists. That is, given `location:Paarl`, if no individual Paarl, which is of the OWLClass Location, (determined off of the range of the hasLocation object property assertion), exists in the system, an individual Paarl, with OWLClass location, will be added to the list of axioms to add to the ontology. These secondary individuals are later referred to as independent individuals as they can be added independently of the addition of the individual whose assertion spawned them. In the case of $X, Paarl$:hasLocation, I call X the dependent individual.

Suppose later in our axiom mapping, in the class assertion axiom mapping phase described below; I ran into the many-valued attribute pair `town:Paarl`, then the axiom Paarl:Town would be added to the list of axioms to add to the ontology.

Following the mapping of object property axioms, class assertion axioms and data property assertion axioms are mapped in the same way.

For the context pair `WhiteWine:  X` the algorithm matches the class assertion axiom X:WhiteWine. Following the above assertions I then have that X in the ontology is a white wine, which has location Paarl and has the varietal Chenin Blanc.

Ontology Enrichment and Inference

Once a context table item has been completely mined for possible attributes, the two sets of axioms are added to the ontology. First, all individuals that are not the context table item itself, (a wine in our running example) are added. In this way, I would first add the individual Paarl and its OWLClass location attribute to the ontology. Then the context table item with its axioms is added. To achieve this, a new OWL individual is created, then each axiom is added to the individual, with a consistency check. If an axiom fails the consistency check, it is not added to the individual and thus the ontology.

This consistency check process is to ensure that while I am altering our ontology by adding new individuals, I am doing so in a manner that does not compromise the consistency of the ontology.

Once the many-valued context table item has been added to the ontology, the algorithm will then mine the next item in our many-valued context table for axioms, build the corresponding OWL individual, and any related individuals, and add them to the ontology. This is performed until the context table is exhausted.

Once all individuals from the context table have corresponding ontology individuals, the inferred ontology is computed by the reasoner.

Lattice Completion

The lattice completion algorithm mines data from the inferred ontology and makes use of it to complete the dataset used to generate the lattice. That dataset is then used as described before in Section 3.2.

The lattice completion algorithm reads in each item in the formal context table, and for each formal context table object, does a corresponding call to the reasoner of the inferred ontology for the individual corresponding to the formal context table object.

If that object exists in the ontology, a secondary call to the reasoner is performed to fetch all of its attributes. These attributes are then added to the formal context table attributes for the object and written to a new data file.

Once each object's extended attributes have been added to the file, ConceptCloud reloads its data using the enriched data set.

Inference Example

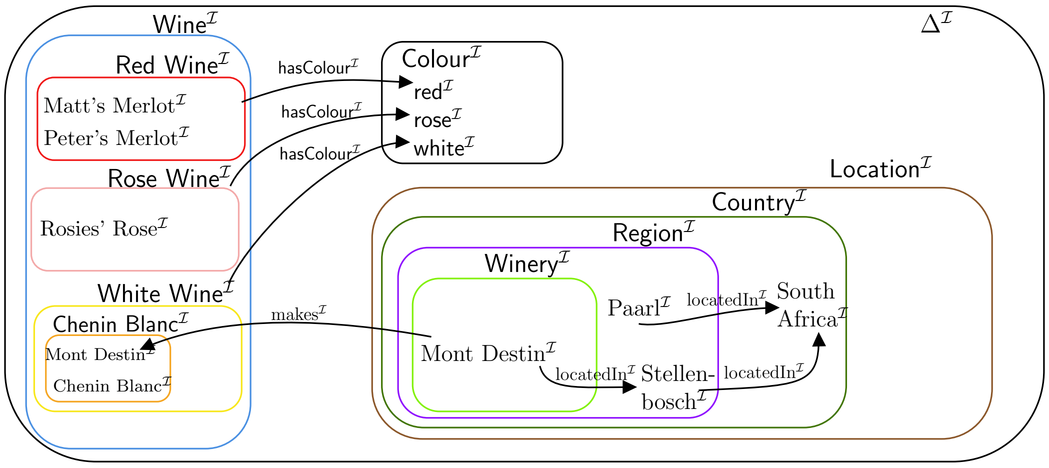
Recall our example domain and interpretation from Section 2.5 presented in Figure 2.17, and again below in Figure 4.3.

Also, recall entailment for a knowledge base defined as in Section 2.5

and that a model is a satisfying interpretation of the knowledge base as defined as in Definition 8.

We will now run through the reasoning process followed for the wine individual Mont Destin Chenin Blanc.

Note in Figure 2.17 Mont Destin appears both as a Chenin Blanc wine and as a Winery. Similarly, Chenin Blanc appears both as an individual, and as a concept name. Ambiguities like this frequently arise in wine review data, and the mapping algorithm must handle the disambiguation thereof.



$\Delta^{\mathcal{I}}$
Wine$^{\mathcal{I}}$
Red Wine$^{\mathcal{I}}$
Matt's Merlot$^{\mathcal{I}}$
Peter's Merlot$^{\mathcal{I}}$
Rose Wine$^{\mathcal{I}}$
Rosies' Rose$^{\mathcal{I}}$
White Wine$^{\mathcal{I}}$
Chenin Blanc$^{\mathcal{I}}$
Mont Destin$^{\mathcal{I}}$ Chenin Blanc$^{\mathcal{I}}$
hasColour$^{\mathcal{I}}$
hasColour$^{\mathcal{I}}$
hasColour$^{\mathcal{I}}$
makes$^{\mathcal{I}}$
Colour$^{\mathcal{I}}$
red$^{\mathcal{I}}$
rose$^{\mathcal{I}}$
white$^{\mathcal{I}}$
Location$^{\mathcal{I}}$
Country$^{\mathcal{I}}$
Region$^{\mathcal{I}}$
Winery$^{\mathcal{I}}$
Mont Destin$^{\mathcal{I}}$
Paarl$^{\mathcal{I}}$
locatedIn$^{\mathcal{I}}$
South Africa$^{\mathcal{I}}$
locatedIn$^{\mathcal{I}}$
Stellen-bosch$^{\mathcal{I}}$
locatedIn$^{\mathcal{I}}$

Figure 4.3: Possible Interpretation $\mathcal{I}$ of the domain information presented in Chapter 2

The algorithm processes the individual Mont Destin Chenin Blanc, with its corresponding context table entry, based on the semi-structured data presented in Figure 4.1 as follows:

Attribute and Axiom Mapping The goal of the attribute mining portion of the algorithm is to extract axioms which match up to the context table data. That is, to form a pairing between an attribute in the context table and an object property axiom in the OWL ontology.

The axiom mining portion of the algorithm begins by creating a cache of all possible object property axioms for the target class, in this case, Wine, in the ontology.

For our example these are as follows:

1. madeBy
2. madeAt
3. hasWinery
4. hasVarietal
5. madeBy
6. hasColour

Then I compile a second list of all possible subclasses for our Wine, as follows:

1. RedWine
2. RoseWine
3. WhiteWine
4. SouthAfricanWine

Finally these are compared with the list of many-valued context table attributes, presented in Figure 4.4:

$$\begin{aligned}\mathcal{A} :=\{&\textsf{name: Mont Destin Mont Destin range Chenin Blanc 2007,}\\ &\textsf{winery:Mont Destin,location:Stellenbosch,region:,}\\ &\textsf{country:, varietal:Chenin Blanc, vintage:2007}\\ &\textsf{price:12, points:78, reviewer:Michael Apstein}\\ &\textsf{reviewyear:2008,review: 06 less beguiling than pvs,}\\ &\textsf{still ample charms though at 14.5\% alc,}\\ &\textsf{with pineapple \& mineral signature.}\}\end{aligned} \tag{4.1}$$

Figure 4.4: DL-Attributes for mDCB07 individual

The algorithm will then match the available object property axioms to the many-valued context table attributes to form the following axioms describing the Wine individual "Mont Destin Mont Destin range Chenin Blanc 2007", which will be referred to as {mDCB07} going forward.

The object property axioms are matched with a template method; that is, I assume that the object property axioms have a fixed format wherein a predefined keyword denotes the property. In the current example `has` is used.Then all object property axioms are matched to many-valued context table attributes on the `has`X form, where X denotes the many-valued context table attribute. These first template-based mappings are performed with the following result:

1. {mDCB07,montDestin}:hasWinery
2. {mDCB07,cheninBlanc}:hasVarietal

This covers the first layer of information I can mine, that is information directly linked to our root class individual, the Wine, mDCB07.

Further information relating to the Winery individual montDestin and the Varietal individual cheninBlanc may be mined. This is covered later.

Ontology Enrichment and Inference Given the mapping of an individual to object property axiom, that is:

1. (mDCB07,montDestin):hasWinery
2. (mDCB07,cheninBlanc):hasVarietal

We note that for these axioms to be valid in our ontology the individuals mDCB07,montDestin,cheninBlanc must exist.

The second step of the algorithm performs exactly that creation, a lookup into the individuals in the ontology is performed, if the Wine individual mDCB07 does not exist in the ontology it must be added, so too for the Winery individual montDestin and Varietal individual cheninBlanc.

