

<http://researchcommons.waikato.ac.nz/>

## Research Commons at the University of Waikato

### Copyright Statement:

The digital copy of this thesis is protected by the Copyright Act 1994 (New Zealand).

The thesis may be consulted by you, provided you comply with the provisions of the Act and the following conditions of use:

- Any use you make of these documents or images must be for research or private study purposes only, and you may not make them available to any other person.
- Authors control the copyright of their thesis. You will recognise the author's right to be identified as the author of the thesis, and due acknowledgement will be made to the author where appropriate.
- You will obtain the author's permission before publishing any material from the thesis.

# **Adaptive User Interfaces for the Semantic Web**

A thesis

submitted **in fulfilment**

of the requirements for the degree

of

**Master of Philosophy in Computer Science**

at

**The University of Waikato**

by

**Emmanuel King Turner**



THE UNIVERSITY OF  
**WAIKATO**  
*Te Whare Wānanga o Waikato*

2018

## ABSTRACT

---

The Semantic Web aims to democratise information by making information open, shareable and recombining. However, the amount of Semantic Web data may be so large that simple listings are not useful, and the variety of data means that templates will be fragile. An adaptive user interface can modify a display based on user preferences that are learned from user actions. The thesis is that a Semantic Web browser that groups and orders the data based on knowledge of user preferences has speed and accuracy that exceed those of alphabetical ordering. The investigation comprises three user studies.

An adaptive user interface has overheads for learning and recording user preferences. The first user study indicates that user preferences are diverse enough to justify the overhead of an adaptive user interface.

The research then proposes two adaptive user interface methods; ListAlg has a top-down user model, and GPRank has a bottom-up user model. The second user study demonstrates that both are capable of learning user preferences for grouping and ordering Semantic Web data. GPRank is both more accurate and preferred by users in a blind selection.

In tasks involving selecting the answer to a question, some participants are faster with GPRank, and the users that are faster with Alphabetical ordering are not much slower when using GPRank. There is no difference in the accuracy of users performing information retrieval tasks when using GPRank or Alphabetical ordering.

# TABLE OF CONTENTS

<b>ABSTRACT</b>	<b>II</b>
<b>TABLE OF CONTENTS</b>	<b>III</b>
<b>LIST OF FIGURES</b>	<b>VII</b>
<b>LIST OF TABLES</b>	<b>X</b>
<b>LIST OF EQUATIONS</b>	<b>XII</b>
<b>CHAPTER ONE INTRODUCTION</b>	<b>1</b>
1.1 MOTIVATION	2
1.2 PROBLEM STATEMENT AND APPROACH	3
1.3 RESEARCH QUESTIONS	4
1.4 THESIS STRUCTURE	5
<b>CHAPTER TWO BACKGROUND</b>	<b>8</b>
2.1 HISTORY OF THE SEMANTIC WEB	8
2.2 MACHINE UNDERSTANDABILITY	9
2.3 THE RESOURCE DESCRIPTION FRAMEWORK	11
2.4 ONTOLOGY	15
2.5 ADAPTIVE AND ADAPTABLE USER INTERFACES	17
2.6 SUMMARY	19

<b>CHAPTER THREE RELATED WORK</b>	<b>20</b>
<b>3.1 LEXICAL APPROACHES</b>	<b>21</b>
3.1.1 DOCUMENT ORDERING	22
3.1.2 ALPHABETICAL ORDERING	26
3.1.3 LEXICAL MATCHING	27
<b>3.2 SEMANTIC APPROACHES</b>	<b>28</b>
3.2.1 PREDICATE MATCHING	28
3.2.2 TYPE MATCHING	30
3.2.3 ONTOLOGICAL REASONING	33
<b>3.3 USER PREFERENCE APPROACHES</b>	<b>34</b>
<b>3.4 SUMMARY</b>	<b>35</b>
<b>CHAPTER FOUR USER STUDY I: DO USERS AGREE ON THE RELATEDNESS OF TRIPLES?</b>	<b>38</b>
<b>4.1 HYPOTHESES</b>	<b>39</b>
<b>4.2 METHODOLOGY</b>	<b>39</b>
4.2.1 MEASURING AGREEMENT	41
4.2.2 STUDY LOCATION AND TIME.	42
4.2.3 PARTICIPANT DEMOGRAPHICS	42
<b>4.3 RESULTS</b>	<b>44</b>
<b>4.4 DISCUSSION</b>	<b>45</b>
<b>4.5 SUMMARY</b>	<b>46</b>
<b>CHAPTER FIVE USER PREFERENCES FOR GROUPING AND ORDERING</b>	<b>47</b>
<b>5.1 NONLEARNER</b>	<b>47</b>
5.1.1 FORMING DISPLAYS FROM NONLEARNER	48

<b>5.2 LISTALG</b>	<b>51</b>
5.2.1 USER MODEL	51
5.2.2 FORMING DISPLAYS FROM LISTALG	52
5.2.3 INCORPORATING NEW USER PREFERENCES	52
5.2.4 DISCUSSION	59
<b>5.3 GROUPED PAIRWISE RANKING (GPRANK)</b>	<b>61</b>
5.3.1 USER MODEL	62
5.3.2 FORMING DISPLAYS WITH GPRANK	68
5.3.3 INCORPORATING NEW USER PREFERENCES INTO THE GPRANK USER MODEL	70
5.3.4 DISCUSSION	71
<b>5.4 COMPARISON BETWEEN LISTALG AND GPRANK</b>	<b>74</b>
5.4.1 FUTURE RESEARCH: A HYBRID USER MODEL	75
<b>5.5 SUMMARY</b>	<b>76</b>

## **CHAPTER SIX USER STUDY II: LEARNING USER PREFERENCES FOR GROUPING AND ORDERING**

	<b><u>77</u></b>
<b>6.1 HYPOTHESES</b>	<b>77</b>
<b>6.2 METHODOLOGY</b>	<b>79</b>
6.2.1 THE THREE METHODS	80
6.2.2 DATASET DESIGN	81
6.2.3 MEASURING DIFFERENCES BETWEEN ALTERNATIVE GROUPING AND ORDERING	82
6.2.4 TEST SEQUENCE	84
6.2.5 STUDY LOCATION AND TIME	91
<b>6.3 RESULTS</b>	<b>91</b>
6.3.1 PERFORMANCE OF NONLEARNER	94
6.3.2 LISTALG AND GPRANK LEARNING	95

6.3.3	LISTALG AND GPRANK PERFORMANCE VERSUS NONLEARNER	99
6.4	CONCLUSIONS AND SUMMARY	102
<b><u>CHAPTER SEVEN USER STUDY III: GPRANK VERSUS ALPHABETICAL ORDERING FOR USER</u></b>		
<b><u>SPEED AND ACCURACY IN INFORMATION RETRIEVAL TASKS</u></b>		<b>105</b>
7.1	HYPOTHESES	105
7.2	METHODOLOGY	106
7.2.1	EXPERIMENT DESIGN CONSIDERATIONS	107
7.2.2	EXPERIMENT IMPLEMENTATION	107
7.3	RESULTS	116
7.4	FINDINGS	119
7.4.1	GPRANK MAY BE MORE ACCURATE FOR SOME PARTICIPANTS	121
7.4.2	OTHER FINDINGS	122
7.5	CONCLUSIONS	125
<b><u>CHAPTER EIGHT SUMMARY AND CONCLUSIONS</u></b>		<b>127</b>
8.1	CONCLUSIONS	128
8.2	RECOMMENDATIONS FOR FUTURE RESEARCH	130
<b><u>REFERENCES</u></b>		<b>133</b>
<b><u>APPENDICES</u></b>		<b>1</b>
APPENDIX ONE: DEMOGRAPHIC QUESTIONNAIRE		2
APPENDIX TWO: PAIRS OF TRIPLES USED TO FORM QUESTIONS IN THE FIRST USER STUDY (CHAPTER FOUR)		2

## LIST OF FIGURES

---

Figure 2.1: An RDF Graph with two nodes (Subject and Object) and Predicate connecting them .....	11
Figure 3.1: Tim Berners-Lee’s FOAF file in Brownsauce. Triples are shown in Document Order. ....	23
Figure 3.2: Berlin from GeoNames in Brownsauce. Triples are shown in Document Order. ....	23
Figure 3.3: Tim Berners-Lee’s FOAF file in the Quick and Dirty RDF Browse. Triples are shown in Document Order. ....	24
Figure 3.4: Berlin from GeoNames shown in the Quick and Dirty RDF Browser. Triples are shown in Document Order.....	24
Figure 3.5: Tim Berners-Lee’s FOAF file in Disco. Data is shown in document order. [Black zig-zags denote edits that omit spaces so that the screenshot shows relevant features] .....	25
Figure 3.6: Berlin from GeoNames in Disco. Triples are shown in document order. [Black zig-zag denote an edit that omits space so that the screenshot shows relevant features].....	26
Figure 3.7: Berlin for DBpedia. Triples are shown in alphabetical order by predicate. ....	27
Figure 3.8: A screenshot of Falcons Concept Search (Cheng & Qu, 2009) .....	28
Figure 3.9: Screenshot of a Semantic Web browser that automatically groups triples using semantic predicate matching based on strings (Seeliger & Paulheim, 2012) .....	29
Figure 3.10: Exhibit showing keyword search, facets, timeline, and map. (Screenshot of <a href="http://www.simile-widgets.org/exhibit/examples/presidents/presidents.html">http://www.simile-widgets.org/exhibit/examples/presidents/presidents.html</a> , taken 19 Jan 2017) .....	31
Figure 3.11: Exhibit showing a timeline overview (left) and a detail view of a person (right). ..	32
Figure 3.12: Marbles’ alternative data presentations (from left to right) Full view, Summary view, Photo view .....	33



Figure 4.1: Screenshot for the study where the user has given a rating of 1 .....	40
Figure 5.1: User Model for ListAlg.....	51
Figure 6.2: Drag and drop interface for a user to express the grouping and ordering preferences .....	85
Figure 6.3: Screen for user to select one of two outputs .....	86
Figure 6.4: Screen for Selecting Dataset .....	87
Figure 6.5: Grouping and Ordering Screen showing grouping and ordering by NonLearner .....	88
Figure 6.6: Grouping and Ordering Screen showing data as grouped and ordered by a participant.....	88
Figure 6.7: Participant selects the best Grouping/Ordering.....	89
Figure 6.8: Participant can fine tune the grouping and ordering .....	90
Figure 6.9: Algorithm Accuracy over time with +/- One Standard Deviation .....	96
Figure 6.10: Plot showing the standard deviation of algorithm accuracy reduces as more user preference data is available .....	97
Figure 6.11: Chart of Change in Participant Drag-and-Drop Operations Over Time .....	98
Figure 6.12: User Choice of Algorithm by Sequence in the second user study .....	99
Figure 6.13: Trend of User Operations over time compared with dataset diversity in the second user study.....	101
Figure 6.14: Percentage predictive disadvantage compared with dataset diversity.....	102
Figure 7.1: Participant selects a topic domain.....	110
Figure 7.2: Training GPRank with User's preferences for grouping and ordering .....	111
Figure 7.3: Question screens with advance indication of the display format for the answering screen (Alphabetical ordering and GPRank) .....	112
Figure 7.4: Answering screen for ABC Ordering .....	113
Figure 7.5: Answering screen for GPRank format.....	113

Figure 7.6: After answering from a GPRank formatted screen, the participant is given the opportunity to adjust their group-order preferences. ....	114
Figure 7.7: Chart plotting normalised speed advantage by Algorithm and Accuracy .....	120
Figure 7.8: Chart plotting normalised speed and accuracy advantage. p-values <0.1 and >0.9. Accuracy outlier removed.....	121
Figure 7.9: Response Times vs Order of Answering by Algorithm.....	124

## LIST OF TABLES

---

Table 4.1: Interpreting inter-rater reliability (Landis & Koch, 1977) .....	42
Table 4.2: Summary of Ratings .....	44
Table 5.1: ListAlg Example - Beginning .....	54
Table 5.2: ListAlg Example - Predicate matches in old and candidate user models .....	54
Table 5.3: ListAlg Example - Mismatch between predicate in old and candidate user models ..	55
Table 5.4: ListAlg Example - Another predicate match .....	55
Table 5.5: ListAlg Example - Another predicate mismatch .....	56
Table 5.6: ListAlg Example - Candidate user model group exhausted .....	56
Table 5.7: ListAlg Example - Pausing for current user model items already in the new user model .....	57
Table 5.8: ListAlg Example - Another candidate user model group exhausted .....	58
Table 5.9: ListAlg Example - Implying an empty group in the current user model .....	58
Table 5.10: ListAlg Example - Line by line .....	59
Table 6.1: Dataset diversity in the second user study .....	82
Table 6.2: Mean and Standard Deviation of Normalised Kendall Tau Closeness by algorithm and sequence for the second user study .....	91
Table 6.3: Mean Normalised Kendall Tau Closeness by Algorithm, Dataset, and Sequence for the second user study .....	92
Table 6.4: Mean count of drag and drop operations by dataset and sequence in the second user study .....	93
Table 6.5: Percentage of times participant choose each algorithm by algorithm and sequence in the second user study .....	94
Table 6.6: Predictive Disadvantage for Algorithm and Sequence in the second user study .....	94

Table 6.7: Quartiles for the NonLearner's Normalised Kendall-Tau Closeness in the second user study.....	95
Table 7.1: Dataset Diversity for the third user study .....	108
Table 7.2: Easy/Hard Question examples for the third user study .....	109
Table 7.3: Time to answer question by Participant and Algorithm in the third user study.....	117
Table 7.4: Accuracy by Participant and Algorithm in the third user study .....	118
Table 7.5: Correlation between Dataset Diversity and answering times in the third user study .....	123
Table 7.6: The effect of Easy/Hard question on Answer Time and Accuracy by Algorithm in the third user study .....	123
Table 7.7: Correlation [-1,+1] of Algorithm Accuracy to Cumulative Questions Answered by Algorithm in the third user study.....	125

## LIST OF EQUATIONS

---

Equation 5.1: Dice Coefficient for lists .....	49
Equation 5.2: Example of Dice Coefficient for the strings “firstName” and “familyName” .....	49
Equation 5.3: NonLearner’s hierarchical cluster distance metric extends Dice Coefficient .....	50
Equation 5.4: Upper bound for iterations in ListAlg .....	59
Equation 5.5: Predicate Diversity for the diversity of predicates in a collection of RDF documents .....	60
Equation 5.6: GPRankEntry tuple as used in a GPRank based user model .....	62
Equation 5.7: Example GPRankEntry .....	63
Equation 5.8: First step defining a lookup function for a particular GPRankEntry .....	64
Equation 5.9: Second step defining a lookup function for a particular GPRankEntry .....	65
Equation 5.10: Second step defining a lookup function for a particular GPRankEntry .....	65
Equation 5.11: Dot notation for named members in a GPRankEntry tuple .....	66
Equation 5.12: a and b cannot be the same in a GPRankEntry tuple .....	66
Equation 5.13: The result of reversing a and b in a GPRankEntry tuple is calculatable .....	67
Equation 5.14: Deriving a new GPRankEntry tuple from the reversal of a and b .....	67
Equation 5.15: The undefined GPRankEntry tuple .....	68
Equation 5.16: A GPRankEntry tuple with 0 confirmations is undefined .....	68
Equation 5.17: The number of user model update operations for ListAlg .....	75
Equation 5.18: The number of user model update operations for GPRank .....	75
Equation 6.1: Dataset Diversity .....	81
Equation 6.2: Normalised Kendall-Tau Distance .....	83



## CHAPTER ONE INTRODUCTION

---

The Semantic Web aims to democratise data by making data open, shareable and recombinaable (Berners-Lee, Hendler, & Lassila, 2001). To achieve these characteristics, Semantic Web data, different to data on the web, is structured. However, unlike traditional databases, the structure of Semantic Web data does not have to follow a single predefined schema. Instead, the data itself contains links to its schema. Semantic Web data is machine-understandable, and therefore data becomes easily shareable and recombinaable.

Semantic Web data is viewable by a Semantic Web browser. The shareable and recombinaable nature of Semantic Web data means that it is not possible to anticipate how users will want to visualise Semantic Web data. Also, recombination means that the potentially large amount of data items to display for any single subject may exhaust available screen space if the data is not filtered.

Taken together, we believe that approaches for building user interfaces for data with a known structure are inadequate for the Semantic Web. An alternative method is to group and order data based on user preferences using an adaptive user interface. In an adaptive user interface for the Semantic Web, the way to display the Semantic Web data would adapt to the user's preferences to suit their individual needs.

In this thesis, we propose an adaptive user interface that categorises the Semantic Web data into groups and orders data according to user preference. Placing similar data items close together may make it easier for users to locate data items, especially when the number of data items is extensive. Showing relevant data at the top is a well-established approach, e.g. for search engine results.

In summary, an approach for grouping and ordering Semantic Web data based on user preferences is developed through this thesis and then evaluated against the established alphabetical ordering.

## 1.1 MOTIVATION

This research explores an approach to grouping and ordering Semantic Web data within an adaptive user interface for a Semantic Web browser. As a motivation for this research approach, this section briefly outlines the deficiencies of current approaches to displaying Semantic Web data (further details are given in Chapter 3). The current methods for displaying Semantic Web data are alphabetical order, source order, graph-based displays, and templates.

Alphabetical ordering of Semantic Web data is used by The Disco Hyperdata Browser (Bizer & Gauß, 2007). These lists may contain a very large number of items (e.g. the DBPedia data for Germany<sup>1</sup> contains over 200 entries), in which data with similar meanings but different text (e.g. `label`, `commonName`, and `name`) have different positions in the list. Alphabetical ordering is therefore not suitable for large lists (Hu, Ma, & Chau, 1999).

Listing data according to the order that data appears in the data source (i.e. source order) is used in Brownsauce (Steer, 2003). The usefulness of this ordering then depends on the quality of ordering in the data source. However, it cannot be assumed that the data is deliberately ordered in the source, as this is not a requirement of Semantic Web standards. Even if the is deliberately ordered in the source, the ordering may not reflect user's goals. Finally, because the Semantic Web encourages recombination of data, it is unclear how to order data that is combined from multiple data sources.

Graph-based displays show Semantic Web data as a directed graph connected by arcs; it is used in LodLive (Camarda & Mazzini, 2012). This form of display can be useful for seeing how

---

<sup>1</sup> <http://dbpedia.org/Fresource/Germany>



entities relate to each other. While graph-based displays often use a large proportion of the screen, they often do not prioritise data display and do not group related data.

Rosenholtz et al. (2009) advocate for displaying data items in groups that match how users mentally group data. Additionally, users find data quicker when the data is displayed in order of relevance (e.g. as used in Google Search<sup>2</sup>). Decisions for grouping and ordering data for display are made based on prior knowledge of the user and their goals (Johnson, Johnson, & Zhang, 2005). Since Semantic Web data has dynamic structure and because Semantic Web data is recombining then it is impossible for the data publisher to predict user goals. Therefore, there may be some benefit to deferring, until runtime, display decisions that rely on knowledge of user goals.