If the montDestin winery individual does not exist in the ontology I must add it, as such, the many-valued context table item needs to be mined for attributes that tie to a winery in the ontology. These are then mapped to the montDestin ontology individual in a similar way to how I mapped axioms to the mDCB07 Wine individual. This is shown below:

1. (montDestin,Stellenbosch):hasLocation
2. (montDestin,cheninBlanc):makes

The key difference here is for montDestin; I find a target list of axioms, hasLocation,makes, based on the available axioms in the ontology with domain Winery, these are then checked against the attributes to find a mapping. Here as I am dealing with individuals that are one relation away from our root class individual, in this case, the winery. I only check or add individuals to the ontology; I do not mine further properties for those individuals. That is to say; I am not trying to mine further information about the location of the Winery Mont Destin, only the location itself. In our use case, I assume the further properties to already be present within the ontology. For example, Mont Destin has location Stellenbosch / Western Cape; I have ontology information stating that Stellenbosch or the Western Cape is in South Africa. I am not trying to determine from my mining, whether Stellenbosch or the Western Cape is located in South Africa. To go further and to determine the above is future work and beyond the scope of this thesis.

Finally let's assume that the Varietal individual cheninBlanc already exists in our ontology with the following axioms:

1. cheninBlanc:Varietal
2. cheninBlanc:WhiteWine
3. cheninBlanc:$\exists$ madeAt.Winery

Now I know Chenin Blanc is a varietal, and that it is a white wine.

The algorithm will then find the Varietal individual cheninBlanc, and be able to use that in the previous object property assertion axiom.

The first step is to add the non-existent independent individual to the ontology, in this case it would be the Winery individual montDestin, which is added to the ontology by adding the following axioms to the $\mathcal{A}$Box:

1. montDestin:Winery
2. (montDestin, Stellenbosch):hasLocation

3. (montDestin, cheninBlanc):makes

After each axiom, 1,2,3 above is added, a check is performed to ensure the ontology remains consistent.

It is important to note that each individual added is assumed to be distinct, as such, axioms are added to ensure it is distinct to all other individuals of the same subclass. This is as `OWL` does not have the unique name assumption. This distinction axiom addition was optimised to be a single large axiom per subclass that is added after all individuals have been added that sets all individuals in our ontology subclass to being distinct from each other. The optimisation was required as setting each individual to distinct from every other individual from its subclass is exponentially computationally expensive as I add more individuals to the ontology. That is, for each individual added axioms were generated to set it to be distinct from every other individual that existed in our subclass. For example, for the case of mDCB07 if it were the 12 000'th Wine individual added to the ontology the algorithm would add 11 999 distinct individual axioms to mDCB07. If another wine mDCB08 was added after it would then require 12 000 distinct individual axioms, and so forth for each new individual added. It became far quicker and less cumbersome to create a single different individual axiom containing all individuals of the ontology per subclass after all individuals had been added.

Following the addition of the independent individual, the dependent Wine individual, mDCB07 can be added to the ontology, $\mathcal{A}$Box with the following axioms:

1. mDCB07:Wine
2. (mDCB07, montDestin):hasWinery
3. (mDCB07, cheninBlanc):hasVarietal

Following the addition of all individuals to the ontology, the reasoner is run to compute the inferred ontology.

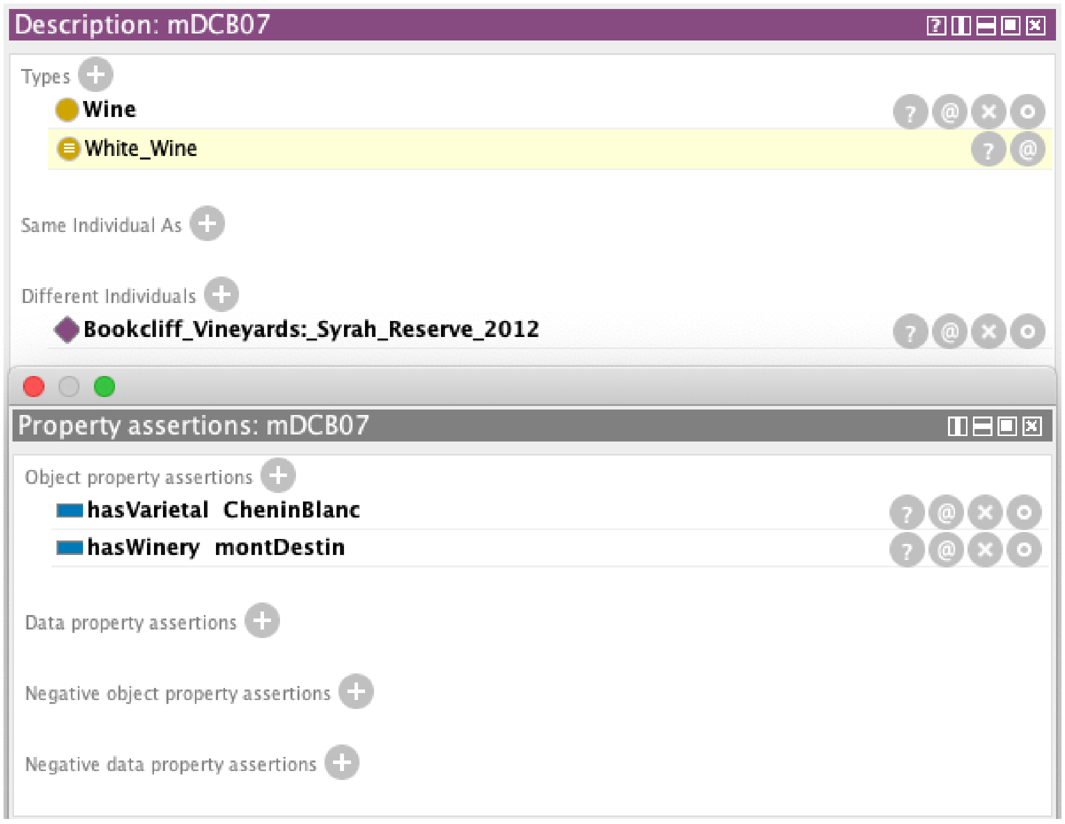
This brings us to the lattice completion phase of the algorithm.

Lattice Completion Following the reasoner's computation of the inferred ontology, the inferred information must be mined and placed back into the original dataset.

We see an example of this inference in Figures 4.5, 4.6 which show the inferred Wine and Winery individuals respectively.

In Figure 4.5 we see the only inference I have is that our Chenin Blanc individual, mDCB07 has been classified as a White Wine. Furthermore we see that the Winery individual has inferred attributes for isLocatedIn the Country individual South_Africa and the Region individual Cape_Winelands.

Recalling the attributes available in Figure 4.4, we see I will need to match the country attributes and add a new subclass attribute for White Wine.



Description: mDCB07

Types
Wine
White_Wine

Same Individual As

Different Individuals
Bookcliff_Vineyards:_Syrah_Reserve_2012

Property assertions: mDCB07

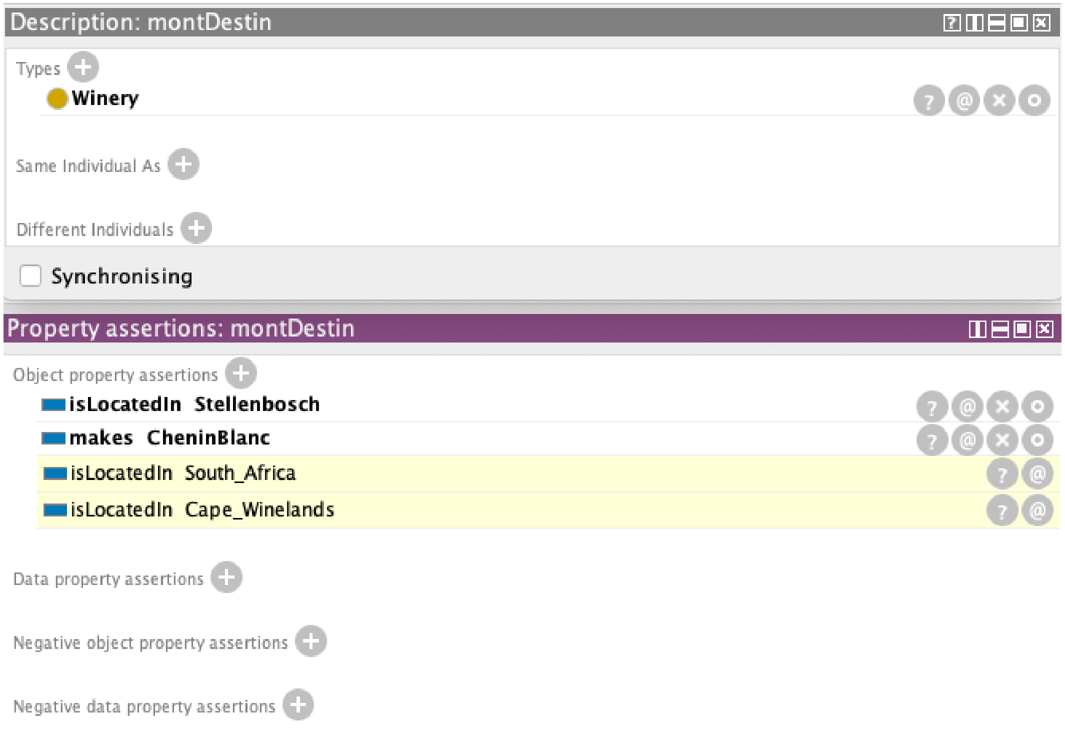
Object property assertions
hasVarietal CheninBlanc
hasWinery montDestin