An adaptive user interface can learn about a user's preferences for data item display at runtime. It can also adapt to different data schemas. Therefore, we believe an adaptive user interface approach is suited for displaying Semantic Web data in response to runtime conditions such as data structure and user preferences. This thesis explores the hypothesis that a Semantic Web browser that groups and orders the data based on run-time knowledge of the user and the data has usability benefits that exceed those of alphabetical ordering.

## 1.2 PROBLEM STATEMENT AND APPROACH

Semantic Web data provides its own schema definition and recombining data from multiple sources is possible. The amount of Semantic Web data may be so large such that simple listings will not display data in useful ways. Some existing Semantic Web browsers group related data together using templates, and these are documented in 3. Seeliger and Paulheim's browser (2012) uses lexical similarity to arrange triples into groups (see Section

---

<sup>2</sup> <http://www.google.com>

3.2.1.1 for a more in-depth description). The lexical similarity used in the Seeliger Browser comes from a single shared lookup source (WordNet). Users may have their own preferences for which triples should be grouped together, and those preferences may differ from the preferences expressed by WordNet or other users.

## 1.3 RESEARCH QUESTIONS

The research investigates the following hypothesis:

*A Semantic Web browser with an adaptive user interface that groups and orders data has speed and accuracy advantages in information retrieval tasks.*

In exploring the hypothesis, the following three questions are addressed.

**Question 1.** *Is there sufficient diversity in user preferences for displaying Semantic Web data to justify an adaptive user interface?*

If users have very similar preferences for grouping and ordering Semantic Web data, then it is not necessary to adapt to user preferences at run-time, and so an adaptive user interface approach will be unnecessary overhead. Conversely, if users have different preferences for how data is grouped, then this lends support to investigating adaptive approaches.

**Question 2.** *Can an adaptive interface learn user preferences for grouping and ordering displays of Semantic Web data?*

If there is sufficient diversity in user preferences (see Question 1), then it is possible an adaptive user interface will improve speed and accuracy in information retrieval tasks. Answering this question involves the exploration of existing approaches to grouping and ordering data based on user preferences and then the proposal of algorithms suited to Semantic Web data. Because the algorithms must quickly learn user preferences, then the proposed algorithms should be tested with real users.

**Question 3.** *Do users perform single screen search tasks quicker and more accurately with an adaptive user interface that groups and orders Semantic Web data or with data in alphabetical order?*

After identifying an adaptive interface model that can learn user preferences, then the next step will determine if the adaptive approach improves speed and accuracy in comparison to alphabetical order. The reason for selecting alphabetical ordering as the comparator is its familiarity to users and its widespread use.

## 1.4 THESIS STRUCTURE

This section outlines the structure of this thesis.

Chapter 2 (Background) introduces key Semantic Web and adaptive user interface concepts. The purpose of this chapter is to aid understanding of this research by providing background information about Semantic Web concepts and the architecture of adaptive user interfaces.

Chapter 3 (Related Work) examines the present state of the art for displaying data in Semantic Web browsers. The chapter is arranged into categories of the different approaches, and this demonstrates that the adaptive user interface approach proposed in this research is novel.

Chapter 4 (User study I: Do users agree on the relatedness of triples?) contributes to the first question by testing the level of agreement between participants regarding the relatedness of pairs of triples. If participants disagree about the relatedness of triples, then this indicates that users may have differing preferences for displaying Semantic Web and so an adaptive user interface may be worth investigating.

Chapter 5 (User Preferences for Grouping and Ordering) presents three methods for grouping and ordering Semantic Web data for display. Two of the methods are adaptive: they learn user preferences, and the third method does not learn, and so its purpose is as a control. This change contributes to the second research question.

Chapter 6 (User Study II: Learning User Preferences for grouping and ordering) tests the ability of the three methods proposed in the previous chapter to learn user preferences for grouping and ordering Semantic Web data. This chapter answers the first research question by directly comparing user preferences for grouping and ordering Semantic Web data. This chapter also answers the second research question by measuring the ability of the two-proposed adaptive user interface methods to learn user preferences.

Chapter 7 (User Study III: GPRank versus Alphabetical ordering for user speed and accuracy in information retrieval tasks) compares the best adaptive user interface algorithm for grouping and ordering against alphabetical ordering for information retrieval. In this user study, participants must locate the answer to a given question on a display of Semantic Web data that is either in alphabetical ordering or Grouped and Ordered according to an adaptive user

interface algorithm. The measurements taken are time to answer and answer accuracy. This chapter directly addresses the third research question.

Chapter 8 (Summary and Conclusions) summarises this thesis in the context of the research questions. The chapter discusses the limitations of the findings and highlights areas for future research.

## CHAPTER TWO BACKGROUND

---

The Semantic Web describes both a set of technologies and the decentralised platform for exchanging data in a manner that is open, shareable and recombining. This chapter introduces background information about the Semantic Web and adaptive user interfaces that are relevant to this thesis.

The chapter begins with a short history of the Semantic Web and where the term “Semantic Web” originates. Following the history is a brief discussion on the role of machine understandability in enabling data to be open, shareable and recombining. Then the chapter discusses how the Semantic Web organises information conceptually into directed graphs that are expressed as triples. The next section discusses how Semantic Web data is expressed using ontologies and why ontological information is unreliable for making decisions regarding the display of Semantic Web data. The final section introduces adaptive and adaptable user interfaces and their architecture.

### 2.1 HISTORY OF THE SEMANTIC WEB

The section is a brief history of the Semantic Web and where the term “Semantic Web” originates. The history provides context to how the Semantic Web differs from the HTML-based World Wide Web (WWW), and this provides a reference point for understanding the intent of the Semantic Web.

As early as 1994, Berners-Lee articulated a method for attaching semantics to data to provide interoperability between systems (Shadbolt, Berners-Lee, & Hall, 2006). The World Wide Web Consortium (W3C) published the first Semantic Web standards<sup>3</sup> in 1997, and these became full

---

<sup>3</sup> <http://www.w3.org/TR/WD-rdf-syntax-971002/>

W3C recommendations<sup>4</sup> by 1999 (Shadbolt et al., 2006; Steer, 2003). Soon afterwards work on servers, stores, ontology languages (e.g. RDFS), and rule-based inference languages (e.g. OWL) began.

In 2006, Berners-Lee published the linked open data blog post that describes the additional criteria that make Semantic Web data into linked open data and describes a method for HTTP-based content negotiation to provide HTML to regular web browsers and Semantic Web data to Semantic Web consumers. The linked open data standard both enabled and encouraged interlinking between Semantic Web data. By 2011 linked open data consisted of 295 data sets with 32 billion triples and 500 million links to other datasets<sup>5</sup> and this grew to over 1,000 data sets in 2014<sup>6</sup>.

Today, the Semantic Web is used in areas such as life sciences, government, and geography. W3C has a list of Semantic Web usage case studies<sup>7</sup> up to 2017. The extent of the Semantic Web can be seen in the Linking Open Data Cloud Diagram<sup>8</sup> which shows known available Semantic Web data sources as nodes and their interlinks as arcs.

## 2.2 MACHINE UNDERSTANDABILITY

The Semantic Web achieves machine understandability by attaching semantics to the data. Machine understandability also enables the Semantic Web data to be recombining because it can link data from different domains. This section discusses the meaning of the term semantics and discusses what it means to be machine understandable in the context of the Semantic Web.

---

<sup>4</sup> <http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/>

<sup>5</sup> [http://lod-cloud.net/state/state\\_2011/](http://lod-cloud.net/state/state_2011/)

<sup>6</sup> [http://lod-cloud.net/state/state\\_2014/](http://lod-cloud.net/state/state_2014/)

<sup>7</sup> <http://www.w3.org/2001/sw/sweo/public/UseCases/>

<sup>8</sup> <http://lod-cloud.net/>

The origin of the term semantics comes from linguistics and describes the relationship between symbols and meanings. The term semantics originates with Michel Breal's "*Essai de Sémantique*" (Bréal, 1904) which discusses the signification of words. The contemporary usage of the term semantics comes via Charles W. Morris' division of semiotic signs into syntax which are the symbols used, pragmatics which is the usage of signs, and semantics which relates to meaning (Morris, 1946).

In the Semantic Web, the link between a representation (i.e. data) and the real-world thing to which the URI refers (i.e. the semantics) is a URI or a literal. A *thing* is "something in the world"<sup>9</sup>, and this can be physical and abstract things. For example, the URI <http://dbpedia.org/resource/Germany> refers to the country called Germany and, Germany is a real-world *thing*. Whenever the same URI is encountered, then a machine can assume that this refers to the same thing.

The basic unit on the Semantic Web is a triple which contains a reference to a subject *thing*, an object *thing* and a predicate that expresses the nature of the relationship between the subject and object. Semantic Web data is recombinable because triples can refer to *subjects* and *objects* from different domains.

Since similar types of things may have similar data about them, for example, all countries have a capital city. The formalisation of similar data structures is called *ontology*.

---

<sup>9</sup> <https://www.w3.org/TR/rdf11-concepts/#resources-and-statements>



## 2.3 THE RESOURCE DESCRIPTION FRAMEWORK

Semantic Web's data model is called the Resource Description Framework (RDF)<sup>10</sup>. RDF can be represented conceptually as a graph-based model with arcs, the predicates, representing the relationships between things (nodes). One thing is termed the subject, and the other thing is termed the object (see Figure 2.1).

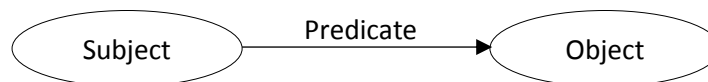


Figure 2.1: An RDF Graph with two nodes (Subject and Object) and Predicate connecting them<sup>11</sup>

Nodes in the Semantic Web can be Uniform Resource Identifier (URI), a blank node or a literal value. However, only objects can contain literal values: the subject and predicate must be URIs. URIs are unique addresses that are like a web page address, except that a Semantic Web URI can refer to any *thing* that exists in a universe of discourse.

The RDF data format represents the predicate relation between a subject and object as a three-member tuple which is called a triple. The three tuple members are (subject, predicate, object)<sup>12</sup>. For example, the following triple represents that Germany has the capital city Berlin.

```
(Germany, capital, Berlin)
```

The following triple, from the DBpedia dataset, represents that Germany has the capital city Berlin.

---

<sup>10</sup> <http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/>

<sup>11</sup> <https://www.w3.org/TR/rdf11-concepts/#fig-an-rdf-graph-with-two-nodes-subject-and-object-and-a-triple-connecting-them-predicate>

<sup>12</sup> <http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/#section-triples>

```
(http://dbpedia.org/resource/Germany,  
http://dbpedia.org/ontology/capital,  
http://dbpedia.org/resource/Berlin)
```

URIs may be shortened using a namespace abbreviation for the first part of the URI, a colon and then the unique part of the URI. Using namespaces makes reading RDF easier. The DBpedia dataset uses `dbr`:<sup>13</sup> and `dbo`:<sup>14</sup> as namespace abbreviations for things and predicates respectively. The Germany has-capital Berlin triple can be written in tuple form as:

```
(dbr:Germany, dbo:capital, dbr:Berlin)
```

For clarity, further examples of triples will dispense the conventions of tuple syntax by omitting the brackets and commas. The members of the triple are tab-delimited, and the triple itself is newline delimited. The Germany has-capital Berlin triple is written as:

```
dbr:Germany      dbo:capital      dbr:Berlin
```

Literal values are used to display labels or to state values. Labels for things are commonly expressed using the `rdfs:label` predicate from the `rdfs:` namespace. For example, the label of the thing Germany is the literal text “Germany”. This is expressed by the following triple with the literal text enclosed in double quotes.

```
dbr:Germany      rdfs:label      "Germany"
```

In practice, it is common for subject URIs to have a triple with an `rdfs:label` predicate and this triple will ordinarily have a string literal value as the object. This string literal is displayed in a user interface as the textual representation of the entity. The following, more complete, example gives the data for a Semantic Web browser to render that Germany has-capital Berlin as the string “Germany capital Berlin”. Each of the URIs within the first triple become the

---

<sup>13</sup> <http://dbpedia.org/resource/>

<sup>14</sup> <http://dbpedia.org/ontology/>

subject of a new triple with an `rdfs:label` predicate and the literal that labels that URI as the object.

<code>dbr:Germany</code>	<code>dbo:capital</code>	<code>dbr:Berlin</code>
<code>dbr:Germany</code>	<code>rdfs:label</code>	<code>"Germany"</code>
<code>dbo:capital</code>	<code>rdfs:label</code>	<code>"capital"</code>
<code>dbr:Berlin</code>	<code>rdfs:label</code>	<code>"Berlin"</code>

When triples have the same subject URI, then those triples have data about the same thing. Triples with the same predicate URI describe the same thing about the subject. Triples with shared object URIs have the same thing in common.

There is no restriction that prevents a triple from having combinations of members in common with another triple. It is common for two triples to have the same subject and predicate to show either members of an unordered list or to specify alternatives. In the following examples both Angela Merkel and Gerhard Schröder are/were leaders of Germany and, Germany is called Germany in English (en) and Deutschland German (de).

<code>dbr:Germany</code>	<code>dbo:leader</code>	<code>dbr:Angela_Merkel</code>
<code>dbr:Germany</code>	<code>dbo:leader</code>	<code>dbr:Gerhard_Schröder</code>

and

<code>dbr:Germany</code>	<code>rdfs:label</code>	<code>"Germany"@en</code>
<code>dbr:Germany</code>	<code>rdfs:label</code>	<code>"Deutschland"@de</code>

A collection of triples expresses an RDF Graph<sup>15</sup>. The set of nodes in the RDF Graph are all the subjects and objects in the collection of triples. The set of arcs in the RDF Graph are all the predicates contained in the triples that represent the graph. An RDF Graph is serialised into

---

<sup>15</sup> <https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/#section-rdf-graph>

triples, and these are transmitted, typically using HTTP/S, across the Semantic Web in an RDF Document.

If an RDF Document is focused around a single subject, then that RDF Document might not include the `rdfs:label` information for all of the predicates and objects that are contained in the RDF Document. For example, the RDF Document returned for Berlin on DBpedia<sup>16</sup> includes an `rdfs:label` triple for Berlin but no `rdfs:labels` for any other entities referenced in that RDF Document. Without caching, finding the `rdfs:label` for each entity requires another HTTP/S transaction to retrieve the RDF Document for each entity. Compared to local machine or local network access, HTTP/S transactions across the internet are comparatively slow – perhaps too slow for a user interface. Since the lookup across the internet for a single item is comparatively slow and there are many RDF Documents to retrieve, resolving `rdfs:labels` in RDF Documents is time expensive. Since `rdfs:label` lookup is a common and time expensive operation, some Semantic Web browsers (e.g. Becker & Bizer, 2009; Seeliger & Paulheim, 2012) cache `rdfs:label` data.

This research will focus on Semantic Web browsers that display data about a single subject at a time. The hypothetical use case is a user queries for information about a single thing and receives a list of possible subjects, and then the user then selects a subject from the list of search results. So, Semantic Web browsers that visualise complete RDF graphs - that may contain data about many subjects - are not addressed in this research. The search facility is outside the scope of this research.

The Semantic Web browsers focused on in this thesis show data about a single thing at a time. Therefore, the underlying RDF Graph need only contain triples that are focused around a single subject (i.e. the thing of interest). For clarity, these RDF Graphs will be referred to as Single Subject RDF Graphs (SSRGs).

---

<sup>16</sup> <http://dbpedia.org/resource/Berlin>

Linked open data are an additional set of criteria upon the Semantic Web data standards, and so the research of this thesis will also apply to linked open data with the caveat that only a single subject is shown at a time. The user interfaces of interest here focus on subjects rather than links between many subjects.

## 2.4 ONTOLOGY

A Semantic Web ontology defines the concepts and relationships used to describe a domain of knowledge<sup>17</sup>. RDF Schema<sup>18</sup> (RDFS), Web Ontology Language<sup>19</sup> (OWL) and Simple Knowledge Organization System<sup>20</sup> (SKOS) are different standards for expressing Semantic Web ontologies. A typical example of ontological linking is the “type” of a thing is represented as a triple with an `rdf:type`<sup>21</sup> predicate and an object that links to ontological information. Another form of ontological linking is found by dereferencing the predicate in a triple. For example, dereferencing the predicate `rdf:type` provides ontological information that specifies `rdf:type` is a Property defined by the RDF standards and the acceptable types of subjects are Resources (anything), and the acceptable types of objects are Classes. So, data is linked to ontology by special triples and predicates. This research focuses on predicates are the basic unit of ontology.

Just because ontology can be expressed by special triples and through predicates does not mean that this always happens in practice or that the links to ontology are accurate. There is no enforcement of ontology on the Semantic Web, and so ontological information is unreliable. For example, the following example states that Germany is an instance (`rdf:type`) of the literal string “Country”, but the ontological information for `rdf:type` states that any triple using `rdf:type` as a predicate will have an object that is an instance of the datatype

---

<sup>17</sup> <http://www.w3.org/standards/semanticweb/ontology>

<sup>18</sup> <http://www.w3.org/TR/2014/REC-rdf-schema-20140225/>

<sup>19</sup> <http://www.w3.org/TR/2012/REC-owl2-primer-20121211/>

<sup>20</sup> <http://www.w3.org/TR/2009/REC-skos-reference-20090818/>

<sup>21</sup> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>

`rdfs:Class`. However, these following two triples are valid and represent a valid RDF Graph.

Semantic Web browsers are not expected to reject these triples.

```
    dbr:Germany      rdf:type      "Country"
    rdf:type          rdfs:range    rdfs:Class
```

Ontological freedom exists because Semantic Web data can be expressed in a mixture of ontologies and there are no restrictions on creating new ontologies. Ontological freedom is an essential feature of an open Semantic Web because individual data creators can express information in ways appropriate to their context. There is no centralised social, cultural, or political viewpoint on is an ideal ontology. For example, the naming of people around the world follows many different patterns. Therefore, an ontology based on Anglosphere conventions of first name, middle names, and family name may not properly represent names from outside the Anglosphere. As a specific example, the Vivo ontology<sup>22</sup> (namespace `vivo:`) is based on Anglosphere naming and so it is not clear how to encode the Chinese name LIN Wanzeng (林宛曾). Should LIN Wanzeng be represented as:

```
<#person1>      vivo:familyName  "Lin"
<#person1>      vivo:givenName    "Wanzeng"
```

or

```
<#person1> vivo:familyName  "Lin"
<#person1> vivo:givenName    "Wan"
<#person1> vivo:middleName   "Zeng"
```

Semantic Web documents can be thought of as having an individual dynamic ontology that follows external ontologies to varying degrees. Since Semantic Web documents can contain triplets from multiple ontologies then, in effect, each Semantic Web document can potentially

---

<sup>22</sup> <http://vivoweb.org/ontology/core>

have a combined ontology that is unique to that document. The problem of multiple ontologies is compounded when combining multiple data sources, which may be missing some details. The effect of this is that there are no certain means to predict which ontologies will be present in an RDF Graph until it is retrieved and read.

When data from multiple sources are combined, especially if the sources do not have complete data about the same topics, the chances of data not conforming to ontology increases. Additionally, the ability of Semantic Web data to be recombined increases the likelihood that data source will be used by a user who has goals from the source's publisher.

When ontology is static (an equivalent example is a table in a database) then a static data presentation can adequately display that data to users. However, since ontology is dynamic, then static approaches to display Semantic Web data are not suitable. Even if the ontological information was static-enough (i.e. less dynamic), there is still no guarantee that the ontological information is reliable. Also, recombination means what users will do with data is not foreseeable, so this means that pre-determined methods for displaying data might not suit the user. Recombination also means that the Semantic Web is innumerable because it is impractical to count all the information available on the Semantic Web. The set of triples for a single subject, once data sources are combined, and inference applied, could potentially be enormous.

## 2.5 ADAPTIVE AND ADAPTABLE USER INTERFACES

An adaptive user interface (AUI) self-modifies content, structure and functionality to meet the needs of individual users (Schneider-Hufschmidt, Malinowski, & Kuhme, 1993). This section introduces the concept of an AUI, and this is later used to evaluate the adaptiveness of current Semantic Web browsers.

An AUI models a user by learning from user actions and stores what is learnt in a user model. The part of the AUI that decides how to self-modify based on the user model is called the recommender (de la Flor, 2004).

A problem in AUI user modelling is that a system begins with no information about a new user and must rapidly learn from few training cases (Langley, 1999). Langley says that the user's time is the "precious resource" and not CPU cycles. This leads Langley to suggest that an AUI that has high accuracy and learns rapidly is more competitive than an AUI that learns slower, even if the slower learner has higher accuracy.

Producing an AUI takes more effort than a non-Adaptive User Interface because of the additional work in producing the action observer, the learner, the user model and the recommender. However, Kules (2000) suggests that the benefits of an AUI are most likely to outweigh the costs when amortised over a large number of users. An AUI based Semantic Web browser would need to capture a broader audience than a domain specific application to justify the creation of that browser.

There is a related form of modifiable user interface, called an Adaptable User Interface, that the user modifies using special configuration actions (Oppermann, 1994). The key distinction between an Adaptive UI and Adaptable UI is that an Adaptable User Interface does not automatically modify itself in response to normal user interactions and an Adaptive User Interface self-modifies. More recent work (e.g. Kaufmann et al., 2007) view Adaptable versus Adaptive UIs as a continuum that measures the degree to which the UI learns from normal user interactions or uses special configuration actions. All else being equal, a more Adaptive UI is preferred over a more Adaptable UI because an Adaptive UI does not require extra user actions for modifications.



## 2.6 SUMMARY

This chapter explained Semantic Web concepts that are relevant to this thesis. Firstly, there was a brief history of the Semantic Web and the origin of the term semantics. This is to assist the reader to understand that semantics is the connection between a representation and its meaning.

The chapter introduced the triple made of a (subject, predicate, object) as the basic unit of information in an RDF Graph. Then there was a discussion on why this thesis will restrict its focus to RDF Graphs focused on a single subject. This restricted RDF Graph termed Single Subject-focused RDF Graph (SSRG).

The chapter then discussed ontology and how the term is used in the context of the Semantic Web. The chapter then showed how ontology is connected via special triples and by predicate. Then the discussion moved on to reason that ontological information on the Semantic Web is dynamic and unreliable and this represents challenges when displaying Semantic Web data using static presentations.

The final section introduced adaptive user interfaces as interfaces that self-modify in response to user actions. An AUI observes user actions, learns from these and forms a user model. A recommender then applies the user model to data to form a display from that data. The AUI approach is applied to the challenges of displaying Semantic Web data, dynamic and unreliable ontology, in the rest of this thesis.

## CHAPTER THREE RELATED WORK

---

This chapter reviews different approaches to displaying Semantic Web data in Semantic Web browsers. Approaches to forming data displays are grouped into lexical, semantic and user preference-based approaches. Lexical approaches base display decisions upon characters strings. Semantic approaches utilise the semantic information in RDF Graphs to form displays. User Preference approaches consider knowledge of the user when producing displays of Semantic Web data.

Dadzie and Rowe (2011) survey many Semantic Web data browsers and evaluate their interfaces from the perspective of ease of use. The survey of Semantic Web browsers in this chapter focus primarily on browsers that display data about a single subject at a time, and the focus of the review is on how the browsers adapt, or not, to different data. While there are some overlaps in the browsers surveyed, the evaluation criteria are different.

A Semantic Web browser is a type of data viewing software that displays Semantic Web data and allows the user to follow links through the Semantic Web (D. A. Quan & Karger, 2004). The primary concerns for producing a suitable display from Semantic Web data are deciding which triples to display (filtering) and how to arrange the triples on screen (ordering). This section looks at current approaches to arranging and filtering Semantic Web data in current Semantic Web browsers, especially approaches that are adaptive.

The approaches to filtering and arranging for display may exploit a combination of three sources of information: lexical, semantic and user preferences. Lexical approaches operate on characters within the labels of Semantic Web subjects, objects, and predicates. Semantic approaches exploit ontological knowledge that is provided in RDF Graphs. User preference approaches are AUI approaches that learn to filter and arrange data from user interactions.

Throughout this chapter, there are screenshots of Semantic Web browsers that illustrate usage of an approach to displaying data that is being discussed. When the Semantic Web browser software is publicly available, the screenshots use selected RDF Documents. Otherwise, the screenshots come from external sources and so show the data used in that screenshot. Using the same RDF Documents in the example Semantic Web browsers gives a better basis with which to compare the way each browser displays data. The selected RDF documents are from different ontologies to highlight how some browsers render Semantic Web data from different ontologies, differently. The selected RDF documents Tim Berners-Lee's FOAF file<sup>23</sup> and either Berlin<sup>24</sup> from the GeoNames ontology or a test calendar file<sup>25</sup> from W3C.

### 3.1 LEXICAL APPROACHES

Lexical approaches operate on characters within strings of text themselves. On the Semantic Web, the text is usually found in the labels of subjects, predicates, and objects in the triples that make up an RDF Graph. The terms lexical analysis and syntactic analysis are often used interchangeably. The term "syntactic approaches" comes from the semantic-syntactic-pragmatic distinction in Charles W. Morris' triadic division of Semiotic Signs (Morris, 1946). Lexical algorithms do not take into account the semantic information contained in RDF Graphs.

Lexical algorithms are computationally efficient because they operate only on characters within strings and do not navigate complex data structures. A lexical algorithm does not require knowledge of the user. On the Semantic Web, once `rdfs:labels` are available for all entities then a lexical algorithm requires no further network transactions.

---

<sup>23</sup> <http://www.w3.org/People/Berners-Lee/card>

<sup>24</sup> <http://sws.geonames.org/2950159/about.rdf>

<sup>25</sup> <http://www.w3.org/2002/12/cal/test/bus-hrs.rdf>

Lexical approaches to filtering and arranging have no external dependencies that require expensive lookups into the corpus. Lexical approaches also need no user preference data. Where lexical similarity is a reasonable approximation for conceptual similarity, then lexical similarity can be a computationally efficient basis to form displays of Semantic Web data.

The main types of lexical approaches are Document Ordering, Alphabetical Ordering and Lexical Matching.

### 3.1.1 Document Ordering

Document ordering displays the triples in the order they appear in the source RDF Document. Brownsauce (Steer, 2003), Disco Hyperdata Browser (Bizer & Gauß, 2007), and the Quick and Dirty RDF Browser (Gutteridge, 2012) use document ordering (see Figures 3.1, 3.2, 3.3 & 3.4). The RDF standard does not expect that RDF documents will have triples in a useful order, but it could be true that a document author (or data source) could supply triples that are meaningfully ordered. It is unclear how to use RDF Document Ordering when triples are combined from multiple sources.

 Source: <a href="http://www.w3.org/People/Berners-Lee/card#i">http://www.w3.org/People/Berners-Lee/card#i</a>	
Person	Other References
<b>Male</b> <b>Timothy Berners-Lee</b>	
<b>family_name:</b> Berners-Lee <b>nick:</b> TimBL timbl <b>givenname:</b> Timothy <b>preferred:</b> <a href="http://www.w3.org/People/Berners-Lee/card#i">http://www.w3.org/People/Berners-Lee/card#i</a> <b>mbox_sha1sum:</b> 965c47c5a70db7407210cef6e4e6f5374a525c5c <b>name:</b> Timothy Berners-Lee <b>label:</b> Tim Berners-Lee <b>title:</b> Sir <b>workplaceHomepage:</b> <a href="http://www.w3.org/">http://www.w3.org/</a> <b>assistant:</b> Amy van der Hiel <b>seeAlso:</b> <a href="http://www.w3.org/2007/11/Talks/search/query?date=All+past+and+future+talks&amp;event=None&amp;activity=None&amp;name=Tim+Berners-Lee&amp;country=None&amp;language=None&amp;office=None&amp;rdfOnly=yes&amp;submit=Submit">http://www.w3.org/2007/11/Talks/search/query?date=All+past+and+future+talks&amp;event=None&amp;activity=None&amp;name=Tim+Berners-Lee&amp;country=None&amp;language=None&amp;office=None&amp;rdfOnly=yes&amp;submit=Submit</a> <a href="#">Tim Berners-Lee's editable FOAF file</a> <b>based_near:</b> <b>longitude:</b> -71.091840 <b>latitude:</b> 42.361860 <b>sameAs:</b> <a href="http://identi.ca/user/45563">http://identi.ca/user/45563</a> <a href="http://www.advogato.org/person/timbl/foaf.rdf#me">http://www.advogato.org/person/timbl/foaf.rdf#me</a> <a href="http://www4.wiwiiss.fu-berlin.de/dblp/resource/person/100007">http://www4.wiwiiss.fu-berlin.de/dblp/resource/person/100007</a> <a href="http://www4.wiwiiss.fu-berlin.de/bookmashup/persons/Tim+Berners-Lee">http://www4.wiwiiss.fu-berlin.de/bookmashup/persons/Tim+Berners-Lee</a>	<a href="http://www.w3.org/2000/10/swap/data#Cwm">http://www.w3.org/2000/10/swap/data#Cwm</a> developer <i>this</i>  <a href="http://dig.csail.mit.edu/2005/ajar/ajaw/data#Tabulator">http://dig.csail.mit.edu/2005/ajar/ajaw/data#Tabulator</a> developer <i>this</i>  <a href="#">Tim Berners-Lee's FOAF file</a> maker <i>this</i>  <a href="#">timbl's blog</a> maker <i>this</i>  <a href="http://dig.csail.mit.edu/2007/01/camp/data#course">http://dig.csail.mit.edu/2007/01/camp/data#course</a> maker <i>this</i>  <a href="#">Tim Berners-Lee's editable FOAF file</a> maker <i>this</i>  <a href="#">W3C</a> member <i>this</i>  <a href="http://dig.csail.mit.edu/data#DIG">http://dig.csail.mit.edu/data#DIG</a> member <i>this</i>  <a href="#">Tim Berners-Lee's FOAF file</a> primaryTopic <i>this</i>  <a href="#">Tim Berners-Lee's editable</a>

Figure 3.1: Tim Berners-Lee's FOAF file in Brownsauce. Triples are shown in Document Order.


 Source: <a href="http://sws.geonames.org/2950159/about.rdf">http://sws.geonames.org/2950159/about.rdf</a>	
Feature	Other References
<b><a href="http://sws.geonames.org/2950159/">http://sws.geonames.org/2950159/</a></b>	
<b>name:</b> Berlin <b>countryCode:</b> DE <b>population:</b> 3426354 <b>longitude:</b> 13.41053 <b>altitude:</b> 74 <b>alternateName:</b> Berlini Berlino Berlino BeirlĀn Berlyn BerlĀ@n Ô²Ô¸Ô?Ô-Ô«Ô¶ á??á??á? á??á??á??á?? Ð?ÐµÑ?Ð»Ñ?Ð½ Ð?ÐµÑ?Ð»Ñ?Ð½ Berlim Berlin Berlin	<a href="http://sws.geonames.org/2950159/about.rdf">http://sws.geonames.org/2950159/about.rdf</a> attributionURL <i>this</i>  <a href="http://sws.geonames.org/2950159/about.rdf">http://sws.geonames.org/2950159/about.rdf</a> primaryTopic <i>this</i>

Figure 3.2: Berlin from GeoNames in Brownsauce. Triples are shown in Document Order.

**https://www.w3.org/People/Berners-Lee/card#i [view]**

Q&D RDF Browser is powered by [Graphite](#) and [ARC2](#) and hosted by [ECS](#) at the [University of Southampton](#).

76 Triples

**Tim Berners-Lee** a Male

<https://www.w3.org/People/Berners-Lee/card#i>

- cert:key → [\\_g1-arc892eb1](#)
- rdf:type → [con:Male](#), [foaf:Person](#)
- sioc:avatar → <https://www.w3.org/People/Berners-Lee/images/timbl-image-by-Coz-cropped.jpg>
- rdfs:label → "Tim Berners-Lee"
- rdfs:seeAlso → Tim Berners-Lee's editable FOAF file, <https://www.w3.org/2007/11/Talks/search/query?date=All+past+and+future+talks&event=None&activity=None&name=Tim+Berners-Lee&country=None&language=None&office=None&rdfOnly=yes&submit=Submit>
- con:assistant → <https://www.w3.org/People/Berners-Lee/card#amy>
- con:homePage → Tim Berners-Lee's FOAF file
- con:office → [\\_g1-arc892eb2](#)
- con:preferredURI → ["https://www.w3.org/People/Berners-Lee/card#i"](https://www.w3.org/People/Berners-Lee/card#i)
- con:publicHomePage → Tim Berners-Lee's FOAF file
- owl:sameAs → <http://www.advogato.org/person/timbl/foaf.rdf#me>
- <http://www.w3.org/ns/pim/space#preferencesFile> → <https://timbl.com/timbl/Data/preferences.n3>
- <http://www.w3.org/ns/pim/space#storage> → <https://timbl.databox.me/>
- foaf:account → <http://en.wikipedia.org/wiki/User:Timbl>, [http://twitter.com/timberners\\_lee](http://twitter.com/timberners_lee), <http://www.reddit.com/user/timbl/>
- foaf:based\_near → [\\_g1-arc892eb5](#)
- foaf:family\_name → "Berners-Lee"
- foaf:givenname → "Timothy"
- foaf:homepage → <https://www.w3.org/People/Berners-Lee/>
- foaf:img → [https://www.w3.org/Press/Stock/Berners-Lee/2001\\_europaem\\_eighth.jpg](https://www.w3.org/Press/Stock/Berners-Lee/2001_europaem_eighth.jpg)



Figure 3.3: Tim Berners-Lee's FOAF file in the Quick and Dirty RDF Browse. Triples are shown in Document Order.

**http://sws.geonames.org/2950159/about.rdf [view]**

Q&D RDF Browser is powered by [Graphite](#) and [ARC2](#) and hosted by [ECS](#) at the [University of Southampton](#).

119 Triples

<http://sws.geonames.org/2950159/about.rdf> a Document

- rdf:type → [foaf:Document](#)
- foaf:primaryTopic → <http://sws.geonames.org/2950159/>
- cc:license → <http://creativecommons.org/licenses/by/3.0/>
- cc:attributionURL → <http://sws.geonames.org/2950159/>
- cc:attributionName → "GeoNames"<sup>^^xsd:string</sup>
- dcterms:created → "2006-01-15"<sup>^^xsd:date</sup>
- dcterms:modified → "2012-09-19"<sup>^^xsd:date</sup>
- ← rdfs:isDefinedBy of ← <http://sws.geonames.org/2950159/>

<http://sws.geonames.org/2950159/> a Feature

- rdf:type → [geonames:Feature](#)
- rdfs:isDefinedBy → <http://sws.geonames.org/2950159/about.rdf>
- geonames:name → "Berlin"
- geonames:alternateName → "베를린"@ko, "برلين"@arc, "ベルリン"@ja, "ബെർലിൻ"@th, ...show 87 more...
- geonames:officialName → "Berlin"@de
- geonames:featureClass → [geonames:P](#)
- geonames:featureCode → [geonames:P.PPLC](#)
- geonames:countryCode → "DE"
- geonames:population → "3426354"
- geo:lat → "52.52437"
- geo:long → "13.41053"
- geo:alt → "74"



Figure 3.4: Berlin from GeoNames shown in the Quick and Dirty RDF Browse. Triples are shown in Document Order.

The Disco Hyperdata Browser (Bizer & Gauß, 2007) can simultaneously display RDF triples from multiple HTTP sources. Disco shows the source of the data using a legend to the right of the triple (see Figures 3.5 & 3.6).

Disco - Hyperdata Browser (About)

## Tim

URI:  

Property	Value	Sources
type	<a href="http://www.w3.org/2000/10/swap/pim/contact#Male">http://www.w3.org/2000/10/swap/pim/contact#Male</a>	G2
type	<a href="#">Person</a>	G2 G4
label	Tim Berners-Lee	G2
seeAlso	<a href="http://dig.csail.mit.edu/2008/webdav/timbl/foaf.rdf">http://dig.csail.mit.edu/2008/webdav/timbl/foaf.rdf</a>	G2
seeAlso	<a href="http://www.w3.org/2007/11/Talks/search/query?date=All+past+and+future+talks&amp;event=None&amp;activity=None&amp;nar">http://www.w3.org/2007/11/Talks/search/query?date=All+past+and+future+talks&amp;event=None&amp;activity=None&amp;nar</a>	G2
seeAlso	<a href="http://www.w3.org/People/Berners-Lee/card.rdf">http://www.w3.org/People/Berners-Lee/card.rdf</a>	G4
assistant	<a href="#">Amy van der Hiel</a>	G2
homePage	<a href="http://www.w3.org/People/Berners-Lee/card">http://www.w3.org/People/Berners-Lee/card</a>	G2
work	...	G2
preferred	<a href="http://www.w3.org/People/Berners-Lee/card#i">http://www.w3.org/People/Berners-Lee/card#i</a>	G2
publicHomePage	<a href="http://www.w3.org/People/Berners-Lee/card">http://www.w3.org/People/Berners-Lee/card</a>	G2
sameAs	<a href="http://identi.ca/user/45563">http://identi.ca/user/45563</a>	G2
sameAs	<a href="http://www.advogato.org/person/timbl/foaf.rdf#me">http://www.advogato.org/person/timbl/foaf.rdf#me</a>	G2
sameAs	<a href="#">Tim Berners-Lee</a>	G2
sameAs	<a href="http://www4.wiwiiss.fu-berlin.de/dblp/resource/person/100007">http://www4.wiwiiss.fu-berlin.de/dblp/resource/person/100007</a>	G2
account	<a href="http://en.wikipedia.org/User:Timbl">http://en.wikipedia.org/User:Timbl</a>	G2
account	<a href="http://identi.ca/timbl">http://identi.ca/timbl</a>	G2
account	<a href="http://twitter.com/timberners_lee">http://twitter.com/timberners_lee</a>	G2
based near	...	G2
family_name	Berners-Lee	G2
Given name	Timothy	G2
homepage	<a href="http://www.w3.org/People/Berners-Lee/">http://www.w3.org/People/Berners-Lee/</a>	G2
image		G1 G2
knows	<a href="#">Danny Ayers</a>	G1
knows	<a href="#">Henry Story</a>	G1
knows	<a href="http://danbri.org/foaf#danbri">http://danbri.org/foaf#danbri</a>	G1
knows	<a href="#">John Rana</a>	G1

### Sources

Displayed information originates from the following RDF graphs:

G1 <http://dig.csail.mit.edu/2008/webdav/timbl/foaf.rdf>

G2 <http://www.w3.org/People/Berners-Lee/card>

G3 <http://www.w3.org/2000/10/swap/data>

G4 <http://heddley.com/edd/foaf.rdf>

### Session Cache

Display all RDF graphs that are currently in your session cache.