Data property assertions

Negative object property assertions

Negative data property assertions

Figure 4.5: Protege view of inferred Wine individual's detail



Description: montDestin

Types
Winery

Same Individual As

Different Individuals

Synchronising

Property assertions: montDestin

Object property assertions
isLocatedIn Stellenbosch
makes CheninBlanc
isLocatedIn South_Africa
isLocatedIn Cape_Winelands

Data property assertions

Negative object property assertions

Negative data property assertions

Figure 4.6: Protege view of inferred Winery individual's detail

In so doing I am no longer only mining the attributes of the base Wine individual, but also of its object property axiom linked individuals, the Winery individual Mont Destin in this case.

This approach could be avoided if the underlying ontology was very well-tailored to the dataset, and made use of rolification [23], to link the attributes of the independent individual directly to the dependent base individual. The above approach is further covered in Section 5.2, but as the scope of this thesis is to show the viability of this form of data imputation and not to fine-tune an ontology to supplement the dataset, it was not implemented.

The algorithm then takes the list of many-valued attributes from the context table, and compiles a list of target attributes, in the form of many-valued context table attributes. Secondly, it takes the list of attributes, in the form of axioms, from the dependent individual, and those from its independent individuals and compiles a second list and then constructs a mapping between the many-valued attribute and axiom attribute lists.

When a class or subclass axiom is present in the axiom mapping list has no corresponding many-valued attribute, in the many-valued context table attribute list, a many-valued attribute based off of the subclass is added to a class attribute in the many-valued attribute list and the mapping is completed. This is as the class attributes are used in ConceptCloud to separate the data during the navigation process, and as such their addition to a many-valued attribute is fairly useful, as seen in Section 4.6. An example of this is seen in Figure 4.12, where the DL subclass Red Wine has been added to the many-valued class attribute and can clearly be seen in the tag cloud as a `class` tag, of value `Red Wine`, corresponding to the many-valued context table attribute `class: Red Wine`.

After the mapping has been completed, the attributes are added to the base dataset in *JSON* format. This is then used as the new dataset for ConceptCloud as described in Chapter 3.

```
{
    "name":"Mont Destin Mont Destin range Chenin Blanc 2007",
    "winery":"Mont Destin",
    "location": "Stellenbosch",
    "region": "Cape Winelands",
    "country": "South Africa",
    "varietal": "Chenin Blanc",
    "vintage": "2007",
    "price": "12",
    "points": "70",
    "reviewer": "Michael Apstein",
    "reviewyear": "2008",
    "review": "06 less beguiling than pvs, still ample charms
    through 14.5% alc, with pineapple & mineral signature."
    "class": "White Wine"
}
```

Figure 4.7: mDCB07 *JSON* object following data imputation process

After this process has been completed on the individual mDCB07, its *JSON* object is as in Figure 4.7

Here we note the following many-valued context table attributes have been added:

$$\begin{aligned} A^* :=\{&\textsf{region: Cape\_Winelands,}\\ &\textsf{country: South\_Africa,}\\ &\textsf{class: White Wine}\\ &\} \end{aligned}$$

Figure 4.8: Inferred attributes for mDCB07 individual

The mapping is performed by checking the owlclass of the axiom against the many-valued context attribute, that is, the owclass:country from the independent individual can only be mapped onto the dependent individual. In this case mDCB07, on a match between the class of the range of the axiom containing the information to be mined from the independent individual, to the dependent individual's many-valued context table entry.

That is the ontology axiom A_1 for the Winery individual, montDestin, linked to the dependent Wine individual, mDCB07 by the axiom (mDCB07,montDestin):hasWinery, where

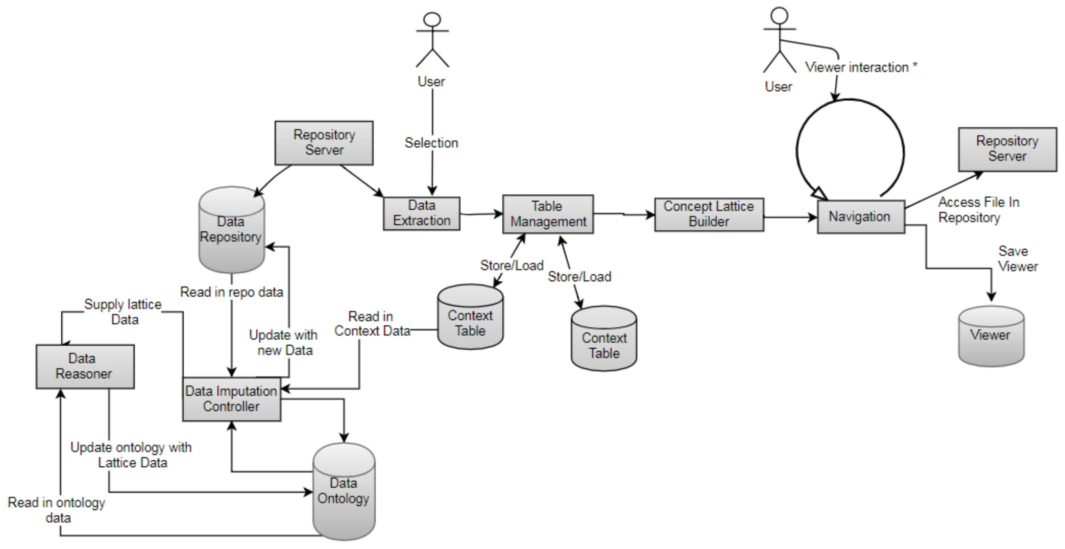


Figure 4.9: Data imputation alongside ConceptCloud architecture

A_1 := (montDestin, South_Africa):isLocatedIn. Here the individual South_Africa is a country individual and is therefore mapped to mDCB07's many-valued context table attributes, as in Figures 4.7, 4.8.

4.5 Implementation details

Architecture

The data-imputation subsystem exists to enrich the underlying data set, as such, it was built to interact purely with the underlying data set, and aside from configuration, it has no user interaction. The subsystem fits into the ConceptCloud system, as shown in Figure 4.9.

The three main components of the data-imputation subsystem are the Data-Imputation Controller, the ontology and the reasoner.

The Data-Imputation Controller, or DIC, is responsible for the orchestration of the events governed by the algorithm, described in Section 4.4.

Based on the user set configuration, the DIC will begin the data-imputation process by reading in the many-valued context table data. Following this, the DIC begins reading in the ontology data; this is then used in conjunction with the context table data and the reasoner to initiate new ontology individuals for the target data class, Wine in the previous example. The reasoner is called at each step of the individual creation to validate the axioms being added with the individual to the ontology. Once all target data class individuals from the many-valued context table have been added to the ontology, the reasoner is then called to calculate the inferred ontology. Once the inferred ontology has been calculated the DIC then mines this inferred ontology and updates the underlying source *JSON* dataset.

Following this, the regular ConceptCloud data extraction pulls the entire updated dataset into the new context-table and concept lattice. Furthermore, it generates a new caching database, based off of the updated input dataset.

The user can then explore the updated dataset as previously described in Section 3.2.

4.6 Case Study

The case study is once again on the same wine review dataset used previously in Section 3.6. That is a dataset of 16306 Wine Reviews for 5-star wines, (points ranging from 82 to 100), from the years 2005 to 2016. The vintages reviewed were from 1995-2014.

The case study is to illustrate the enrichment of a dataset by ontology-driven data-imputation as described in Section 4.4.

The ontology used as a basis had a low-class density with 34 classes, the bulk of which were for South_African_Wine type axioms, where South_African was replaced with the wine of origin for the wine, with 26 distinct Country individuals, and 45 distinct region individuals. The ontology also has 100 distinct varietal individuals.

The base $\mathcal{T}$Box and $\mathcal{R}$Box axioms were purposefully made similar to the running example $\mathcal{T}$Box and $\mathcal{R}$Box presented in Section 2.5.

Main View

In Figure 4.11, we see the main view of the wine review data following data-imputation. Again as before discerning between the items by simple eye-test is difficult, we note that still Italy, France and the United States are the highest contributing countries in our dataset. Hovering over each we see their counts are 2144, 2583 and 6734 respectively.