Figure 3.5: Tim Berners-Lee's FOAF file in Disco. Data is shown in document order. [Black zig-zags denote edits that omit spaces so that the screenshot shows relevant features]

**Resource <http://sws.geonames.org/2950159/>**

URI:

Property	Value	Sources
alternateName	Berlien	<a href="#">G1</a>
alternateName	Berliini	<a href="#">G1</a>
alternateName	Berlino	<a href="#">G1</a>
alternateName	Berlyn	<a href="#">G1</a>
alternateName	Berlynas	<a href="#">G1</a>
alternateName	Berlín	<a href="#">G1</a>
alternateName	Berlín	<a href="#">G1</a>
alternateName	Berolinum	<a href="#">G1</a>
alternateName	Birlinu	<a href="#">G1</a>
alternateName	Беро́ливо	<a href="#">G1</a>
alternateName	Берлин	<a href="#">G1</a>
alternateName	Берлін	<a href="#">G1</a>
alternateName	بیرلین	<a href="#">G1</a>
alternateName	බර්ලින්	<a href="#">G1</a>
alternateName	เบอรัลลิน	<a href="#">G1</a>
alternateName	ベリン	<a href="#">G1</a>
alternateName	베를린	<a href="#">G1</a>
ISO country code	DE	<a href="#">G1</a>
feature class	<a href="http://www.geonames.org/ontology#P">http://www.geonames.org/ontology#P</a>	<a href="#">G1</a>
feature code	<a href="http://www.geonames.org/ontology#P.PPLC">http://www.geonames.org/ontology#P.PPLC</a>	<a href="#">G1</a>
map	<a href="http://www.geonames.org/2950159/berlin.html">http://www.geonames.org/2950159/berlin.html</a>	<a href="#">G1</a>
nearby features	<a href="http://sws.geonames.org/2950159/nearby.rdf">http://sws.geonames.org/2950159/nearby.rdf</a>	<a href="#">G1</a>
official name	Berlin	<a href="#">G1</a>

Figure 3.6: Berlin from GeoNames in Disco. Triples are shown in document order. [Black zig-zag denote an edit that omits space so that the screenshot shows relevant features]

### 3.1.2 Alphabetical Ordering

Alphabetical ordering is a collation method for ordering a list of strings according to the ordering of an alphabet, specifically the Latin alphabet and its variants. The default browser for DBPedia<sup>26</sup> (see Figure 3.7) displays triples alphabetically by predicate label, whether that label is an `rdfs:label` or heuristically derived from the predicate URI.

<sup>26</sup> Live demo: <http://dbpedia.org/page/Berlin>



## About: Berlin

An Entity of Type : [administrative region](#), from Named Graph : <http://dbpedia.org>, within Data Space : [dbpedia.org](#)

Berlin (/bərˈlɪn/, [bɛʁˈliːn]) is the capital of Germany and one of the 16 states of Germany. With a population of 3.5 million people, it is the second most populous city proper and the seventh most populous urban area in the European Union. Located in northeastern Germany on the banks of Rivers Spree and Havel, it is the centre of the Berlin-Brandenburg Metropolitan Region, which has about six million residents from over 180 nations. Due to its location in the European Plain, Berlin is influenced by a temperate seasonal climate. Around one-third of the city's area is composed of forests, parks, gardens, rivers and lakes.

Property	Value
<a href="#">dbo:PopulatedPlace/areaTotal</a>	<ul style="list-style-type: none"><li>891.85</li></ul>
<a href="#">dbo:abstract</a>	<ul style="list-style-type: none"><li>Berlin (<span><span>/<span><span>b</span><span>ər</span><span>ˈ</span><span>l</span><span>ɪ</span><span>n</span></span>/</span></span>, <span><span>[</span>bɛʁˈliːn<span>]</span></span>) is the capital of Germany and one of the 16 states of Germany. With a population of 3.5 million people, it is the second most populous city proper and the seventh most populous urban area in the European Union. Located in northeastern Germany on the banks of Rivers Spree and Havel, it is the centre of the Berlin-Brandenburg Metropolitan Region, which has about six million residents from over 180 nations. Due to its location in the European Plain, Berlin is influenced by a temperate seasonal climate. Around one-third of the city's area is composed of forests, parks, gardens, rivers and lakes. First documented in the 13th century and situated at the crossing of two important historic trade routes, Berlin became the capital of the Margraviate of Brandenburg (1417–1701), the Kingdom of Prussia (1701–1918), the German Empire (1871–1918), the Weimar Republic (1919–1933) and the Third Reich (1933–1945). Berlin in the 1920s was the third largest municipality in the world. After World War II, the city was divided; East Berlin became the capital of East Germany while West Berlin became a de facto West German exclave, surrounded by the Berlin Wall (1961–1989) and East Germany territory. Following German reunification in 1990, Berlin was once again designated as the capital of united Germany. Berlin is a world city of culture, politics, media and science. Its economy is based on high-tech firms and the service sector, encompassing a diverse range of creative industries, research facilities, media corporations and convention venues. Berlin serves as a continental hub for air and rail traffic and has a highly complex public transportation network. The metropolis is a popular tourist destination. Significant industries also include IT, pharmaceuticals, biomedical engineering, clean tech, biotechnology, construction and electronics. Modern Berlin is home to world renowned universities, orchestras, museums, entertainment venues and is host to many sporting events. Its urban setting has made it a sought-after location for international film productions. The city is well known for its festivals, diverse architecture, nightlife, contemporary arts and a high quality of living. Over the last decade Berlin has seen the emergence of a cosmopolitan entrepreneurial scene. <sup>(en)</sup></li></ul>
<a href="#">dbo:areaCode</a>	<ul style="list-style-type: none"><li>030</li></ul>
<a href="#">dbo:areaTotal</a>	<ul style="list-style-type: none"><li>891850000.000000 <sup>(xsd:double)</sup></li></ul>
<a href="#">dbo:country</a>	<ul style="list-style-type: none"><li><a href="#">dbr:Germany</a></li></ul>

Figure 3.7: Berlin for DBpedia. Triples are shown in alphabetical order by predicate.

### 3.1.3 Lexical Matching

Lexical matching use patterns in the triples to make decisions about arranging triples in a display. Alshukaili, Fernandes, and Paton (2016) use lexical matching as part of a probabilistic soft logic program that combines RDF graphs.

Falcons (Cheng & Qu, 2009) is a keyword-based Semantic Web search engine (see Figure 3.8). Filtering and ranking search results is analogous to filtering and ordering triples for display. Falcons generates a virtual document for each entity in a corpus of Semantic Web data. The virtual document contains a string derived heuristically from the URI of the entity, and the string literals associated with the entity; typically the object values of `rdfs:label` and `rdfs:comment`. Keyword searches in Falcon are ranked according to how well the keywords

are similar to the virtual documents. Falcons' keyword search weights terms according to how often they appear in the corpus of virtual documents; terms that appear less often are given higher weight than terms that appear more often. Falcons is a lexical approach because the process of filtering and ordering is lexical.

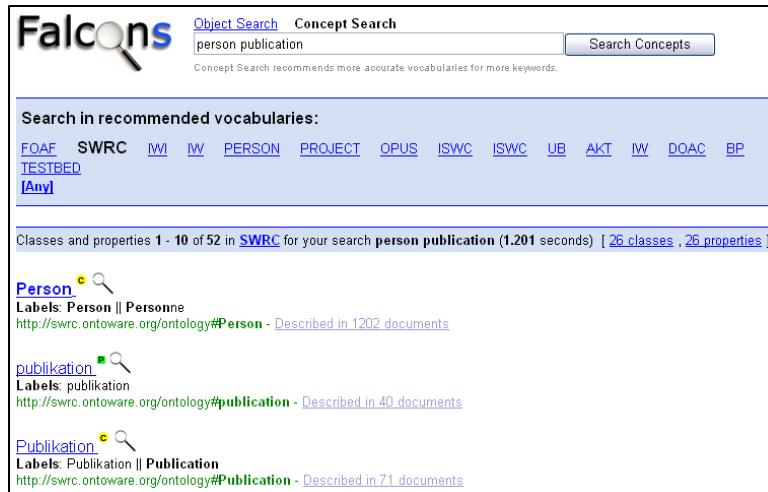


Figure 3.8: A screenshot of Falcons Concept Search (Cheng & Qu, 2009)

## 3.2 SEMANTIC APPROACHES

Semantic approaches to displaying Semantic Web data operate on the meanings attached to data, usually by reference to external sources. The Semantic Web provides facilities for exploiting semantic information because the semantics are attached to the data, and the semantics are often formalised into ontologies. Common semantic approaches include predicate matching, type matching, and ontological reasoning.

### 3.2.1 Predicate Matching

Predicate Matching forms data into displays by matching predicates in the data with predicates expected by an ontology. Predicate matching is a semantic approach because predicates connect Semantic Web data to ontology. The two approaches to predicate matching are string matching and graph matching.

### 3.2.1.1 String Matching

String matching approaches use the string contents of predicates, or the predicate labels, to decide how to arrange Semantic Web data in a display. An example of semantic predicate string matching is Seeliger and Paulheim's browser (2012) that automatically arranges related triples into groups (see Figure 3.9). The browser uses the WordNet distance of predicate labels from pairs of triples as distance metrics for a clustering algorithm. The groups are automatically labelled by using the nearest ancestor word of all predicates in a group from WordNet's tree. If the most common ancestor word is too high level, then the most frequent predicate label in the group is used instead.

Subject	Predicate	Object
<a href="http://dbpedia.org/resource/Tomorrow_in_a_Year">http://dbpedia.org/resource/Tomorrow_in_a_Year</a>	label	Tomorrow, in a Year@en
<a href="http://dbpedia.org/resource/Tomorrow_in_a_Year">http://dbpedia.org/resource/Tomorrow_in_a_Year</a>	label	<a href="http://dbpedia.org/resource/Rabid_Records">http://dbpedia.org/resource/Rabid_Records</a>
<a href="http://dbpedia.org/resource/Tomorrow_in_a_Year">http://dbpedia.org/resource/Tomorrow_in_a_Year</a>	record label	<a href="http://dbpedia.org/resource/Rabid_Records">http://dbpedia.org/resource/Rabid_Records</a>
release		
Subject	Predicate	Object
<a href="http://dbpedia.org/resource/Tomorrow_in_a_Year">http://dbpedia.org/resource/Tomorrow_in_a_Year</a>	release date	2010-01-28** <a href="http://www.w3.org/2001/XMLSchema#date">http://www.w3.org/2001/XMLSchema#date</a>
<a href="http://dbpedia.org/resource/Tomorrow_in_a_Year">http://dbpedia.org/resource/Tomorrow_in_a_Year</a>	release date	2010-03-01** <a href="http://www.w3.org/2001/XMLSchema#date">http://www.w3.org/2001/XMLSchema#date</a>
<a href="http://dbpedia.org/resource/Tomorrow_in_a_Year">http://dbpedia.org/resource/Tomorrow_in_a_Year</a>	released	2010-01-28** <a href="http://www.w3.org/2001/XMLSchema#date">http://www.w3.org/2001/XMLSchema#date</a>
<a href="http://dbpedia.org/resource/Tomorrow_in_a_Year">http://dbpedia.org/resource/Tomorrow_in_a_Year</a>	released	2010-03-01** <a href="http://www.w3.org/2001/XMLSchema#date">http://www.w3.org/2001/XMLSchema#date</a>
Length		
Subject	Predicate	Object
<a href="http://dbpedia.org/resource/Tomorrow_in_a_Year">http://dbpedia.org/resource/Tomorrow_in_a_Year</a>	Length	5563.0** <a href="http://dbpedia.org/datatype/second">http://dbpedia.org/datatype/second</a>

Figure 3.9: Screenshot of a Semantic Web browser that automatically groups triples using semantic predicate matching based on strings (Seeliger & Paulheim, 2012)

The Quick and Dirty RDF Browser (Gutteridge, 2012) and Marbles (Becker & Bizer, 2009) recognises some predicates from the Friend of a Friend (FOAF) and GeoNames (GEO) ontologies and will display triples containing those predicates differently. Specifically, for certain predicates indicating images, they display the images linked by the object member of the triple instead of printing the URI as a text string.

### 3.2.1.2 *Graph Matching*

Graph matching approaches form displays of Semantic Web data by applying transformation steps which match parts of an RDF graph to style rules. For example, using XSLT to transform an RDF document into a display matching XPath expressions (Pietriga, Bizer, Karger, & Lee, 2006) to spaces in templates. Graph Stylesheets<sup>27</sup> (GSS) is a transformation language for displaying RDF Graphs. GSS is primarily used in IsaViz (W3C, 2007) a tool for visualising RDF graphs. Another approach, Xenon (D. Quan & Karger, 2004) uses a different RDF-based stylesheet ontology that matches using SPARQL queries instead of XPath. Stegeman, Ziegler, Hussein & Gaulke (2012) uses XSLT to match the results of a SPARQL query to widgets.

A limitation for Graph Matching is that a given RDF Graph has many representations in RDF/XML. This means the XSLT rules, which match and transform RDF/XML, are sensitive to the how an RDF Graph is encoded into RDF/XML.

### 3.2.2 *Type Matching*

Type Matching approaches select between alternative displays depending on the `rdf:type` of the subject being displayed. The most common form of type matching is template based. Templates are used to arrange and filter triples in Exhibit (Huynh, Karger, & Miller, 2007), IsaViz (W3C, 2007, p. 3), Marbles (Becker & Bizer, 2009), Tabulator (Berners-Lee et al., 2006), Haystack (D. Quan, Huynh, & Karger, 2003), and Zitgist DataViewer (OpenLink Software, 2009). This section discusses the templating systems Exhibit Lens (Huynh et al., 2007), Fresnel (Pietriga et al., 2006) and Ozone (D. Quan et al., 2003).

---

<sup>27</sup> <http://www.w3.org/2001/11/IsaViz/gss/gssmanual.html>

### 3.2.2.1 Exhibit Lens

Exhibit (Huynh, Karger, & Miller, 2007) is an HTML/CSS/JSON/Javascript based template system for browsing a fixed corpus that includes a template system called “Lens.” Exhibit collections are shown using Views and navigated using Facets. The Exhibit View uses the Lenses as templates to display individual data items from a collection (see Figures 3.10 & 3.11). Exhibit works on a fixed corpus of data, and so it is assumed the author of the Exhibit-based visualisation will invest time in data comprehensiveness, having a stable ontology and matching visualisation methods to the data. Exhibit, therefore, works on more restrictive assumptions than the Semantic Web.

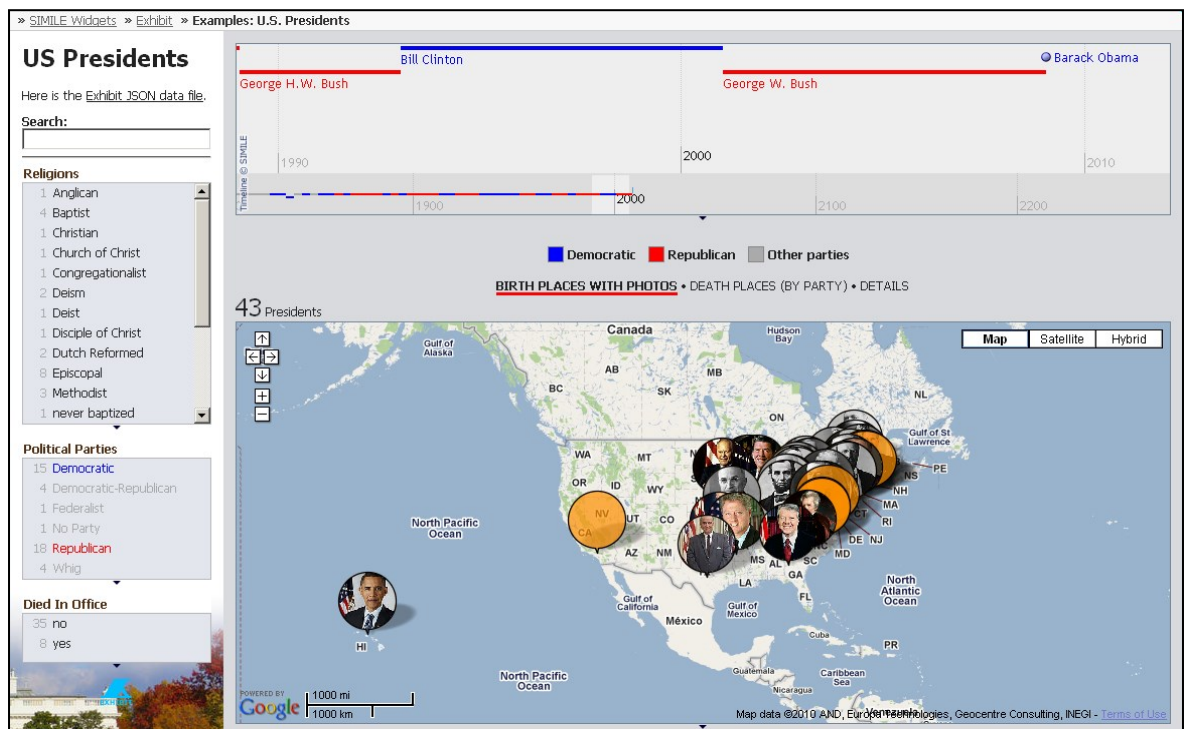


Figure 3.10: Exhibit showing keyword search, facets, timeline, and map. (Screenshot of <http://www.simile-widgets.org/exhibit/examples/presidents/presidents.html>, taken 19 Jan 2017)

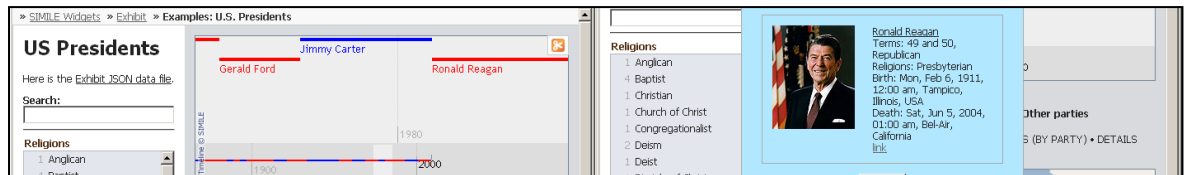


Figure 3.11: Exhibit showing a timeline overview (left) and a detail view of a person (right).

### 3.2.2.2 Fresnel

Fresnel (Pietriga et al., 2006) is a template system for Semantic Web data. Fresnel is supported by the Semantic Web browsers IsaViz (W3C, 2007), Longwell (MIT, 2005) and Marbles (Becker & Bizer, 2009). Fresnel achieves some measure of display device independence by conceptually separating into Fresnel Lenses and Fresnel Formats. Lenses specify how triples are filtered and ordered for display. Fresnel Formats specify how Lenses are visually presented. Formats have hooks that are styled using Cascading Stylesheets (CSS). Fresnel templates are RDF-based and described in the Web Ontology Language (OWL)<sup>28</sup>.

Fresnel matches triples using SPARQL expressions or the XPath-like Fresnel Selector Language (FSL). Fresnel supports grouping of Lenses and Formats. Groups allow lens or format instructions to be attached to groups as a whole. Groups also specify larger units analogous to a compound widget made of multiple components.

Marbles (Becker & Bizer, 2009) allows the user to switch between different Formats when these are available. For example, Marbles has switchable data presentations for full data, photos only or mobile versions (see Figure 3.12). The view is selected depending on the URI used to access the Marbles browser. No other user interface affordances are given to switch the view.

<sup>28</sup> <http://www.w3.org/TR/2012/REC-owl2-overview-20121211/>



Figure 3.12: Marbles' alternative data presentations (from left to right) Full view, Summary view, Photo view

### 3.2.2.3 Ozone

Ozone (D. Quan et al., 2003) is an RDF based user interface ontology for the Haystack Semantic Web browser. Ozone allows the definition of user interface elements called views. In Haystack, operations on Semantic Web data can be performed by linking control widgets to operations (called Adenine). Ozone views are attached to RDF data via the `rdfs:Class` that they support. Ozone views have attached requirements for display size so that Haystack can select the appropriate view compared to the available display space, e.g. full-screen mode or summary mode. Ozone views can also be embedded by other views to allow reuse.

### 3.2.3 Ontological Reasoning

Ontological Reasoning extends type matching, based on matching the `rdf:type` or `rdfs:Class`, to reasoning about the ontology of the RDF Graph.

Alshukaili, Fernandes and Paton (2016) have an approach that uses Programmable Soft Logics (PSL) to combine inferences from lexical, semantic and user preferences to improve keyword search of Semantic Web data. Their approach focuses on learning about the structure of the data. While the approach focuses on search results, it is relevant because the search process

filters and orders for relevance which is a similar task to filtering and ordering to build a display.

A PSL program consists of weighted first-order logic assignment rules where the weighting denotes “more or less likely to be true or false.” Situations are evaluated against the rules and the weights combined to give an overall likelihood of true/falseness. Alshukaili et al.’s approach uses a lexical similarity in many rules. Semantic information is gathered by inferences on the ontology itself, and user preferences are incorporated via user feedback from previous interactions with the system.

Alshukaili et al.’s approach is robust because the use of PSL means that partial, uncertain and inductive inferences can be gathered together to build a case for likelihood of similarity. If a certain form of information (e.g. ontological information about `rdf:type`) is missing then, the approach will still produce good results. However, like many semantic approaches on the Semantic Web, Alshukaili’s approach requires dereferencing many URIs, and that could translate into many network transactions. Moreover, the user feedback mechanism begins with no knowledge of user preferences and these will take some time to learn.

### 3.3 USER PREFERENCE APPROACHES

User preference approaches make decisions about arranging Semantic Web data by incorporating information about the user. User preferences are a special case of semantic information but are distinguished from other semantic approaches in that the source of the information is the user’s conceptual relationship with the data rather than from either the author of the data, the author of the ontology or some other third party. Marbles (Becker & Bizer, 2009) allows the user to switch between available views: this is an adaptable approach but is not adaptive because view switching occurs in response to explicit user actions.



### 3.4 SUMMARY

Lexical approaches to grouping and ordering have the advantage that they operate quickly with no external dependencies that require expensive lookups into the corpus. Lexical approaches also need no user preference data. Common lexical approaches are to leave triples in the order they appear in the RDF Document, Alphabetical ordering or String Similarity. Where lexical similarity is a reasonable approximation for conceptual similarity, then string similarity can be a computationally efficient way to group triples for display.

Semantic approaches exploit knowledge of semantics to make decisions about filtering and ordering for display. Semantic approaches include matching; which matches at the level of predicate and templates that match at the level of ontology. Semantic approaches will work efficiently if the program has high-speed access to its entire corpus, such as if the corpus is entirely contained (enclosed) on a single server. An enclosed corpus is a reasonable assumption for a search engine because a search engine can only be expected to search for items which it knows (i.e. in its corpus). However, the Semantic Web is closed in any practical sense. Enclosure might be achieved if network access to the corpus has speed increases to the level they are comparable to local corpus access. Until this happens, approaches that rely on semantic approaches may be too slow to be of practical use.

Templates are a semantic approach for displaying Semantic Web data that operates at the level of ontology. Templates are more likely to filter and arrange displays of Semantic Web data in ways that make more sense to users. However, templates are selected based on the `rdf:type` or `rdfs:class` of the subject. Semantic Web has unstable ontologies because the presence of a triple matching a particular subject and predicate cannot be relied upon. A user could write templates to suit their needs, however in practice template writing may be too time-consuming.

Under the right conditions – stable predicate patterns, known ontology and some match with user goals – templates can work well because users learn where a particular template places the data for which they are looking. Templates often display the most useful data more prominently and display conceptually related data in groups.

A common goal of arranging data into a display is to place conceptually related data in close proximity. In a template, it is the template's designer that applies their estimations of conceptual proximity when they design the template. Seeliger and Paulheim's browser (2012) groups triples using a clustering algorithm that groups by the semantic similarity of predicate labels. Alshukaili, Fernandes and Paton's approach (2016) combines lexical, semantic and ontological information using a probabilistic soft logic program to select data that is related.

However, where semantic information is available, stable and known, then it should be made use of when forming displays of Semantic Web data. As discussed earlier, User Preferences are a special case of semantic information where the user preferences are stored where the Semantic Web browser can access them quickly. A Semantic Web-browser can also cache certain useful semantic information for quick access. There is a strong case to cache `rdfs:label`. Additionally, if other ontological information was cached, then it could be used to reason for the purposes of display. The caveat is that ontologies on the Semantic Web are unstable and so ontological information might not be available or reliable. Also, ontological information does not necessarily represent the user's view on how data should be displayed.

Other semantic web browsers exist with similar features to those above. Notably, Dipper<sup>29</sup>, Sig.ma (Tummarello et al., 2010) and URI Burner<sup>30</sup> as surveyed in Dadzie and Rowe (2011). There are no current Semantic Web browsers that use an AUI and so the use of an AUI is novel. While some Semantic Web browsers do self-modify in response to the data (e.g. Marbles), to qualify as an AUI, the Semantic Web browser must self-modify in response to the user.

---

<sup>29</sup> <http://api.talis.com/stores/iand-dev1/items/dipper.html> and <http://notes.3kbo.com/talis>

<sup>30</sup> <http://linkeddata.uriburner.com>

## CHAPTER FOUR USER STUDY I: DO USERS AGREE ON THE RELATEDNESS OF TRIPLES?

---

This chapter explores the first research question by studying the extent to which human raters agree on the relatedness of triples. If users do not agree on the relatedness of triples, this will provide support for the argument that users may have different preferences for the display of triples.

**Question 1.** *Is there sufficient diversity in user preferences for displaying Semantic Web data to justify the overhead of an adaptive user interface that learns how to group and order?*

The chapter is structured as follows: The user study's introduction and hypothesis, methodology, the results of the user study, and conclusions.

The study described here was originally part of a larger study that investigated lexical algorithms for predicting the relatedness of pairs of triples. However, only the part of that study relevant to this thesis is described in this chapter.

This study measures the agreement between multiple raters regarding the relatedness of triples. Relatedness means to be associated or connected. In Single Subject RDF Graph (SSRGs), relatedness is most strongly expressed in the similarity in meanings of the predicate. For example, `firstName` and `familyName` are related because they refer to information about a user's name, whereas `firstName` and `dateOfConstruction` are unrelated because the first refers to information about a name and the second to information about time.

## 4.1 HYPOTHESES

The hypotheses are:

H0: Raters do not give the same ratings when asked to rate the relatedness of triples

H1: Raters give the same ratings when asked to rate the relatedness of triples

## 4.2 METHODOLOGY

The study compares the participant ratings for the relatedness of two triples with the same subject. There were twenty participants and each participant rated fifty pairs of triples.

Participants were recruited by personal contact. Almost all participants were acquaintances of the researcher before the study. Participants complete a demographic questionnaire which is summarised in “Participant Demographics” (see Section 4.2.3). Participants also give informed consent. A copy of the demographic questionnaire is in Appendix One.

Participant performed the rating tasks on an iPad2 running a custom HTML web app. Each participant first completed five familiarisation ratings and then rated fifty pairs of triples. Of the fifty pairs, forty asked about the relatedness of pairs of triples and ten asked about the extent to which a pair of triples contain the same. The databank of triples used in this user test is available on GitHub at <https://github.com/Stormrose/GPRank>.

Figure 4.1 shows the screen for a rating task. A guiding question is at the near the top of the screen. The pair of triples have a common subject, and so this subject is integrated as part of the question and bolded (e.g. subject The Beatles). Underneath the question is the predicate and object for a pair of triples. Each triple is shown with a colon separating the

predicate (left) from the object (right), for example, “pastMembers: Pete Best” represents a triple with the subject “The Beatles”, the predicate “pastMembers” and the object “Pete Best”. Underneath the pair of triples is the rating slider. Labels appear on the ends of the slider to guide the participant. The slider snaps to five positions and defaults to the left-most position. There are approximately 2 centimetres between choices on the slider. The five positions on the slider are encoded as the whole numbers {0, 1, 2, 3, 4}. The left of the slider corresponds to a value of 0, and the rightmost end of the slider corresponds to a rating value of 4. The numbers were not labelled on the slider. Instead, the sliders are labelled on the two ends, as explained below.

### Question

---

To what degree do these two snippets give related information about  
**The Beatles?**

activeYearsEndYear : 1970-01-01 00:00:00
pastMembers : Pete Best

Totally  
unrelated

Very strongly  
related

Continue

---

Progress:

*Figure 4.1: Screenshot for the study where the user has given a rating of 1*

Figure 4.1 shows where a participant has moved the slider one position from the left and this is encoded as a rating of 1 out of {0,1,2,3,4}.

Participants give relatedness ratings for forty pairs of triples. The question for participants is: To what degree do these two snippets give related information about SubjectX? For this

category of questions, the slider ends are labelled “Totally unrelated” and “Very strongly related”. The pairs of triples used in the relatedness questions are listed in Appendix Two and have the question numbers 10 to 49.

Additionally, participants give ratings for ten pairs of triples based on the extent to which the two triples contain the same information. The question presented to participants is: To what extent do these two snippets provide the same information about SubjectX? For this category of questions, the slider ends are labelled “Completely different” and “Exactly the same”. The pairs of triples used in these types of questions are listed in Appendix Two and have the question numbers 0 to 9.

After supplying a rating, the participant presses the continue button to move to the next rating task. The slider was reset to the left (corresponding to a value of zero out of five) for each question. At the bottom of the screen, there is a progress bar that shows how much of the study has been completed. The participant has a forced break of a few seconds every ten questions.

Participants rated the same fifty pairs of triples, but each participant encounters the pairs of triples in an individually random order. Participant responses were stored on the iPad2 using HTML local storage.

#### **4.2.1 Measuring Agreement**

This study uses inter-rater reliability to measure the reliability of the agreement between raters for the rating tasks in this study. Inter-rater reliability is a statistical measure of the reliability of the agreement amongst different raters.

The ratings given by participants are encoded as ordinal numbers, so although the ratings are made on a ranked scale, the difference between the ranks is not quantifiable. Krippendorff's Alpha Reliability Coefficient (Krippendorff, 2004), henceforth referred to as  $\alpha$ , is a measure of inter-rater reliability for multiple raters and ordinal data. Krippendorff's  $\alpha$  is a real number between 0 and 1 with zero representing perfect disagreement and 1 representing perfect agreement between the raters. This study will interpret  $\alpha$  using the scales from Landis and Koch (1977), which is given in Table 4.1.

Range	Agreement
0 to .20	Slight
.21 to .40	Fair
.41 to .60	Moderate
.61 to .80	Substantial
.81 to 1	Near Perfect

*Table 4.1: Interpreting inter-rater reliability (Landis & Koch, 1977)*

#### 4.2.2 Study Location and Time.

The study was carried out between the 3<sup>rd</sup> and 14<sup>th</sup> of June 2012 in the Waikato area. Twenty people participated in the study.

#### 4.2.3 Participant Demographics

Participants were asked to complete a demographic questionnaire to give an overall indication of demographic biases that may exist in the participant pool. The results are not analysed by demographic attribute. A copy of the questionnaire is in Appendix One. The demographic questions ask about participant gender, age, ethnicity, first languages, profession, and education level. Participants are not required to respond to any questions



and could supply multiple answers so the numbers of responses might not equal the number of participants. A summary of the participant demographic attributes follows.

There were seven female and 13 male participants. There was one participant aged 15 – 19 years, 11 aged 20 – 24, two aged 25 - 29, two aged 30 – 39 and four aged 40 – 50. Genders and age bands are taken from those used by Statistics New Zealand.

Participant ethnicity included New Zealand European (10), Asian (5), New Zealander or Kiwi (3), European (3), Maori (1) and White (1). Three participants listed dual ethnicities. Ethnicity labels are supplied by the participants.

Seventeen participants had English as a first language, and three participants did not list English as a first language. Other first languages included five Asian languages and one European language. Two participants listed more than one first language. The participants supply language names. Since this user study is primarily a language task, participants without English as a first language (3) may have found completing the user study more difficult.

Participant profession included 11 graphic designers, two educators, two from other design/media professions, one business person, an individual in the IT industry and three who did not specify a profession. Participants could specify more than one profession. Professions categories are self-nominated by the participants.

Participant education levels varied from 3 with Masters, seven with post-graduate qualifications, four at Bachelors, 5 in tertiary education at an unspecified level and one who did not specify an education level. Participants self-nominated their level of education.

### 4.3 RESULTS

Table 4.2 shows the number of ratings at a given value for each question. Each row represents one of the fifty questions. The second to sixth columns contain the number of participants giving the rating at the column heading for the question indicated in the first column.

Question#	Rating (Number of Raters)				
	0	1	2	3	4
0	18	0	1	1	0
1	14	1	3	2	0
2	8	6	4	2	0
3	2	6	4	8	0
4	4	6	6	3	1
5	4	5	3	5	3
6	1	1	2	9	7
7	0	1	3	11	5
8	0	0	1	14	5
9	0	0	6	11	3
10	10	7	3	0	0
11	11	3	2	3	1
12	15	3	2	0	0
13	11	3	4	1	1
14	0	2	2	5	11
15	4	8	2	2	4
16	0	1	2	8	9
17	0	1	1	3	15
18	0	0	0	3	17
19	0	0	1	6	13
20	6	5	7	2	0
21	5	6	3	3	3
22	13	2	2	2	1
23	4	5	4	7	0
24	14	6	0	0	0

Question#	Rating (Number of Raters)				
	0	1	2	3	4
25	4	5	4	6	1
26	3	5	4	5	3
27	5	7	2	3	3
28	6	7	3	2	2
29	1	0	5	8	6
30	3	7	5	4	1
31	9	6	4	1	0
32	6	5	6	2	1
33	12	1	0	0	7
34	7	5	3	4	1
35	7	2	8	3	0
36	0	0	1	5	14
37	3	1	0	3	13
38	2	1	3	10	4
39	4	0	0	10	6
40	16	4	0	0	0
41	16	2	2	0	0
42	15	2	1	1	1
43	11	5	2	2	0
44	4	4	3	6	3
45	6	4	7	3	0
46	1	0	3	5	11
47	3	5	4	3	5
48	3	0	2	8	7
49	3	1	1	8	7

Table 4.2: Summary of Ratings

For half of the questions (25 out of 50 questions), half of the participants (10 or more of 20 participants) gave the same rating. The greatest number of participants agreeing on a single rating for a single question is 18 participants agreeing on a rating of 0 for question number 0. More than two-thirds of the participants (14 or more out of 20 participants) gave the same rating in 11 questions of 50. Questions number 5 and 26 had the lowest agreement (measured as having 3, 4 or 5 participants supplying each of the possible ratings).

For half of the questions about related pairs (20 questions of 40), half of the participants (10 or more out of 20 participants) gave the same rating. The greatest number of participants agreeing on a single rating for a single question about related pairs is 17 participants agreeing on a rating of 0 for question number 18.

More than two-thirds of the participants (14 or more out of 20 participants) gave the same rating in 3 questions regarding related pairs out of 10 questions. Question number 5 had the lowest agreement out of all the pairs being rated for identical information.

The Krippendorff  $\alpha$  for all questions is 0.449 and this, on the scale from Landis and Koch (1977), indicates a moderate level of inter-rater reliability.

## 4.4 DISCUSSION

Half the participants were aged in their early twenties, and there were no participants older than fifty years of age. Most participants were first language speakers of English, over half are graphic designers, and almost all have specified some degree of tertiary education. This may have introduced some bias into the results.

There is a moderate level of reliability in the agreement between raters (0.449), and so the null hypothesis  $H_0$  is rejected, and  $H_1$  accepted with the caution that the reliability of agreement is moderate.

$H_1$ : Raters give the same ratings when asked to rate the relatedness of triples

The moderate level of agreement demonstrates that raters differ on how they interpret the relatedness of triples. This indicates that users may have different expectations for how triples are arranged in a display. However, this implication rests on the assumption that relatedness of triples and proximity in a display of triples are linked.

The moderate level of inter-rater agreement provides support for answering the first research question in the affirmative; that there is sufficient diversity in user preferences for displaying Semantic Web data to justify the overhead of an adaptive user interface.

## 4.5 SUMMARY

The chapter documented a user study that addressed the first research question to justify an adaptive user interface approach to displaying Semantic Web data. The user study tested how twenty participants rated the relatedness of fifty pairs of triples on an ordinal scale from 0 to 4. There is only a moderate level of inter-rater agreement between participants, and this indicates that an adaptive user interface approach that learns individual user preferences may give better individual results. Two user preference learning algorithms are proposed in Chapter Five Chapter Five User preferences for grouping and ordering and then tested in Chapter Six User Study II: Learning user preferences for grouping and ordering.

## CHAPTER FIVE USER PREFERENCES FOR GROUPING AND ORDERING

---

This chapter proposes three methods for forming grouped and ordered displays of Semantic Web data (called NonLearner, ListAlg, and GPRank); the three methods are tested in the following chapter. NonLearner does not learn user preference but is used as a baseline to compare the other two methods. ListAlg uses a top-down user model and GPRank a bottom-up user model.

The chapter is structured with a section for each of the three methods: NonLearner, ListAlg, and GPRank. The description of each the methods first gives an overview and then defines the user model, how grouping and ordering decisions are made using the user model, how the method learns new user preferences, and then a discussion on the approach. Then there is a discussion that compares ListAlg and GPRank. The chapter is then summarised to contextualise key parts of this chapter within the thesis.