Figure 4.12 shows the linearised view of this main tag cloud. We begin to see the far more pronounced difference between our country tags, with a large Red Wine class tag also occurring.

Comparing these results to our previous case study on the same data, we begin to see some immediate variances in the data, shown in Table 4.10.

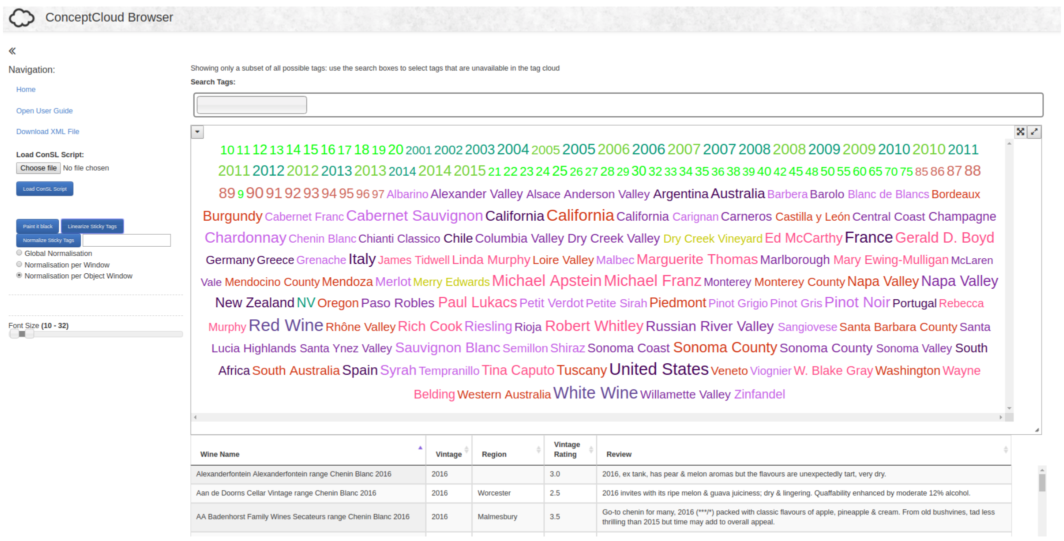
The main cause of these variances was due to the country for each wine being inferred from the region attribute on the winery.

As the region has the isLocatedIn object property axiom, it lead to the correct countries being mined from the inferred ontology. This was particularly prevalent in the cases where the country attribute listed was itself a region in the original dataset, following the mining of the inferred ontology, the correct country would be extracted. An obvious example of this can be seen when I compare the linearised Figures 3.9 and 4.12 where the country tag California is present in Figure 3.9 but barely present in Figure 4.12. Further inspection of the data present after data imputation revealed that the country California items only existed due to their region attribute and location and winery attributes being null.

With either present, due to the transitive nature of the isLocatedIn axiom, the data imputation would have correctly completed country attribute.

Country	Original Count	Post Data Imputation	variance
Italy	2144	2144	0
France	2579	2583	4
United States	5335	6734	1399

Figure 4.10: Variance introduced by data-imputation



Showing only a subset of all possible tags: use the search boxes to select tags that are unavailable in the tag cloud

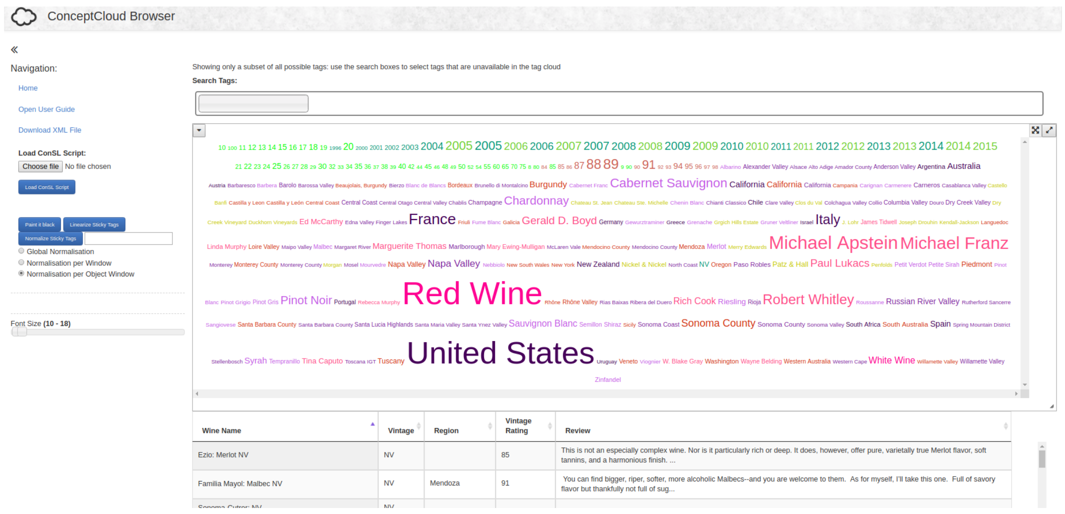
Search Tags:

10 11 12 13 14 15 16 17 18 19 20 2001 2002 2003 2004 2005 2005 2006 2006 2007 2007 2008 2008 2009 2009 2010 2010 2011 2011 2012 2012 2013 2013 2014 2014 2015 21 22 23 24 25 26 27 28 29 30 32 33 34 35 36 38 39 40 42 45 48 50 55 60 65 70 75 85 86 87 88 89 9 90 91 92 93 94 95 96 97 Albarino Alexander Valley Alsace Anderson Valley Argentina Australia Barbera Barolo Blanc de Blancs Bordeaux Burgundy Cabernet Franc Cabernet Sauvignon California California California Carignan Carneros Castilla y León Central Coast Champagne Chardonnay Chenin Blanc Chianti Classico Chile Columbia Valley Dry Creek Valley Dry Creek Vineyard Ed McCarthy France Gerald D. Boyd Germany Greece Grenache Italy James Tidwell Linda Murphy Loire Valley Malbec Marguerite Thomas Marlborough Mary Ewing-Mulligan McLaren Vale Mendocino County Mendoza Merlot Merry Edwards Michael Apstein Michael Franz Monterey Monterey County Napa Valley Napa Valley New Zealand NV Oregon Paso Robles Paul Lukacs Petit Verdot Petite Sirah Piedmont Pinot Grigio Pinot Gris Pinot Noir Portugal Rebecca Murphy Red Wine Rhône Valley Rich Cook Riesling Rioja Robert Whitley Russian River Valley Sangiovese Santa Barbara County Santa Lucia Highlands Santa Ynez Valley Sauvignon Blanc Semillon Shiraz Sonoma Coast Sonoma County Sonoma County Sonoma Valley South Africa South Australia Spain Syrah Tempranillo Tina Caputo Tuscany United States Veneto Viognier W. Blake Gray Washington Wayne Belding Western Australia White Wine Willamette Valley Zinfandel

Wine Name	Vintage	Region	Vintage Rating	Review
Alexanderfontein Alexanderfontein range Chenin Blanc 2016	2016		3.0	2016, ex tank, has pear & melon aromas but the flavours are unexpectedly tart, very dry.
Aan de Doorns Cellar Vintage range Chenin Blanc 2016	2016	Worcester	2.5	2016 invites with its ripe melon & guava juiciness; dry & lingering. Quaffability enhanced by moderate 12% alcohol.
AA Badenhorst Family Wines Secateurs range Chenin Blanc 2016	2016	Malmesbury	3.5	Go-to chenin for many, 2016 (***/*) packed with classic flavours of apple, pineapple & cream. From old bushvines, tad less thrilling than 2015 but time may add to overall appeal.

Figure 4.11: Wine review data main tag cloud

The case study performed previously in Section 3.6 highlighted the possibility to perform a further comparison between the high points scoring wines between red and white wine varietals. The baseline is presented below in Figures 4.13, 4.14. Figure 4.13 shows 8 viewers, each with a points value from 87-94 selected. Here I have normalised by country value, so the country tags become most prominent, as expected we see that France, Italy and the United States are the most prominent and that the country tag California, has been almost entirely removed.



Global Normalisation
Normalisation per Window
[✓ Normalisation per Object Window]

Showing only a subset of all possible tags: use the search boxes to select tags that are unavailable in the tag cloud

Search Tags:

Wine Name	Vintage	Region	Vintage Rating	Review
Ezio: Merlot NV	NV		85	This is not an especially complex wine. Nor is it particularly rich or deep. It does, however, offer pure, varietally true Merlot flavor, soft tannins, and a harmonious finish. ...
Familia Mayol: Malbec NV	NV	Mendoza	91	You can find bigger, riper, softer, more alcoholic Malbecs--and you are welcome to them. As for myself, I'll take this one. Full of savory flavor but thankfully not full of sug...

Figure 4.12: Wine review data main tag cloud linearised

To further investigate the relationship between points, varietals and vintages, I isolated the vintage, varietal and wineclass tags for our points viewers between 87 and 94.