### 5.1 NONLEARNER

NonLearner does not learn user preferences but instead makes grouping decisions based on lexical similarity. The input is an SSRG represented as a list of triples, and the output are the triples grouped and ordered. NonLearner provides a baseline against which to compare the two learning methods. It could be argued that random results should be the baseline, but should NonLearner perform better than randomness then the adaptive algorithms must also out perform NonLearner. A Javascript implementation of NonLearner is available on Github at <https://github.com/Stormrose/GPRank>.

NonLearner uses a similar rationale to that in Alshukaili (2016): that nothing is known about how to group items, then lexical methods provide an acceptable fallback. Falcons (Cheng & Qu,

2009) also extends lexical similarity to work with triples and uses a clustering algorithm to group similar triples together.

Since NonLearner does not learn, then it does not require a user model or a method for incorporating user preference information into a user model. Accordingly, the description of NonLearner focuses on how NonLearner groups and orders a list of triples that represent and SSRG.

### 5.1.1 Forming Displays from NonLearner

An SSRG that is represented as a list of triples is grouped and ordered using the following steps:

1. hierarchical clustering
2. collapsing into groups and orders
3. eliminating redundant triples.

Each of these steps is now described.

#### 5.1.1.1 *Hierarchical Clustering*

NonLearner groups triples from an SSRG using a hierarchical clustering algorithm. The input to this step is an SSRG represented as a list of triples. The output of hierarchical clustering is a tree with weights at each of the nodes and the triples at the leaf nodes.

Another clustering algorithm used in lexical clustering is k-means. Hierarchical clustering does not require a starting estimate of the number of groups in the data whereas k-means clustering requires an estimate. The number of clusters/groups may vary between SSRGs, so a universal estimate is of limited usefulness.

The clustering algorithm uses a pairwise distance metric to decide which triples to group together. The distance metric used in NonLearner is an extension of the Dice Coefficient (Dice,

1945; Sørensen, 1948). The Dice Coefficient is a value between 0 and 1, expressing the similarity of two lists of samples (with duplicate entries permitted). A zero value means that the lists are not similar and a one value means that the lists are the same. We assume the existence of two ordered lists,  $X$  and  $Y$ . The cardinality  $|X|$  of a list is the number of elements it contains. The intersection between two lists contains all elements that occur in both lists (allowing for repeated elements). Then the Dice Coefficient is defined in Equation 5.1:

$$\text{dice}(X, Y) = \frac{2|X \cap Y|}{|X| + |Y|}$$

*Equation 5.1: Dice Coefficient for lists*

The Dice Coefficient for string similarity is calculated by first decomposing the strings into case-sensitive lists of bigrams (strings of two characters). For example, we consider the two strings “firstName” and “familyName”, which are converted into the following two lists of bigrams:  $\{\text{"fi"}, \text{"ir"}, \text{"rs"}, \text{"st"}, \text{"tN"}, \text{"Na"}, \text{"am"}, \text{"me"}\}$  and  $\{\text{"fa"}, \text{"am"}, \text{"mi"}, \text{"il"}, \text{"ly"}, \text{"yN"}, \text{"Na"}, \text{"am"}, \text{"me"}\}$ . For this example, the Dice Coefficient is calculated as follows (see Equation 5.2):

$$\text{dice}(\text{"firstName"}, \text{"familyName"}) = \frac{2|\{\text{"Na"}, \text{"am"}, \text{"me"}\}|}{8 + 9} = \frac{6}{17} = 0.35$$

*Equation 5.2: Example of Dice Coefficient for the strings “firstName” and “familyName”*

The distance metric between two triples,  $t_1$  and  $t_2$ , used in NonLearner is one minus the average between the Dice Coefficients of the predicate and objects (see Equation 5.3).

$$\text{DistanceMetric}: \text{Triple} \times \text{Triple} \rightarrow [0, 1]$$

$$\begin{aligned}
& \text{DistanceMetric}(t_1, t_2) \\
&= 1.0 \\
&- \frac{\text{dice}(t_1.\text{predicate}, t_2.\text{predicate}) + \text{dice}(t_1.\text{object}, t_2.\text{object})}{2}
\end{aligned}$$

Equation 5.3: NonLearner's hierarchical cluster distance metric extends Dice Coefficient

The implementation of NonLearner uses the `hcluster()` method from the javascript library `clusterfck[sic]`<sup>31</sup> to perform the hierarchical clustering. NonLearner combines items when they are 90% similar.

#### 5.1.1.2 Collapsing into Groups and Orders

The tree structure given by the hierarchical clustering algorithm is converted into a balanced tree of height 2. The tree nodes at level one become groups, and the leaves become triples within that group. All other depth information is discarded. NonLearner cannot order groups or their member triples because hierarchical clustering has no concern for the order of the triples. At this point, NonLearner has constructed a set of groups containing triples, where the triples within each group are lexically similar.

---

<sup>31</sup> <https://github.com/harthur/clusterfck>

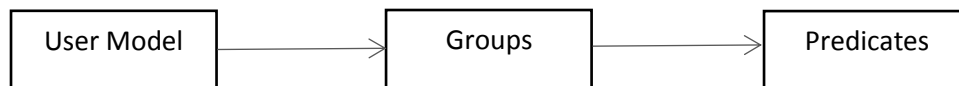


## 5.2 LISTALG

ListAlg attempts to learn user preferences for grouping and ordering triples by recording how users group and order predicates. Since ListAlg is a learning algorithm for an adaptive user interface it has a user model, a procedure for grouping and ordering a list of triples, a procedure for bringing new user preference information into a user model and these are discussed below. ListAlg is based on taking a data model that supports a grouped and ordered view and extending this data model with a set of heuristic rules to store user preferences. A Javascript implementation of ListAlg is available on GitHub at <https://github.com/Stormrose/GPRank>.

### 5.2.1 User Model

ListAlg's user model is an ordered list of groups in which each group is an ordered list of predicates (see Figure 5.1). Each predicate is unique within the user model.



*Figure 5.1: User Model for ListAlg*

The structure of the ListAlg user model has a close relationship to how triples would be grouped and ordered for display; that is, the ListAlg user model is itself stored as ordered groups containing ordered predicates.

### 5.2.2 Forming Displays from ListAlg

ListAlg creates a display from an RDF document by placing triples into display groups that correspond to groups in the user model. The order within groups also follows order within groups in the user model.

If ListAlg encounters predicates with triples for which it has no user preference information, then the UI should take this into account and encourage the user to place those triples according to their preference. An example implementation would be to place triples with no user preference information into a single group and then to display this group differently to draw user attention to these triples.

### 5.2.3 Incorporating New User Preferences

ListAlg attempts to merge information about new preferences from a candidate grouping and ordering into the user model using a series of heuristic rules. The input to this process is a current user model and new preference data, and this produces a new user model. The new preference data is represented as triples that are grouped and ordered, for example, taken from a grouped and ordered display that the user rearranged via drag and drop. The process for incorporating the user preferences is described below. A worked example follows to demonstrate the process and aid understanding.

ListAlg extracts the predicates from the new preference data and places them into an empty ListAlg user model (candidate user model) such that the grouping and ordering of the predicates correspond to the grouping and ordering of the triples that contain those predicates from the new preference data. The remainder of the process merges two ListAlg user models to produce a new user model that would replace the current user model.

ListAlg iterates over the predicates in both the current and candidate user models together in depth first order from the first predicate in the first group. If the predicates from the candidate user model and the current user model are the same, then ListAlg appends the predicate to the corresponding group in the new user model.

If predicates differ then the predicates from the candidate user model are appended into the corresponding group in the new user model until either a predicate from the candidate user model matches the predicate from the current user model, or there are no further predicates in that group within the candidate user model.

Another way to explain this is that transferring predicates from the current user model into the new pauses when there is a mismatch with the candidate user model. If the iteration of a candidate user model group has ended then predicates from the current user model are appended.

Predicates from the current user model are only appended into the corresponding group in the new user model if the predicate is not already present and does not exist elsewhere in the new or candidate user models. This restriction preserves the requirement that predicates only appear once in the new user model.

The following step-by-step example demonstrates ListAlg's procedure for incorporating new preference data. The examples use a JSON-like notation where square brackets represent ordered lists. The outer square brackets enclose a ListAlg user model. The inner square brackets contain groups of predicates. Strings within the groups represent predicates. Strings are unquoted for clarity. At the end of the examples, there is a condensed summary of the example with each step represented on a single line.

The process removes predicates from the old and candidate user models as the algorithm adds predicates to the new user model. The new user model begins empty (see Table 5.1).

Current User Model	Candidate User Model	New User Model
<pre>[   [     predicateA,     predicateB,     predicateC,     predicateD   ], [     predicateE,     predicateF   ] ]</pre>	<pre>[   [     predicateA,     predicateE,     predicateB,     predicateG   ], [     predicateD   ], [     predicateH   ] ]</pre>	<pre>[ ]</pre>

Table 5.1: ListAlg Example - Beginning

ListAlg transfers the first predicate directly into the new user model because the first predicate in the current user model and the candidate user model match (see Table 5.2).

Current User Model	Candidate User Model	New User Model
<pre>[   [     predicateA,     predicateB,     predicateC,     predicateD   ], [     predicateE,     predicateF   ] ]</pre>	<pre>[   [     predicateA,     predicateE,     predicateB,     predicateG   ], [     predicateD   ], [     predicateH   ] ]</pre>	<pre>[   [     predicateA   ] ]</pre>

Table 5.2: ListAlg Example - Predicate matches in old and candidate user models

The predicate from the candidate user model is added to the new user model because there is a mismatch between the current predicate in the current user model and the candidate user model (see Table 5.3).

Current User Model	Candidate User Model	New User Model
[ [ predicateB, predicateC, predicateD ], [ predicateE, predicateF ] ]	[ [ predicateE, predicateB, predicateG ], [ predicateD ], [ predicateH ] ]	[ [ predicateA, predicateE ] ]

Table 5.3: ListAlg Example - Mismatch between predicate in old and candidate user models

The predicate from the current user model and the candidate user model match again, so they are both appended to the corresponding group in the new user model which, in this case, is predicateB (see Table 5.4).

Current User Model	Candidate User Model	New User Model
[ [ predicateB, predicateC, predicateD ], [ predicateE, predicateF ] ]	[ [ predicateB, predicateG ], [ predicateD ], [ predicateH ] ]	[ [ predicateA, predicateE, predicateB ] ]

Table 5.4: ListAlg Example - Another predicate match

The predicates from the current user model and the candidate user model do not match, so ListAlg adds the predicate from the candidate user model to the new user model (see Table 5.5).

Current User Model	Candidate User Model	New User Model
[ [ predicateC, predicateD ], [ predicateE, predicateF ] ]	[ [ predicateG ], [ predicateD ], [ predicateH ] ]	[ [ predicateA, predicateE, predicateB, predicateG ] ]

Table 5.5: ListAlg Example - Another predicate mismatch

The first group in the candidate user model is exhausted of all predicates, so the remaining predicates from the first group of the current user model are appended to the first group in the new user model in order. However, predicates from the current user model that remain in either the new or candidate user models are not appended to the new user model. In this example, ListAlg appends predicateC to the first group of the new user model, but ListAlg ignores predicateD (for now) because predicateD appears in the candidate user model (see Table 5.6).

Current User Model	Candidate User Model	New User Model
[ [ predicateC, predicateD ], [ predicateE, predicateF ] ]	[ [ predicateD ], [ predicateH ] ]	[ [ predicateA, predicateE, predicateB, predicateG, predicateC ] ]

Table 5.6: ListAlg Example - Candidate user model group exhausted

ListAlg skips predicateE in the current user model because predicateE already exists in the new user model (in the first group). PredicateF and predicateD do not match, so ListAlg adds predicateD from the candidate user model to the corresponding (second) group in the new user model (see Table 5.7).

Current User Model	Candidate User Model	New User Model
[ [  ], [ predicateE, predicateF ] ]	[ [ ], [ predicateD ], [ predicateH ] ]	[ [ predicateA, predicateE, predicateB, predicateG, predicateC ], [ predicateD ] ]

Table 5.7: ListAlg Example - Pausing for current user model items already in the new user model

The second group in the candidate user model is now exhausted of predicates. ListAlg appends the remaining predicates from the second group of the current user model to the second group of the new user model in order. In this case, ListAlg appends predicateF to the second group in the new user model. If a group in the current user model was exhausted of predicates and the corresponding group in the candidate user model had remaining predicates, then ListAlg would append the remaining predicates from the candidate user model to the corresponding group in the new user model (see Table 5.8).

Current User Model	Candidate User Model	New User Model
<pre>[   [     ], [       predicateF     ]   ]</pre>	<pre>[   [     ], [       ], [         predicateH       ]     ]</pre>	<pre>[   [     predicateA,     predicateE,     predicateB,     predicateG,     predicateC   ], [     predicateD,     predicateF   ] ]</pre>

Table 5.8: ListAlg Example - Another candidate user model group exhausted

When there is no corresponding group between the current user model or the candidate user model, then that is treated as if there was an empty group there instead. In the following case, there is no third group in the current user model, so ListAlg treats the situation as if there was an empty group in the current user model. ListAlg inserts predicateH to a third group in the new user model. The merge is complete and the new user model is stable because ListAlg has processed all predicates in both the current and candidate user models (see Table 5.9).

Current User Model	Candidate User Model	New User Model
<pre>[   [     ], [       ]     ]</pre>	<pre>[   [     ], [       ], [         predicateH       ]     ]</pre>	<pre>[   [     predicateA,     predicateE,     predicateB,     predicateG,     predicateC   ], [     predicateD,     predicateF   ], [     predicateH   ] ]</pre>

Table 5.9: ListAlg Example - Implying an empty group in the current user model



Table 5.10 shows the current user model and candidate user models in their original states and the new user model when ListAlg has merged the old and candidate user models. The table shows an overview of the whole process. Extra line breaks aid following the process step-by-step from top to bottom.

Current User Model	Candidate User Model	New User Model
[ [ predicateA,  predicateB,  predicateC, predicateD ], [ predicateE,  predicateF ] ]	[ [ predicateA, predicateE, predicateB, predicateG  ], [  predicateD  ], [ predicateH ] ]	[ [ predicateA, predicateE, predicateB, predicateG, predicateC  ], [  predicateD, predicateF ], [ predicateH ] ]

Table 5.10: ListAlg Example - Line by line

#### 5.2.4 Discussion

ListAlg's user model is a compact format and runs in  $O(n)$  (linear) time because the integration process iterates over current user model and candidate user models in tandem. That is, the number of operations required to produce a new user model will never be greater than the count of predicates in the old and candidate models added together (see Equation 5.4).

$$\begin{aligned}
 & \text{ListAlg runs in } O(n) \text{ time, where } n \\
 & < |\text{predicates in old user model}| \\
 & + |\text{predicates in candidate user model}|
 \end{aligned}$$

Equation 5.4: Upper bound for iterations in ListAlg

ListAlg can use predicates that are URIs, so ListAlg does not incur network delays dereferencing labels or risk the inaccuracies introduced by using a heuristic derivation of predicate labels. This gives ListAlg an advantage over methods that use label data, such as NonLearner.

ListAlg can match user preferences where the diversity of predicates seen across different RDF documents is low. The calculation for Predicate Diversity is one minus the mean number of unique predicates per document encountered divided by the total number of unique predicates encountered in all documents (see Equation 5.5).

$$PredicateDiversity = 1.0 - \frac{Mean\ Unique\ Predicates\ in\ the\ RDF\ Document\ collection}{Total\ Unique\ Predicates\ in\ the\ RDF\ Document\ collection}$$

*Equation 5.5: Predicate Diversity for the diversity of predicates in a collection of RDF documents*

When Predicate Diversity is low, then the patterns of groups will not change much. ListAlg may not reflect user preferences when there are groups missing in the candidate user model that are present in the current user model. Missing groups are more likely to occur when the Predicate diversity is high.

ListAlg also assumes that users will always want predicates in the same order. While this might be the case, it is possible that a user may prefer the same predicate placed differently in different situations. For example, a user prefers the `dc:comment` predicate displayed in one place for movies and in another place for historic events. ListAlg cannot cater for this level of sophistication.

During preliminary self-testing by the researcher, predicates in groups lower in the order tended to frequently move around groups when predicate diversity was high due to ListAlg not dealing with missing groups in candidate user models.

Whether ListAlg is sufficient for real-world SSRGs is tested in the next chapter. If ListAlg performs well according to user expectations, then ListAlg's memory compactness and  $O(n)$  computation may suit constrained computation environments.

### 5.3 GROUPED PAIRWISE RANKING (GPRANK)

Group Pairwise Ranking (GPRank) is a supervised learning method learning user preferences for grouping and ordering triples. GPRank is supervised learning because a supervisor (the user) guides the learning by providing their preferred grouping and ordering. The difference between ListAlg and GPRank is that ListAlg's balanced tree user model is a top-down method and GPRank's pairwise user model is a bottom-up method.

GPRank is pairwise: all decisions are made based upon pairs of predicates. GPRank decides group membership using weighted voting between pairs of triples. GPRank orders group members using pairwise comparisons with accommodation for partial orders. GPRank must support partial orders because user preference information is incomplete. GPRank orders groups by comparing the sorting order of each predicate in one group versus every other predicate in the other group. GPRank was influenced by other sorting algorithms that are pairwise and support partial orders and extends these to support grouping Semantic Web data. Additional influences are Bayes (Bayes & Price, 1763) and simulated annealing (Kirkpatrick, 1984). The differences to Bayes are discussed in Section 5.3.4.

GPRank incorporates new preference data by adjusting the weights between pairs.

Since GPRank is a learning algorithm for an adaptive user interface it has a user model, a procedure for grouping and ordering a list of triples, a procedure for bringing new user preference information into a user model and these are discussed below. A Javascript implementation of GPRank is available on GitHub at <https://github.com/Stormrose/GPRank>.

### 5.3.1 User Model

The user model for GPRank (see Equation 5.6) stores two predicates ( $a$  and  $b$ ) and the values which associate between them: `order`, `group` and `confirmations`. The GPRank user model for a single user is of type *usergroupranks*, that is, a set of GPRankEntries.

$$GPRankEntry := (a, b, order, group, confirmations)$$

$$a \in RDFPredicates$$

$$b \in RDFPredicates$$

$$order \in [0,1] \in \mathbb{R}, \text{ the confidence that } a \text{ is ordered higher than } b$$

$$group \in [0,1] \in \mathbb{R}, \text{ the confidence that } a \text{ and } b \text{ are members of the same group.}$$

$$confirmations = \mathbb{N}_0, \text{ the count of confirmations for this tuple.}$$

$$usergroupranks := \{ x \mid x \in GPRankEntry \}$$

Equation 5.6: GPRankEntry tuple as used in a GPRank based user model

Each relation ( $a, b$ ) stores an ordering weight (`order`) in the range [0-1], a group membership affinity in the range [0-1] and a count of the number of user `confirmations` this relation has. Logically a *usergroupranks* forms a sparse table with all predicates present on both axes. Order values less than 0.5 represent the probability that the first

predicate in the relation is ordered before the second predicate. Order values greater than 0.5 represent the probability that the second predicate is ordered before the first predicate. The divergence of the order value from 0.5 indicates the strength – further from 0.5 being a stronger indication. The group affinity value works similarly with lower group affinity values representing the likelihood that the two predicates in the relation are not in the same group and higher values representing that the two predicates are in the same group.

In the following example, shown in Equation 5.7, for the predicates `firstName` and `familyName`, `gprank` is low (0.15) which indicates that triples containing the predicate `familyName` should be ordered after triples containing the predicate `firstName`, the group affinity is high (0.95) so triples containing these two predicates should be in the same group and there are four `confirmations` of this information.

$$GPRankEntry("familyName", "firstName", 0.15, 0.95, 4)$$

*Equation 5.7: Example GPRankEntry*

In order to support an open-world assumption *a*, *b*, *order* and *group* attributes can also have a value that is *undefined* and the *confirmations* attribute can also be 0 or infinite. The above specifications (see Equation 5.6) represent the usual case for brevity. The open-world assumption is so that GPRank's user model properly reports when information is incomplete and does not assume default values that might mislead a display algorithm.

To be thorough in the conceptual definition, technically the composite candidate key for user model entries is (*a*, *b* and *confirmations*) together. However, the *order* and *group* affinity for the most recent confirmation value is the most useful, so GPRank discards data from earlier confirmations. Thus, (*a*, *b*) is treated as the composite candidate key.

### 5.3.1.1 A Lookup Function

It is useful to define a function for referring to a `GPRankEntry` tuple from within a `usergrouppranks` (`ugr`) by its compound primary key (`a`, `b`). This function is used in explanations later in the chapter and for understanding how GPRank's user model operates. By definition, tuple attributes are functionally dependent upon the primary key. The lookup function will always return a single tuple of type `GPRankEntry`. The open world cases, where tuples are returned for combinations of `a` and `b` that are not explicitly stored by the implementation, are discussed later.

Defining the lookup function, `gpr`, (Equation 5.10) has several steps. Firstly, the function `gprb` (Equation 5.8) returns a single member `usergrouppranks` set. The function `gpri` (Equation 5.9) extracts the single set member returned by `gprb` to return a `GPRankEntry` tuple. For readability, `gpri` is partially composed with the current user's user model (type `usergrouppranks`, a set of `GPRankEntries`) to define the `gpr` lookup function. Finally, dot notation is used to access the value of attributes in the `GPRankEntries` (Equation 5.11).

The `gprb` function will always return a set with exactly one member. The chapter discusses the open world cases where parameters are unknown later.

$$gprb: \text{usergrouppranks} \times \text{predicate} \times \text{predicate} \rightarrow \text{usergrouppranks}$$

$$gprb(ugr, a, b) := \{ x \mid x \in (GPRankEntry \cap ugr) \wedge x.a = a \wedge x.b = b \}$$

$$\forall a \forall b: |gprb(ugr, a, b)| = 1$$

Equation 5.8: First step defining a lookup function for a particular `GPRankEntry`

The lookup function `gpr` returns a single `GPRankEntry` tuple, but `gprb` always returns a single member set. So, the definition for the intermediate function `gpri` borrows square

bracket notation with an ordinal parameter from C-Style computer programming languages. In this case, the ordinal parameter 1 means the first, and in this case only, member.

$$gpra: \text{usergroupranks}, \text{predicate}, \text{predicate} \rightarrow \text{GPRankEntry}$$

$$gpra(ugr, a, b) := gprb(ugr, a, b)[1]$$

*Equation 5.9: Second step defining a lookup function for a particular GPRankEntry*

For the sake of brevity, the parameter `ugr` will usually be omitted when it refers to the user model for the current user. This partial function is called `gpr` and will normally be written without the partial subscript. To use terminology from functional programming, this step binds the user model parameter to a function and returns a new function. `gpr` always returns a single `GPRankEntry` – it inherits this property from `gpra`.

$$gpr := \text{partial}(gpra, ugr): \text{predicate} \times \text{predicate} \rightarrow \text{GPRankEntry}$$

$$gpr_{\text{partial}}(a, b) := gpra(ugr, a, b) \text{ } ugr \text{ is bound as the first parameter.}$$

*Equation 5.10: Second step defining a lookup function for a particular GPRankEntry*

For readability, this research writes the partial function without the partial subscript

e.g. `gpr(a, b)` without the partial function subscript.

Dot notation accesses the value of a named attribute in the `GPRankEntry` tuples, such as those returned by  $gpr(a, b)$ .

$gpr(a, b).attribute$

where `attribute` is one of: `a`, `b`, `order`, `group`, `confirmations`

e.g.  $gpr(a, b).order$

$GPRankEntry(Cheese, Chalk, 1.0, 1.0, 5).a = Cheese$

*Equation 5.11: Dot notation for named members in a GPRankEntry tuple*

#### 5.3.1.2 Constraints and Assertions

There are constraints and assertions that stem from the design of GPRank's user model and its open world assumption. The undefined tuple and when confirmations are zero support the open world assumption. The predicate reversal constraint makes the user model indifferent to the order in which predicates pairs are assigned to `a` and `b` for lookup. The constraint that `a` and `b` cannot be the same is because GPRank cannot order pairs of triples that have the same predicate.

##### 5.3.1.2.1 Constraint: $a$ and $b$ cannot be the same

When  $a = b$  then `order` is undefined, `group` is the same (1.0) and `confirmations` are infinite (see Equation 5.12). This means that there is no ordering information but the two predicates are always members of the same group. The implementation used in this research avoids this situation so the definition below is included for conceptual completeness.

$$\forall x (x \in GPRankEntry \wedge x.a = x.b \\ \rightarrow GPRankEntry(x.a, x.a, undefined, 1.0, \infty))$$

*Equation 5.12:  $a$  and  $b$  cannot be the same in a GPRankEntry tuple*



#### 5.3.1.2.2 Constraint: Predicate Reversal

Given any *GPRankEntry* within a particular *usergrouppranks* (user model) another *GPRankEntry* can be calculated which is its reverse (Equation 5.13). In this calculation, the *a* and *b* predicates are swapped, the *order* is subtracted from one and the *group* and *confirmations* are transferred as they are. When the predicates are swapped then the *gprank* (ordering) is inverted but the group affinity and confirmations remain the same. The term *u* is given here to emphasise that the inversion is relevant within only a single user model.

$$\begin{aligned} \forall u \forall x: (u \in \text{usergrouppranks} \wedge x \in (\text{GPRankEntry} \cap u)) \\ \rightarrow \text{GPRankEntry}(x.b, x.a, (1.0 - x.order), x.group, x.confirmations)) \end{aligned}$$

Equation 5.13: The result of reversing *a* and *b* in a *GPRankEntry* tuple is calculatable

Which encapsulates the following shown in Equation 5.14:

$$\begin{aligned} \text{gpr}(a, b).order &= 1.0 - \text{gpr}(b, a).order \\ \text{gpr}(a, b).group &= \text{gpr}(b, a).group \\ \text{gpr}(a, b).confirmations &= \text{gpr}(b, a).confirmations \end{aligned}$$

Equation 5.14: Deriving a new *GPRankEntry* tuple from the reversal of *a* and *b*

#### 5.3.1.2.3 Statement: The undefined tuple

When a *GPRankEntry* tuple referred to by the predicates *a, b* does not exist in the implementation of a particular *usergrouppranks* then an undefined tuple is returned to satisfy the constraint that *gprb(a, b)* must always return a single *GPRankEntry* (Figure 5.15). This supports the open world assumption.

$$undefinedgpranktuple := GPRankEntry(a, b, undefined, undefined, 0)$$

$$\forall a \forall b: (|gpr\langle a, b \rangle| = 0 \rightarrow undefinedgpranktuple)$$

Equation 5.15: The undefined GPRankEntry tuple

In situations where  $a = b$  then the “ $a$  cannot be the same as  $b$ ” rule takes precedence because the same predicates are in the same group and this is unaffected by the number of confirmations. When the predicates are not the same and are also not stored in the user model, then nothing can be inferred about the GPRankEntry except that confirmations are zero.

#### 5.3.1.2.4 Statement: Confirmations as zero

To complete the open world assumption, a further rule is needed to define what happens when *confirmations* are zero. This is called the undefined tuple. The meaning is that when confirmations are zero, then `order` and `group` affinity are always undefined (Equation 5.16).

$$\forall x: (x \in GPRankEntry \wedge x.confirmations = 0 \rightarrow undefinedgpranktuple)$$

Equation 5.16: A GPRankEntry tuple with 0 confirmations is undefined

### 5.3.2 Forming Displays with GPRank

GPRank forms displays by using pairwise comparisons of predicates to organise triples into ordered groups and then order within those groups. An overview of the process follows:

1. Group: Assign triples to groups
2. Order Groups: sort the groups into preferred order
3. Order Triples: sort the triples within groups into preferred order

GPRank must perform the first step before the others, but steps 2 & 3 can be performed in either order – or even in parallel.

#### *5.3.2.1 Step One - Grouping*

The grouping stage uses a weighted peer election to decide which predicates are grouped together. The group weighting is the weighted average of the `group` affinity values for the candidate predicate and the current members of the group. However, weights are also adjusted by confirmations to favour the newest user preference information. The implementation starts each predicate in a group by itself and then uses simulated annealing (Kirkpatrick, 1984) to move predicates into more suitable groups.

The basic process is to iterate over all predicates looking for the change that would give the strongest result in a weighted peer election. In the first iteration, since all groups contain a single predicate, this will be equivalent to moving the predicates with the highest `group` affinity value into the same group. The value of the peer-election value is then assigned to the simulated annealing target. This has the effect of reducing the annealing target value over time. Further iterations merge groups when the peer election strength is stronger than the annealing target. The grouping process ends when the annealing target reduces to 0.5.

#### *5.3.2.2 Step Two – Ordering Groups*

GPRank orders groups by a weighted comparison of the order probability between all predicates in one group against all predicates in the other group. This results in a value that can be used in a sorting algorithm. It is possible that there are no relations in the user model that contain predicates from both groups; in this case, there is no group ordering information so the algorithm that does the group sorting must work with partial orders. The implementation used by this research is an exhaustive matrix sort, but any pairwise sort that supports partial orders will work.

#### 5.3.2.3 *Step Three – Ordering Predicates Within Groups*

To ordering triples within groups, GPRank uses comparisons of the pairwise ordering weights (`order`) from the user model. This step is the most sensitive to partial orders because information about the relations between might be incomplete and only the predicates from a single group can contribute ordering information. The implementation used in this research is an exhaustive matrix sort, but any pairwise sort that supports partial orders will work.

#### 5.3.2.4 *Resolving Conflicts Between `.gprank` and `grouprank()`*

A situation may arise when using GPRank where a higher ordered triple ends up in a lower ordered group. This rule resolves the inconsistency.

The order of the group takes precedence over the order of triples. If two triples are in the same display-group then they are ordered using the value of the `order` attribute from the *GPRankEntry* found in the user model, otherwise, they are ordered according to the order of the groups to which they belong.

Implementations of GPRank can avoid this situation by first ordering the groups of triples and then ordering triples within groups.

### 5.3.3 *Incorporating New User Preferences into the GPRank User Model*

Incorporating new user preference information into a GPRank user model from grouped and ordered triples begins by tagging predicates with their group number into an ordered list. The gathering process can extract this information from a grouped and ordered set of triples, such as that provided by the user from a drag and drop interface.

GPRank compares each predicate in the new preference data to every other predicate in the new preference data. If the predicates are the same, then the loop continues – there is no need to compare a predicate with itself. If the predicates are not in alphabetical order, then

the loop continues because the information from this predicate pair will be recorded when the predicates are in alphabetical order. Otherwise, if the predicates are different and in alphabetical order, then GPRank retrieves the relation, with the two predicates as primary-key, from the current user model.

If the implementation does not an existing record for `gpr(a,b)`, then a new record is created with the starting values for `order` and `group affinity` set at either 0.3333 or 0.6667 and `confirmations` is set to one. Higher ordering values indicate a greater likelihood that the predicate in the first index is ordered before the predicate in the second index. Higher group affinity means that the two predicates are likely to be in the same display group. These two numbers have a range `[0.0 - 1.0]` with 0.5 being the midpoint denoting no information or no particular preference.

If this is an existing relation, then the `confirmations` are incremented by one. If the ordering in the incoming user preference information places the first predicate earlier than the second, then the `order` value is moved halfway between its current value and 1.0. If the ordering is instead the other way, then the `order` value is moved halfway between its current value and 0.0. If the predicates are members of the same group, then the `group affinity` value is moved halfway between its current value and 1.0 otherwise if the predicate in the new preference information is not in the same group then the `group affinity` value is moved halfway between its current value and 0.0. The number of `confirmations` is increased by one, and the relation is saved back into the user model. The process repeats until all predicates in the new preference information have been processed against each other and each iteration updates the user model as above.

#### 5.3.4 Discussion

This section discusses some of the considerations that underlie GPRank. The first topic examines why GPRank is based upon predicates patterns and not ontological information from

RDF. The next section examines the GPRank weight adjustment formula and why it differs from Bayes (Bayes & Price, 1763).

GPRank does not use ontological reasoning because detecting when users might want a predicate displayed in a different place also has some challenges as described in here. At first glance, `rdf:type` seems to be a good basis for ontological reasoning – an algorithm could associate different display preferences with different values of `rdf:type`. However, this would rely on `rdf:type` being present in each SSRG and introduces challenges when multiple `rdf:type` triples are present in an SSRG. This might be made more deterministic if `rdf:type` entries were resolved to find their `rdfs:class` and `rdfs:subClassOf` values but that would come at the cost of numerous network operations and rely on the dereferenced RDF data being both available and of useful quality. These requirements are presumptive and so a solution should not rely upon them.

GPRank adjusts half-way between the current `order` or `group` affinity values and 0.0 or 1.0, and this has advantages over Bayes (Bayes & Price, 1763) that are discussed below.

The effect of halfway weighing is that new preference information has an immediate effect. For example, if the current value of `group` affinity for two predicates is 0.94 and the incoming preference data does not have the two predicates in the same group, then the new `group` affinity value will become 0.47. The effect is a signal to GPRank's display process that these two predicates would prefer to be in different groups, but this preference is weak because the weighting is close to 0.5. When the number of samples is high, Bayes requires more contrary samples to change to reflect a change in user preferences.

GPRank's move-half-way weight adjustment enables predicates that are displayed in different groups depending on which other predicates are around. The grouping process uses simulated

annealing to create groups where the members maximise their pairwise group affinity. With each iteration, the predicate to group affinities are recalculated and the highest chosen again. A group's affinity to a predicate is the average of the group member predicate's affinities with that predicate. The effect of this is that predicates that are sometimes in the same group as another predicate and sometimes not when both are present will have group-affinity values close to 0.5. Such predicate pairs will have less effect on grouping decisions because the weighting value has less effect on the group-average that forms a predicate-to-group affinity. Predicates that are consistently placed in the same group, or consistently placed in different groups, will develop values that tend towards 0.0 or 1.0. This has more effect upon the weighted average for predicate-to-group affinity.

In addition, the averages for predicate-group affinity is weighted by the number of confirmations. The strength of the weighting becomes stronger as more confirmations are received. The strength is currently capped at six confirmations based on a preliminary investigation by the researcher. The confirmation-based weighting could be the subject of more robust further research.

The previous paragraphs discuss the effect of the move-half-way weight adjustment on group affinity. The weight adjustment also affects the ordering of groups because the pairwise value used to decide which group is order before the other is also weighted based on comparing the pairwise ordering values of all members in both groups. Predicate pairs that are ordered consistently will develop strong `order` values (closer to 0.0 or 1.0), and predicate pairs that are inconsistently ordered will have `order` ordering values that converge on 0.5.

In summary, GPRank's weight adjustment encapsulates both a pairwise preference and a weighting for how much effect the grouping and ordering decisions should place upon that preference. This enables displays where predicates are displayed in different groups in different circumstances.

## 5.4 COMPARISON BETWEEN LISTALG AND GPRANK

This section discusses when ListAlg or GPRank might be a more suitable choice for a particular context. The discussion compares the projected processing and storage overheads between ListAlg and then proposes a future investigation into a hybrid user model that combines the advantages of ListAlg and GPRank. This discussion is to round out the understanding of ListAlg and GPRank.

In GPRank, the number of loop iterations will be the square of the number of predicates in the incoming user preference data. However, the number of operations to the user model will be the number of predicates in the incoming preference data squared minus itself, or  $n * (n - 1)$ . This is because predicates that are equal are skipped and the two predicates in the loop only affect the user model when the first is alphabetically earlier than the second. The number of updates to the user model is not affected by the current size of the user model. In ListAlg, the number of updates to the user model is always less than the number of predicates in the current user model plus the number of predicates in the incoming preference data. More precisely, the number of user model updates in ListAlg will be the number of unique predicates in the union of the current user model and the incoming preference data.

While it may appear that ListAlg's linear scaling model will have fewer user model updates, thus being more efficient, in real-world usage this will only remain true up to a point. The number of predicates in a document is expected to remain relatively stable while the number of predicates contained in the user model is expected to grow as the user encounters more diverse content.

There eventually becomes a point where GPRank is more efficient (fewer user model updates) for incorporating new user preference information when user models become sufficiently



large. This cross-over point is when ListAlgUMUpdateCount (Equation 5.17) is equal to GPRankUMUpdateCount (Equation 5.18).

*ListAlgUMUpdateCount*

$$:= | \text{PredicatesInIncomingUserPreferenceData} \cup \text{PredicatesInCurrentListAlgUserModel} |$$

*Equation 5.17: The number of user model update operations for ListAlg*

$$\text{GPRankUMUpdateCount} := |\text{PredicatesInIncomingUserPreferenceData}| * ( |\text{PredicatesInIncomingUserPreferenceData}| - 1 )$$

*Equation 5.18: The number of user model update operations for GPRank*

The uniqueness constraint in the union operation for ListAlgUMUpdateCount means that the actual crossover point where GPRank's update process has fewer user model update operations than ListAlg is guaranteed once the number of predicates in the user model is at least the number of predicates in the incoming user preference data squared and then plus twice itself.

#### 5.4.1 Future research: a hybrid user model

Although GPRank has a higher computational cost and its user model takes more storage than ListAlg, it is possible to improve GPRank's efficiency by looking for fragments for which ListAlg's assumptions hold (same grouping and order all the time) and then treating those fragments as if they were predicates within GPRank. This is an area for further research that may improve the efficiency of GPRank implementations.

## 5.5 SUMMARY

This chapter presented three algorithms for forming displays of Semantic Web data. NonLearner uses hierarchical clustering with a lexical distance metric. ListAlg uses heuristics to learn user preferences in a way that is space and computationally efficient. GPRank uses pairwise measures of group affinity and ordering between predicates.

The extent to which NonLearner reflects user preference is the extent to which users prefer lexically similar predicates to be grouped together. ListAlg assumes that grouping can be indexed from top to bottom and that users always want predicates in the same order under all circumstances. GPRank assumes that users will behave consistently enough that group membership can be determined by peer election.

The next chapter tests the performance of the three proposed algorithms - NonLearner, ListAlg, and GPRank - against user preferences.