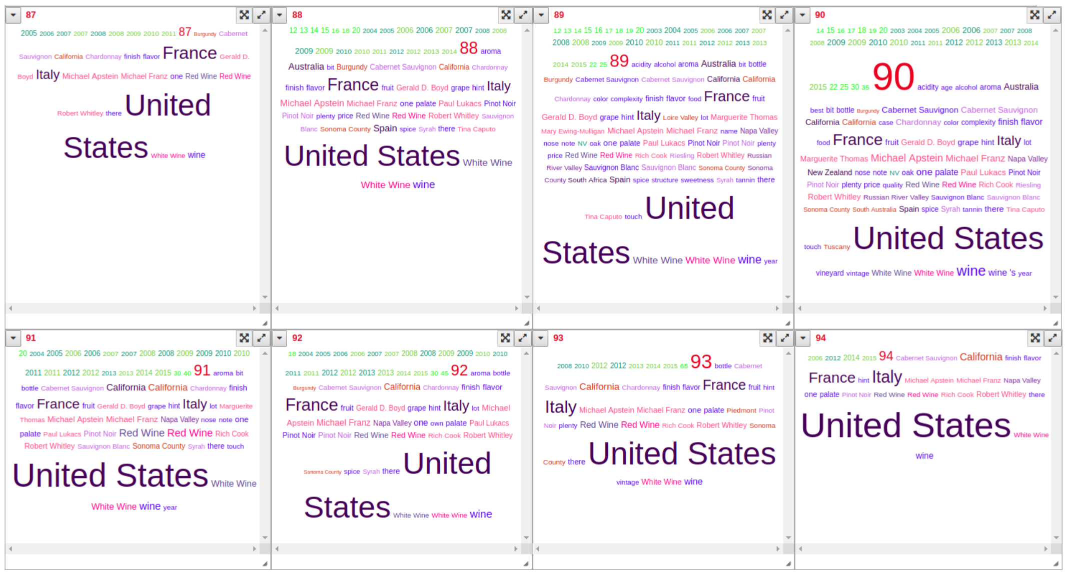


Figure 4.13: Points viewers for 87-94 normalised by country

This is displayed in Figure 4.14. Here we can see the expected trend of Red wines being more prevalent across all points values, as Red wines are generally more prevalent in the dataset.

The interesting points are that though they form a lesser part of the dataset the white wines seem to be over-represented in the 89-92 points range when compared with their red wine counterparts.

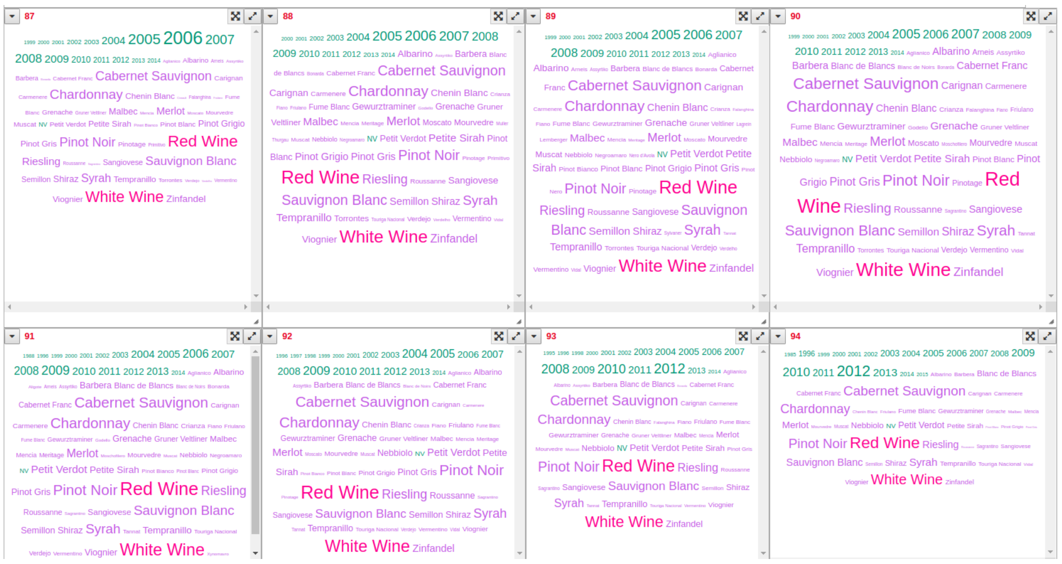


Figure 4.14: Vintage and varietal viewers for points 87-94

To see how each varietal is represented within the Red and White wine wineclasses, I separated them into separate breakdowns shown in Figures 4.15, 4.16 respectively.

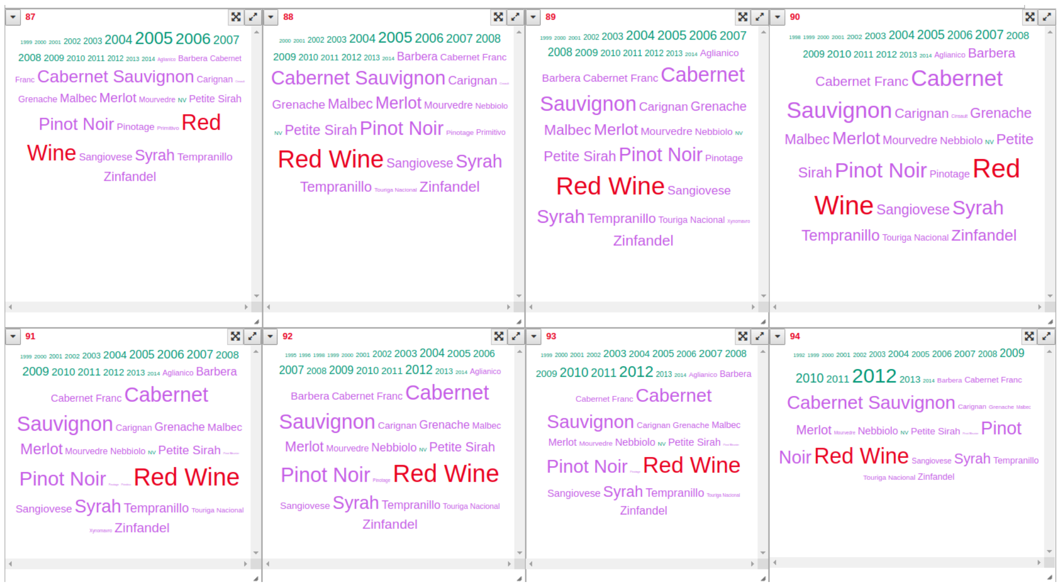


Figure 4.15: Red wine viewers for points 87-94

In Figure 4.15, we see that Cabernet Sauvignon, Pinot Noir and Syrah are well represented across all of the points ranging from 87-94. In contrast Merlot is well represented from 87-91 but drops off reasonably significantly in the higher points range in comparison to the three other aforementioned varietals.

We also note the vintages 2005-2007, 2009 and 2012 generally outperformed all others across the points range from 87-94.

Finally, Figure 4.16 shows the points breakdown for white wine varietals. Here we see that the Chardonnay and Sauvignon Blanc varietals are well represented across the entire points range, followed by the Riesling varietal. Notably, Chardonnay maintains the strongest presence throughout whilst Sauvignon Blanc and Riesling drop off in the higher points ranges. Finally, we note that the distribution of vintages from points ranges 87-92 is fairly even while for the 93,94 2012 and 2013 seem to be particularly well representing, hinting that these were good vintages.

A more in-depth study could be done to determine why these particular white wine vintages scored so highly.

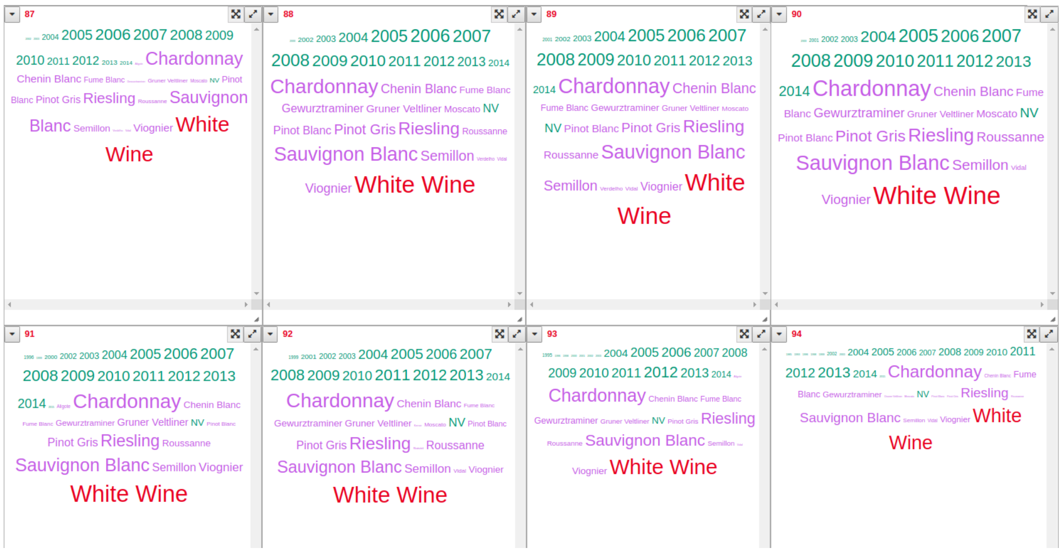

87
2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 Chardonnay Chenin Blanc Fume Blanc Gruner Veltliner Moscato NV Pinot Blanc Pinot Gris Riesling Roussanne Sauvignon Blanc Semillon Viognier White Wine
88
2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 Chardonnay Chenin Blanc Fume Blanc Gewurztraminer Gruner Veltliner Moscato NV Pinot Blanc Pinot Gris Riesling Roussanne Sauvignon Blanc Semillon Verdelho Vidal Viognier White Wine
89
2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 Chardonnay Chenin Blanc Fume Blanc Gewurztraminer Gruner Veltliner Moscato NV Pinot Blanc Pinot Gris Riesling Roussanne Sauvignon Blanc Semillon Verdelho Vidal Viognier White Wine
90
2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 Chardonnay Chenin Blanc Fume Blanc Gewurztraminer Gruner Veltliner Moscato NV Pinot Blanc Pinot Gris Riesling Roussanne Sauvignon Blanc Semillon Vidal Viognier White Wine
91
1998 2000 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 Chardonnay Chenin Blanc Fume Blanc Gewurztraminer Gruner Veltliner NV Pinot Blanc Pinot Gris Riesling Roussanne Sauvignon Blanc Semillon Viognier White Wine
92
1999 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 Chardonnay Chenin Blanc Fume Blanc Gewurztraminer Gruner Veltliner Moscato NV Pinot Blanc Pinot Gris Riesling Roussanne Sauvignon Blanc Semillon Vidal Viognier White Wine
93
1995 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 Chardonnay Chenin Blanc Fume Blanc Gewurztraminer Gruner Veltliner NV Pinot Gris Riesling Roussanne Sauvignon Blanc Semillon Viognier White Wine
94
2002 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 Chardonnay Chenin Blanc Fume Blanc Gewurztraminer NV Riesling Sauvignon Blanc Semillon Vidal Viognier White Wine

Figure 4.16: White wine viewers for points 87-94

Conclusions I was able to remove many location errors in the data, as well as introduce new subclasses and thus many-valued context table attributes into the data which helped in the separation of data enabling us to explore both red and white wine varietals easily.

As such, our data imputation process was successful in enriching the data by correcting errors and inconsistencies, as well as adding in explicit subclass attributes for many-valued context table attributes that were previously only explicit. This was all achieved using only the controlled vocabulary attributes, all those not the review in the above examples and case study.

The primary issue I encountered was that the approach simply could not fix heavily missing data. That is if there is no driving attribute to imply the existence of another within the many-valued context table's controlled vocabulary attributes our approach and algorithm is unable to accomplish much.

There is still an issue when I am missing many attributes. One approach one may use is to mine the free-text available in the review attribute.

4.7 Free Text Data-Imputation

Free text data-imputation in this thesis refers to the mining of a body of free text and linking it to a set of axioms for use in aiding inference tasks. The use of free text is often coupled with a controlled vocabulary of some kind, for instance, in the form of attributes of a JSON object in the web space. Often this body of free text will contain information that is not contained in the controlled vocabulary.

In this thesis, I wished to mine this data in the case that even after ordinary data imputation had been performed, there were still many items with null or empty attributes.

Approach

The approach employed was to perform data-imputation as above. Then perform a second pass on the data to specifically checked for empty entries, for those many-valued context table items I performed key-phrase extraction on the free-text and then attempted to match the extracted phrases to the available axioms.

That is, if for some wine W_1, the key-phrases extracted, in the same way as was described and demonstrated in Section 2.4, contained the key-phrases $\{\mathsf{California}, \mathsf{Red\ Wine}\}$, I perform a lookup of the individual California, check if there are any available axioms available for it or its independent individuals and map successfully based off of that. In this case, for California to be used by the algorithm W_1 must have an object property such as hasWinery linking it to Winery individual with no isLocatedIn axiom. Then I can link California to the independent Winery individual, which can then be mined in the inference process.

For items directly linked to W_1, the process is somewhat clearer as there is an axiom with domain Wine, that matches Red Wine, that is $\mathsf{W}_1 : \mathsf{RedWine}$.

The reason for only using this on missing data items is that the approach is inherently flawed, as it relies on the correctness of the key-phrases being extracted, and creates an implicit link that is not stated in our ontology between the individual and the key phrases extracted from the free text, which in turn can lead to inconsistencies.

Dangers of Approach

I found that the key phrase extraction was inherently flawed as it does not take the subject of the sentence into account upon extraction. This means that all the key phrases are assumed to be directly related to the root dependent individual, or its independent individuals, depending on the available axioms. This approach is unsafe and may lead to information that is logically consistent in the ontology, as I never add axioms that lead to inconsistencies. Still, the information is, however, incorrect.

Consider the following text excerpt for some W_1, "The Red Wine W_1 has a far more pleasant rounded nose than that of its sister wine from California W_2."

The key phrase extraction will extract the following key-phrases:

$$
\begin{aligned}
KF :=\{ & \\
& \mathsf{Red\ Wine} \quad \mathsf{W}_1, \mathsf{Red\ Wine,W}_1, \\
& \mathsf{pleasant\ rounded\ nose,\ rounded\ nose,} \\
& \mathsf{sister\ wine,\ California} \quad \mathsf{W}_2 \\
& \mathsf{California}, \mathsf{W}_2 \\
& \}
\end{aligned}
$$

Figure 4.17: Example key phrases for some W_1 review text

Here in Figure 4.17 we can see that even though the key phrase California is intrinsically linked to the wine W_2, as California is in a review that is attached to the many-valued context tale entry W_1, the algorithm will link it to W_1 if it leads to no logical inconsistencies.

This is unlike if the above text excerpt were to say: "Mont Destin's Red Wine W_1 has a far more pleasant rounded nose than that of its sister wine from California W_2." The key phrases shown in Figure 4.18 would be extracted.

KF :={
Mont Destin
Red Wine W_1, Red Wine,W_1,
pleasant rounded nose, rounded nose,
sister wine, California W_2
California, W_2
}

Figure 4.18: Example key phrases for some W_1 review text

As in this case both Mont Destin and California would be extracted, the {W_1,Mont Destin}:hasWinery axiom would be added first, followed by {Mont Destin,California}:isLocatedIn axiom, but as Mont Destin already has an isLocatedIn axiom for Stellenbosch which is in turn located in Cape Winelands which is a distinct region individual from California, it leads to a logical inconsistency and the second axiom is not added to the ontology.

The issue here is as this approach is being used specifically in the case of lacking data, due to this innate ability to produce so many false positives, which may or may not fail a consistency check. The key phrase extraction process being fairly computationally expensive already, is that there will often not be enough data present to cause the match to fail a consistency check. Then I end up in a situation where I have begun to compromise the underlying ontology as a source of truth, which may lead to invalid data being added to the base dataset through the imputation process.

As such, this was added to the data-imputation subsystem but disabled for the case study as it may in its current form, lead to unreliable results.

Possible Solutions

The most obvious solution to this problem is to improve the key-phrase extraction to the point that not only does it only extract phrases linked to the dependent base individual and its axiomatically linked independent individuals, but explicitly links the extracted phrase to the base individual and the linked independent individuals.

That is, in the latter of the above examples, Mont Destin would be linked to W_1 and California would be explicitly linked to W_2.

This level of NLP processing was left out as it was not in the scope of this thesis but does present opportunity for future work.

4.8 Summary

In this chapter, I have outlined and described the problems introduced by implicit information in ConceptCloud, namely that ConceptCloud has no good way to allow the user to navigate by implicit information, particularly when this is paired with incompleteness in the data.

I proposed a description logic based approach making use of the *OWL2* language to pair highly structured data in the form of an ontology with the existing semi-structured data present in ConceptCloud's formal concept lattice and many-valued context table.

I presented a small case study showing the viability of the approach. I discussed the changes seen when compared to the case study performed earlier on the same initial data set.

Finally, I covered free-text data imputation and outlined the initial approach used and the pitfalls thereof, followed by a short outline of a possible solution to the problems encountered.

Chapter 5

Conclusions

5.1 Summary

This thesis covered improvements and extensions to the ConceptCloud formal concept analysis based semi-structured data exploration tool.

I established the existing architecture and outlined the shortcomings thereof. I then outlined the scope of this thesis addressing the original ConceptCloud implementation's poor scaling, inadequate support for specialised datasets, and its inability to supplement the existing dataset.

Additionally, a chapter outlining all the required formalisms was provided.

I summarise each chapter below.

Background

The background chapter provides a background into formal concept analysis and its application to ConceptCloud. The background provided by way of example. Wine reviews in a binary context table are used and built upon to form many-valued contexts, following which I introduce the idea of a formal concept lattice. The lattice's intrinsic navigational support is explained. Various formal concept analysis based data exploration tools are introduced, and a brief overview and application to a domain dataset to be used in the rest of the thesis is provided.

Following this ConceptCloud is introduced and brief overviews of the tool's functionality, interface and architecture are provided.

I then cover the natural language processing techniques used in ConceptCloud.

Finally, the chapter concludes with an introduction to description logics and ontologies. I primarily focused on the description logic concept language $\mathcal{SROIQ}$ as it is closely related to the *OWL2* language and its API, which formed the basis of the data-imputation implementation.

Once again by way of wine review running example I introduce the concepts of a $\mathcal{SROIQ}$ ontology and its various components, these were used to outline a wine domain and ontology later used in Chapter 4. I introduce description logic semantics, once again continuing with the running wine domain example and provide an interpretation of this domain from which I introduce logical consequence.

Server Based Architecture

The server-based architecture chapter briefly outlines the maintenance and refactoring performed, followed by an introduction to the updated scalable ConceptCloud architecture. The extensibility and architectural support for specialised viewers making use of ConceptCloud's explorative search process as a service is explained. This is followed by an explanation of changes made to the data representation to aid work with larger datasets.

A small set of experiments displaying the speed-up provided by the new architecture are shown, followed by a case study on a wine review dataset.

Maps Extension

The maps extension section of the Server-Based Architecture chapter describes the implementation of a specialised map-based viewer for ConceptCloud, as well as the implementation of specialised support for biclustering the data within the map visualisation. This is followed by a large case study on a South African Crime dataset illustrating the usefulness of the map-based viewer and data biclustering.

Data Imputation

The data imputation chapter outlines the need to supplement the existing ConceptCloud data. A general overview of the approach is provided, followed by quantification of the goals of the chapter. I then provide details of the specific approach and demonstrate the algorithm used by way of a running example. The algorithm is broken down into three parts, each of which I go through describing how the algorithm processes an example wine review data item. I follow this with an overview of the architecture for the ConceptCloud data imputation component.

I then give a case study based on the previous wine review case study in chapter 3. The case study demonstrates the usefulness of the data imputation component in ConceptCloud.

Finally, I describe the free text data imputation component and the shortcomings thereof.

5.2 Future work

Description Logic Rolificaton

Description logic rolification is a transformation from a concept to a new role, such that the rolification of a concept Wine is the new role R_{Wine} defined by the axiom $\mathsf{Wine} \equiv \exists R_{\mathsf{Wine}}.\mathsf{Self}$.

Consider the rule related to our previous ontology.

$$\begin{gathered}\mathsf{Wine(x)} \wedge \mathsf{hasWinery(x,y)} \wedge \mathsf{Winery(y)} \wedge \\ \mathsf{isLocatedIn(y,SouthAfrica)} \rightarrow \mathsf{SouthAfricanWhiteWine(x)}\end{gathered}$$

That is for some wine x which has a winery y, if that winery is located in South Africa, then x is a South African Wine.

Using rolification I could now translate this rule to something which I can then translate to *OWL2* syntax.

$$\mathsf{R}_{\exists\mathsf{hasWinery}} \circ \mathsf{R}_{\exists\mathsf{isLocatedIn.Country}} \circ \mathsf{R}_{\mathsf{Country(SouthAfrica)}} \sqsubseteq \mathsf{SouthAfricanWine}$$

Which I can express as an axiom in *OWL2*, one would only need to add these class axioms for each possible country individual in the ontology.

Each of which requires the *OWL2* rolification axioms for the involved concepts, and a property chain axiom for the rule.

Since the rolifcation axioms generally have a fixed format, and our inference would generate a class axiom, which I already mine, I could mine all this information to enrich the existing ConceptCloud dataset.

I could then easily add to this rolification to express the previously desired South African White Wine attribute, which would allow for further segmentation of the ConceptCloud dataset, allowing for new levels of comparison between subsets of the total dataset.

NLP Improvements

Due to the pivotal nature of NLP within ConceptCloud, both for general free-text phrase extraction, and free-text phrase extraction for the purpose of data-imputation, any improvements to the overall phrase extraction and natural language processing component should lead to significant improvements in the quality of data the ConceptCloud system will be able to produce for itself.

Key-phrase extraction linking phrases to their natural language subjects, as described in the possible solutions subsection of section 4.7, would allow for major strides to be made towards reliable free text data-imputation.

Ontology fine-tuning

Fine-tuning the ontology towards the source dataset would lead to the algorithm being able to mine and correct far more of the base dataset.

What is meant by this is the language of the ontology needs to be consistent with the language of the semi-structured dataset, adding equivalence axioms for terms to act as a thesaurus would greatly aid this process.

I could also use a finely tuned ontology with explicit geo-location aspects to enrich a wine dataset with explicit geo-location properties, i.e. GPS coordinates based on the nearby locations or towns. This would, in turn, enable map-based visualisation of the data set and easy use of the biclustering aggregation.

Algorithm Improvements

The algorithm and implementation of the data-imputation subsystem were built as a proof of concept. As such, many opportunities exist to improve performance. The only performance improvements added in the scope of this thesis were to establish a base level of usability.

5.3 Conclusion

The main goal of this thesis was to enrich the explorative search process in the ConceptCloud tool, by

1. improving the tool's scalability and flexibility,
2. improving the support of visualisation for specific datasets, and
3. improving ConceptCloud's ability to handle less well-curated semi-structured datasets.

Scalability and flexibility

One of the main problems with the ConceptCloud tool is that it generally scaled poorly. Specialised versions of the tool existed but these were highly optimised towards specific datasets.

I generalised the scaling approaches used were able to show a generalised architecture which showed an order of magnitude speed-up in rendering operations. These results were presented in *Proceedings of CARI 2018 (African Conference on Research in Computer Science and Applied Mathematics* [15], *Scaling the ConceptCloud browser to large semi-structured data sets*[3].

To further illustrate the scaling improvements. I conducted a case study on a wine review dataset ConceptCloud was previously unable to process.

Visualisation Support

ConceptCloud's default tag-cloud visualisation while being flexible and intuitive enough for general datasets poorly visualises specialised data. To address this, I added support to ConceptCloud for specialised viewers. To display these architectural changes, I collaborated with Tiaan Du-Toit to implement a map-based visualisation. Alongside this map-based visualisation, I added support for data-biclustering, pairs (A, B) of inclusion maximal sets of objects and attributes, such that almost all objects in A have almost all attributes in B.

This viewer and approach were successfully utilised to conduct a case study on South African crimes. The dataset contained 2.2million crimes and 250 000 were displayed in the map viewer.

This case study once again displayed the flexibility of the ConceptCloud tool as well as its scalability. The specialised map viewer and biclustering approach allowed for a new and alternate way to group data and perform an explorative search.

Data-Imputation

The main goal of the data-imputation component of this thesis was to enrich the explorative search process by improving the underlying data.

To accomplish this goal I wished to take advantage of domain ontologies to enrich the underlying semi-structured data to make implicit data and correct inconsistencies, allowing for better data exploration.

In the case study provided, I show the improvements gained by performing the data imputation. I was able to both make implicit subclasses in the underlying data explicit and correct inconsistencies in the dataset.

The success of this approach is, however, heavily predicated by the ontology. That is, for an ontology that generates many implications based off of the mined many-valued context table data and is closely linked to the base dataset; I will have more success. This is as the process relies on natural language processing and phrase matching to link formal context attributes and objects to individuals and axioms. Following that information can only be inferred based on the axioms which are contained in the ontology. As I am only adding individuals and object property assertion axioms, I rely on the existence of pre-existing object properties, classes and subclasses to drive the inference process.

This requirement on both the natural language processing and the ontology has revealed opportunities for both future work and research. This research could explore the results of improvement to the natural language processing used and the fine-tuning of the ontology.

Final Conclusions

The work covered in this thesis achieved the goals set for it and provided insights into possible further research areas.

This work also highlighted that while theoretically the practice of reflecting formal context data into an ontology is straight forward and simple, in practice it is heavily reliant on the natural language processing component to perform the mapping between the formal context attributes and the DL vocabulary.

Improvements can be still be made to much of the ConceptCloud tool and the additions made provide the basis for these improvements. Ultimately the ConceptCloud tool is now more flexible and robust than its previous iterations.

Bibliography

[1] Simon Andrews and Constantinos Orphanides. 2010. Analysis of large data sets using formal concept lattices. (2010).

[2] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider (Eds.). 2007. *The Description Logic Handbook: Theory, Implementation and Applications* (2 ed.). Cambridge University Press.

[3] Joshua Berndt, Bernd Fischer, and Arina Britz. 2018. Scaling the ConceptCloud browser to large semi-structured data sets. (2018).

[4] Pavel Braslavski, Nikolay Karpov, Marcel Worring, Yana Volkovich, and Dmitry I. Ignatov. 2014. 8th Russian Summer School in Information Retrieval (RuSSIR 2014). *SIGIR Forum* 48, 2 (Dec. 2014), 105–110. https://doi.org/10.1145/2701583.2701598

[5] Peter Buneman. 1997. Semistructured data. In *Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*. ACM, 117–121.

[6] Rob Bygrave. 2018. Ebean ORM. https://github.com/ebean-orm/ebean. (2018). Accessed: 2018-05-20.

[7] Min Chen, Shiwen Mao, and Yunhao Liu. 2014. Big Data: A Survey. *Mobile Networks and Applications* 19, 2 (01 Apr 2014), 171–209. https://doi.org/10.1007/s11036-013-0489-0

[8] Justin Poehnelt Chris Arriola. 2018. Google Maps JavaScript API. https://developers.google.com/maps/documentation/javascript/tutorial. (2018). Accessed: 2018-05-20.

[9] Tiaan du Toit, Joshua Berndt, Katarina Britz, and Bernd Fischer. 2019. ConceptCloud 2.0: Visualisation and Exploration of Geolocation-Rich Semi-Structured Data Sets. (2019).

[10] Marcel Dunaiski, Gillian J Greene, and Bernd Fischer. 2017. Exploratory search of academic publication and citation data using interactive tag cloud visualizations. *Scientometrics* 110, 3 (2017), 1539–1571.

[11] Ecma International. 2013. The JSON Data Interchange Format. Standard ECMA-404. (Oct. 2013).

[12] Erich Gamma and Kent Beck. 2006. JUnit. (2006).

[13] Bernhard Ganter and Rudolf Wille. 2012. *Formal concept analysis: mathematical foundations.* Springer Science & Business Media.

[14] Janet Gie and Craig Haskins. 2007. CRIME IN CAPE TOWN: 2001-2006. (2007).

[15] Nabil Gmati, Eric Badouel, and Bruce Watson (Eds.). 2018. *Proceedings of CARI 2018 (African Conference on Research in Computer Science and Applied Mathematics).* Stellenbosch, South Africa. `https://hal.inria.fr/hal-01881376`

[16] Gillian J Greene, Marvin Esterhuizen, and Bernd Fischer. 2017. Visualizing and exploring software version control repositories using interactive tag clouds over formal concept lattices. *Information and Software Technology* 87 (2017), 223–241.

[17] Gillian J Greene and Bernd Fischer. 2014. Conceptcloud: A tagcloud browser for software archives. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering.* ACM, 759–762.

[18] Gillian J Greene and Bernd Fischer. 2015. Interactive tag cloud visualization of software version control repositories. In *Software Visualization (VISSOFT), 2015 IEEE 3rd Working Conference on.* IEEE, 56–65.

[19] Gillian J. Greene and Bernd Fischer. 2016. Single-Focus Broadening Navigation in Concept Lattices. In *CDUD@CLA.*

[20] Ian Horrocks, Oliver Kutz, and Ulrike Sattler. 2006. The Even More Irresistible SROIQ. *Kr* 6 (2006), 57–67.

[21] Mehdi Kaytoue, Sergei O Kuznetsov, Juraj Macko, Wagner Meira, and Amedeo Napoli. 2011. Mining biclusters of similar values with triadic concept analysis. *arXiv preprint arXiv:1111.3270* (2011).

[22] Slava Kisilevich, Florian Mansmann, and Daniel Keim. 2010. P-DBSCAN: A Density Based Clustering Algorithm for Exploration and Analysis of Attractive Areas Using Collections of Geo-tagged Photos. In *Proceedings of the 1st International Conference and Exhibition on Computing for Geospatial Research &#38; Application (COM.Geo '10).* ACM, New York, NY, USA, Article 38, 4 pages. `https://doi.org/10.1145/1823854.1823897`

[23] Adila Krisnadhi, Frederick Maier, and Pascal Hitzler. 2011. OWL and Rules. In *Reasoning Web International Summer School*. Springer, 382–415.

[24] Markus Krötzsch, Frantisek Simancik, and Ian Horrocks. 2012. A description logic primer. *arXiv preprint arXiv:1201.4089* (2012).

[25] Sergei O Kuznetsov. 2004. On the Intractability of Computing the Duquenne-Guigues Base. *J. UCS* 10, 8 (2004), 927–933.

[26] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*. 55–60. `http://www.aclweb.org/anthology/P/P14/P14-5010`

[27] Boris Motik, Peter F Patel-Schneider, Bijan Parsia, Conrad Bock, Achille Fokoue, Peter Haase, Rinke Hoekstra, Ian Horrocks, Alan Ruttenberg, Uli Sattler, et al. 2009. OWL 2 web ontology language: Structural specification and functional-style syntax. *W3C recommendation* 27, 65 (2009), 159.

[28] Gareth Newham. 2008. RECLAIMING OUR HOMES? Tackling residential robbery in Gauteng. *SA Crime Quarterly* 2008, 23 (2008), 7–12.

[29] Justin Poehnelt. 2018. Marker Clustering Library. `https://github.com/googlemaps/v3-utility-library/tree/master/markerclusterer`. (2018). Accessed: 2018-05-20.

[30] Sebastian Rudolph. 2011. Foundations of description logics. In *Reasoning Web International Summer School*. Springer.

[31] Guus Schreiber. 2008. Knowledge engineering. *Foundations of Artificial Intelligence* 3 (2008), 929–946.

[32] Rob Shearer, Boris Motik, and Ian Horrocks. 2008. HermiT: A Highly-Efficient OWL Reasoner.. In *OWLED*, Vol. 432. 91.

[33] Steffen Staab and Rudi Studer. 2009. *Handbook on Ontologies* (2nd ed.). Springer Publishing Company, Incorporated.

[34] Col S. Weber. 2018. SAPS Common Law Offences-Definitions. `https://www.saps.gov.za/faqdetail.php?fid=9`. (2018). Accessed: 2018-06-29.

[35] Serhiy Yevtushenko, Julian TANE, Tim B KAISER, Sergei OBIEDKOV, Joachim HERETH, and Heiko REPPE. 2006. ConExp-The Concept Explorer. (2006).

[36] Mohammed J Zaki, Ching-Jui Hsiao, et al. 1999. *CHARM: An efficient algorithm for closed association rule mining.* Technical Report. Citeseer.

Appendix A

A.1 Setup

The basic setup for the system is as follows:

1. Create postgres ConceptCloud user "conceptcloud" with password "cc".
2. Create postgres database JsonDB.
3. Modify datasetconfig.json in the project root directory to apply to the dataset you wish to process, this dataset should also be placed into the project root directory.
4. Execute ModelGenerator.py, note this reads datasetconfig.json to get the name of the dataset, so it is imperative that datasetconfig.json is updated prior to the execution of this python script. The python script creates a java model from the dataset to be used in querying the database.
5. Finally the user will need to update both TagDisplay.java, getTableData() to reflect the table data the user wishes to be displayed in the context table. Additionally the headings should be updated in viewer.scala.html, this will be in the tableview table.

After this execution of the ConceptCloud application with activator run will cause the ConceptCloud system to generate the database tables required, populate them, generate the underlying concept lattice, and generate and display the tagclouds for the underlying lattice.

It should be noted that for any dataset contained in a properly formed *JSON* file, see Section A.1 for more, this approach will work. The system will when used correctly generate any required database and lattice, and then display the top 5000 most relevant tags within a tagcloud.

System Requirements

For a system to run ConceptCloud it is recommended that the user use a unix based operating system, windows support exists, but using the Play! framework with sbt on windows requires additional setup. The system additionally requires the following:

1. Java 8 runtime environment.

2. posgresql, at the time of writing the system was stable on postgresql 10.3

Dataset Constraints

The ConceptCloud system expects the input dataset to be well formed. What this means is that the input json file conforms to existing json syntax, as seen at ¡insert link to w3schools json syntax here¿. And that the file contains data in the following format:

In this way we can have a JSON object array containing an arbitrary amount of objects, each containing an arbitrary list of attributes, so long as the list of attributes is consistent on all objects, if the object has no value for a particular attribute, as with ObjectTwo AttributeThreeName, it is omitted by leaving the value empty with empty quotation marks as above.

Configuration File

The JSON configuration file, found in the `conf/datasetconfig.json` directory of the project root, denotes the metadata relating to the input dataset, in the project root, a sample configuration file is shown below: Note that the names of all fields should be case sensitive. The attributes are the following:

1. **datasetFileName** The title of the dataset in the project root to be used.

2. **ContextTableColumns** The attribute / category titles of the attributes to be used in the context table below the tagcloud viewers. See Section 2.3 item 2.3.

3. **KeyPhraseExtractionFields** The attribute / category titles of the attributes on which keyphrase extraction should be performed, performing keyphrase extraction will create additional new fields with the titles of `attributeName_phrases`, in the above example these new categories would be called `blurb_phrases` and `additional_info_phrases`.

4. **ExtractKeyphrases** A Yes / No flag of whether to perform keyphrase extraction on the aforementioned keyphrase extraction fields.