## CHAPTER SIX USER STUDY II: LEARNING USER PREFERENCES FOR

### GROUPING AND ORDERING

---

This chapter tests the grouping and ordering methods proposed in the last chapter against user expectations for grouping and ordering triplets in displays of Semantic Web data. This chapter addresses the second research question by measuring the ability of the two proposed adaptive user interface methods, ListAlg and GPRank, to learn user preferences for grouping and ordering. The chapter addresses the first research question by gathering and comparing user preferences for grouping and ordering Semantic Web data.

The user study documented in this chapter uses a drag-and-drop user interface that allows users to express their grouping and ordering preferences.

#### 6.1 HYPOTHESES

Testing the algorithms involves a few questions. Firstly, do the three algorithms have better than random performance against user expectations? Secondly, do the two algorithms that learn user preferences (ListAlg and GPRank) learn? Thirdly, do the ListAlg and GPRank match user preferences more than the NonLearner? Finally, does one learning algorithm learn user preferences better than the other? The following hypotheses explore the above questions:

The experiment silently tests NonLearner performance against random ordering.

H0: NonLearner is no better than random grouping and ordering when compared to users' preferred grouping and ordering of Semantic Web data.

H1: NonLearner is worse than random grouping and ordering when compared to users' preferred grouping and ordering of Semantic Web data.

H2: NonLearner is better than random grouping and ordering when compared to users' preferred grouping and ordering of Semantic Web data.

This chapter tests the claim from the previous chapter (above) that ListAlg and GPRank can learn user preferences.

GL0: ListAlg does not learn user preferences for grouping/ordering Semantic Web documents.

GL1: ListAlg does learn user preferences grouping/ordering Semantic Web documents.

GG0: GPRank does not learn user preferences for grouping/ordering Semantic Web documents.

GG1: GPRank does learn user preferences grouping/ordering Semantic Web documents.

There is a computational cost to using the learning algorithms (ListAlg and GPRank) over NonLearner, so it is useful to know if the learning algorithms outperform the non-learning algorithm.

JL0: ListAlg is as accurate as NonLearner for predicting user grouping/ordering preferences.

JL1: ListAlg is more accurate than NonLearner for predicting user grouping/ordering preferences.

JG0: GPRank is as accurate as NonLearner when predicting user grouping/ordering preferences.

JG1: GPRank is more accurate than NonLearner when predicting user grouping/ordering preferences.

The experiment tests user preferences for grouping and ordering compared with the predictions made by ListAlg and GPRank. This gives three possible hypotheses:

K0: There is no discernible difference between ListAlg and GPRank when learning user preferences for grouping and ordering Semantic Web data.

K1: ListAlg outperforms GPRank when learning user preferences for grouping and ordering Semantic Web data.

K2: GPRank outperforms ListAlg when learning user preferences for grouping and ordering Semantic Web data.

## 6.2 METHODOLOGY

This section outlines how the hypotheses were investigated. Broadly, the investigation is in the form of a user test using a laptop with a mouse. Participants choose which of two example group/orderings are closest to their preference. Participants then provide their ideal grouping/ordering via a drag and drop interface. The test then applies user preferences to another Semantic Web document within a similar topic domain (dataset). The participant chooses which prediction is the best, and the difference between each algorithm's grouping/ordering and the user's grouping/ordering are recorded.

The experiment design requires a set of example data and a method to measure the differences between the user's preferred and algorithm calculated grouping and ordering. This section discusses the dataset and its design considerations, then the difference measurement and then there is more information about the design of the experiment itself.

### 6.2.1 The three methods

NonLearner (see 5.1 above) groups pairs of triples based on a lexical distance metric. NonLearner does not learn user preferences but may have some use when user preferences are unknown.

ListAlg (see 0 above) is a heuristic approach to combining user preferences for grouping and ordering. ListAlg mirrors how a user has grouped and ordered previously encountered Semantic Web data. ListAlg then attempts to merge in any new group/order preference information.

GPRank (see 5.3 above) uses pairwise partial orders with two weighted properties: group affinity and the top-to-bottom ordering of the pair. Groups anneal by weighted peer election. The weightings for group affinity and ordering are adjusted as more user preference data becomes available.

The ability of NonLearner to group and order triples according to user preferences will be a baseline measure to compare the performance of ListAlg and GPRank: both learning algorithms are expected to perform better than NonLearner.

### 6.2.2 Dataset design

The experiment's dataset should emulate Semantic Web conditions, and so the dataset is taken from DBPedia, which derives its data from Wikipedia infoboxes. Semantic Web data is not perfect; there are spelling errors, missing information and redundant information. An imagined scenario is a set of subjects returned in response to a query. These subjects are related, usually by a real-world type even if the `rdf:type` is not the same. The SSRGs in a set have some commonality in predicates used to express the data, but there are also be some differences in predicate patterns between SSRG. The dataset used in this user study is available on Github at <https://github.com/Stormrose/GPRank>.

Dataset Diversity is a useful way to measure the relative stability of predicate patterns between data sets. Different dataset diversities allow comparison between the performance of the GPRank and ListAlg under situations with stable predicate patterns and less stable predicate patterns.

The calculation for dataset diversity is one minus the mean number of unique predicates per document in a dataset divided by the total number of unique predicates in the dataset (Equation 6.1). The range is [0.0 – 1.0) with lower values representing a more homogeneous data set and higher values representing datasets with a greater diversity of predicates per document in the dataset.

$$\text{Dataset Diversity} = 1.0 - \frac{\text{Mean unique predicates per document in dataset}}{\text{Total unique prediates in dataset}}$$

Equation 6.1: Dataset Diversity

This study has five sets of data and one training set. The training set contains three SSRGs while the other sets contain 10 SSRGs. The SSRGs within a set share a common theme but have variation in the predicate patterns used to express the data. The training set is biological

data on plants. The datasets are Movies, Historical Events, Political Leaders, Tourist Destinations, and Books.

The data was then manually edited to ensure that each SSRG had between seven and twenty triplets so that the triples from a single SSRG could fit onto a laptop screen without scrolling.

The edits also ensured a spread of dataset diversities as shown in Table 6.1.

Dataset	Diversity
Plants (training)	0.10
Movies	0.22
Historical Events	0.62
Political Leaders	0.40
Tourist Destinations	0.75
Books	0.40

*Table 6.1: Dataset diversity in the second user study*

### 6.2.3 Measuring differences between alternative grouping and ordering

This study requires measure for the distance of two different grouping/orderings of the same set of data. In this case, the distance between the grouping/ordering expressed by an algorithm and the user-supplied grouping/ordering. The measure chosen is a modified Kendall Tau Distance (Kendall, 1938). The Kendall Tau Distance is also known as the Bubble Sort Distance because it represents the number of swap operations a bubble sort would perform to change one list into another. The modifications to Kendal Tau Distance and their rationale are described in the following paragraphs.

Kendall Tau Distance measures the distance between two lists but NonLearner, ListAlg and GPRank output their data in a grouped and ordered form. So, we flatten the groups into a list



such that the top-to-bottom order is preserved. Unique group end markers are added to the list.

Predicates that are not in both lists are removed because this represents a situation where that item does not appear in the user model, and so the method cannot group/order that item: This occurs when an algorithm group/orders a document for which it does not have group/order information for all predicates in its user model.

The Kendall Tau Distance is the minimum number of swap operations required to transform one list into the other. The Kendall Tau Distance can be normalised (Equation 6.2) so that comparisons can be made between lists of different lengths – in the case of this experiment grouping of and ordering of SSRGs that have different numbers of triples. The denominator for normalising the Kendall Tau Distance is the maximum number of bubble sort swaps possible on a list of that length, which is  $(n * (n - 1)) / 2$ . The result of the normalisation calculation is a number in the range [0,1] where 0.0 represents that the two lists are in the same order and 1.0 represents that the lists are in perfect reverse order from each other. Over many repeated measures, a randomly ordered list will average a score of 0.5 in comparison to another list of the same members.

$$\text{Normalised Kendall Tau Distance} = \frac{\text{bubble sort swap operations}}{\left( \frac{\text{listlength} * (\text{listlength} - 1)}{2} \right)}$$

*Equation 6.2: Normalised Kendall-Tau Distance*

This results in this study are expressed as Normalised Kendall Tau Closeness (Equation 6.3), which is calculated by subtracting the Normalised Kendall Tau Distance from 1.0. Higher numbers denote two lists are more similar. It is calculated by subtracting the normalised Kendall Tau Distance from 1.0.

$$\text{Normalised Kendall Tau Closeness} = 1.0 - \text{Normalised Kendal Tau Distance}$$

Figure 6.3: Normalised Kendall-Tau Closeness (NKTC) - calculation

The Normalised Kendall Tau Closeness NKTC is used by user study documented in this chapter to compare the performance of predictions for grouping and ordering made by NonLearner, ListAlg and GPRank against participant supplied grouping/orderings.

#### 6.2.4 Test Sequence

The sequence of testing for a single participant is as follows.

The researcher demonstrates the task to the participant. First research demonstrates the drag and drop interface for specifying a user's preferred grouping and ordering. Then the researcher shows the screen for selecting between the grouping and ordering created by ListAlg and GPRank.

The participant familiarises themselves with the interface. The researcher needs to be satisfied the participant can create groups, remove groups, reorder triplets within a group, move triplets into another group and reorder groups. The training round is identical to the research rounds except that nothing is recorded, and the round is shorter. The screens for the researcher demonstration and participant familiarisation are the same with different titles, so only a single set is shown here.

The drag and drop interface, shown in Figure 6.2, affords the participant a means to express their grouping and ordering preferences. Clicking on a triplet, dragging it to its new position and releasing the mouse button moves the triplet to the new location. A space, marked with a red outline, shows the participant valid drop locations. Dropping a triplet within a group

moves the triplet into that group (if not already there) and orders it accordingly. Dropping a triplet between, before or after groups creates a new group containing only that triplet. Moving all triplets out of a group, leaving the group empty, deletes the group. Finally, the participant may change the order of groups by dragging and dropping using the medium grey rectangle on the right side of each group.

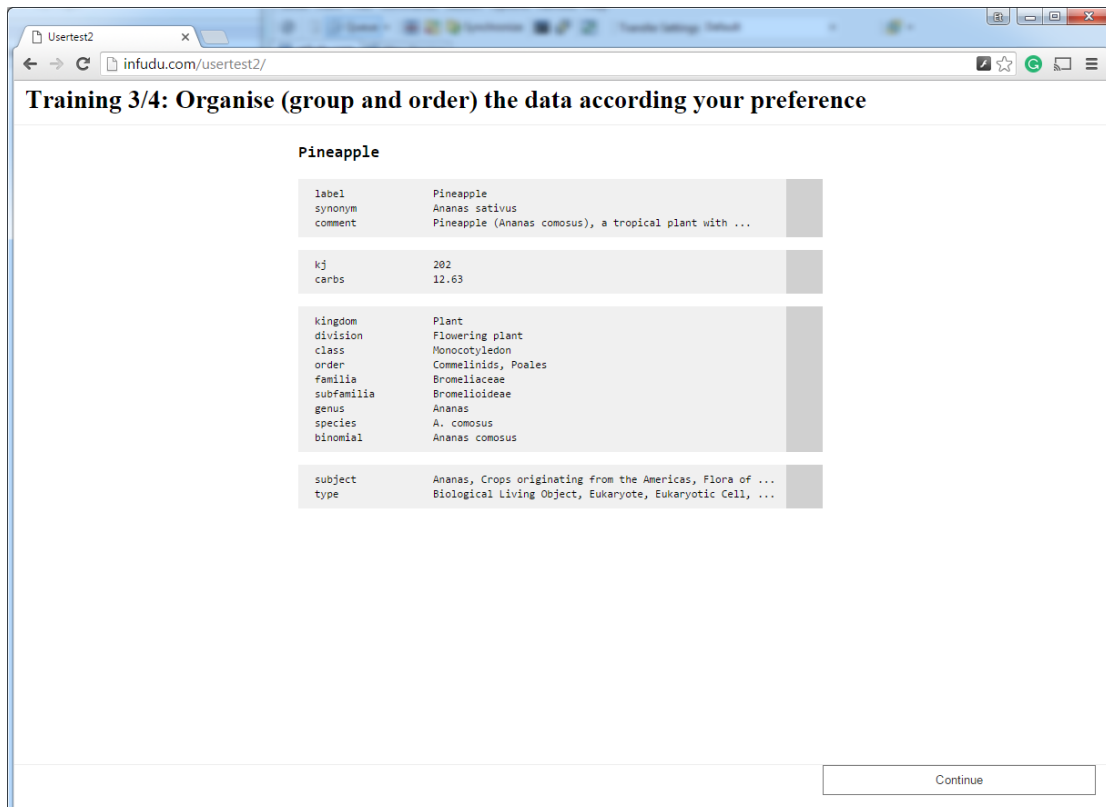


Figure 6.1: Drag and drop interface for a user to express the grouping and ordering preferences

Once the participant has expressed their preferred grouping and ordering and then pressed continue, both ListAlg and GPRank record the preferences into their user models. A new SSRG is selected and the ordered by both ListAlg and GPRank. A display is then formed using GPRank and ListAlg and the participant selects which grouping/ordering they think is best (Figure 6.3). The algorithms are not labelled, and the position of each algorithm is randomised every time this screen is displayed so that the participant does not develop a habit of always clicking one

side of the screen. In cases where the GPRank and ListAlg output are the same output then the selection of a single grouping/ordering is disabled, and the user must click the middle button labelled “Both sides appear the same.”

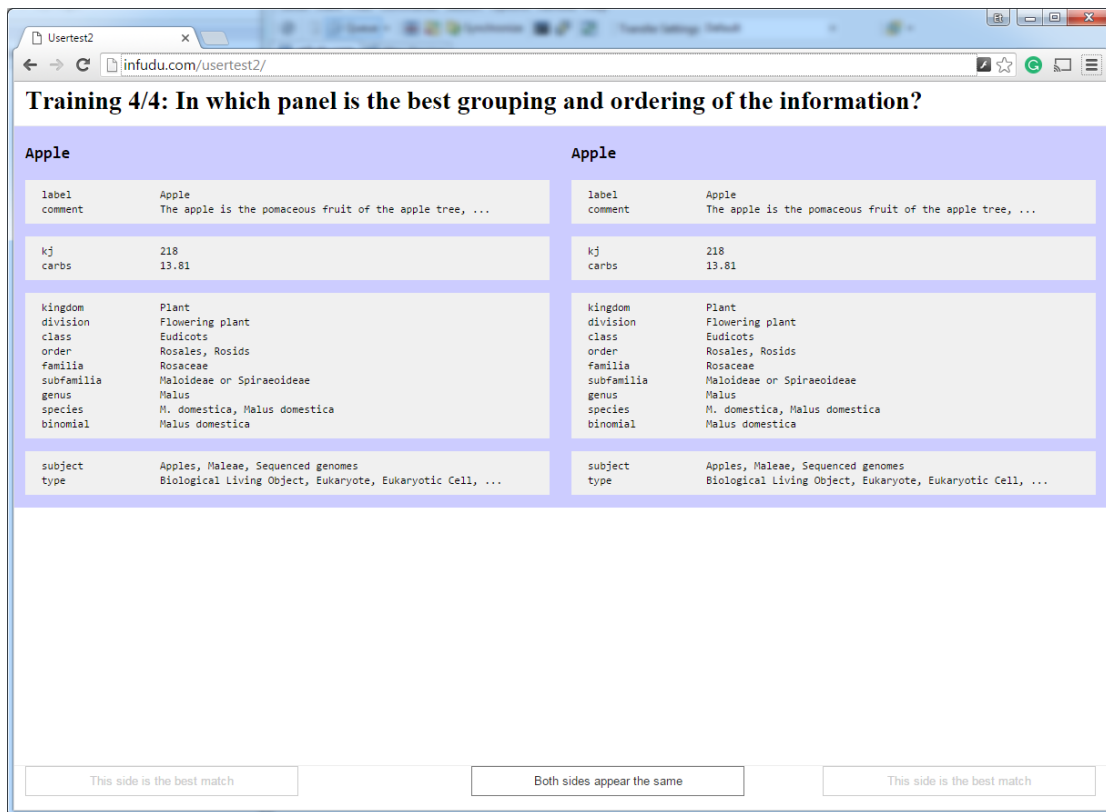
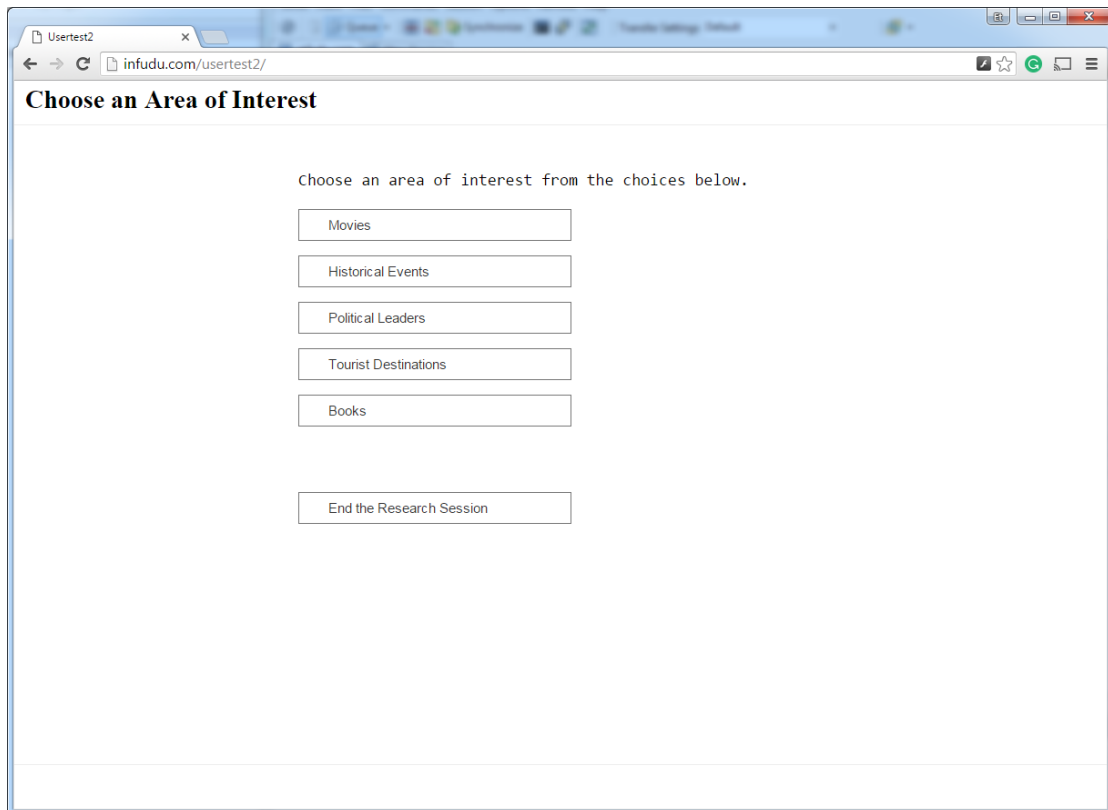


Figure 6.2: Screen for user to select one of two outputs

Following training, the user selects from one of the datasets (Figure 6.4). The participant may complete as many or as few datasets as they choose but are only allowed to attempt each dataset once. Furthermore, there is a restriction of twenty participants per dataset enforced by the programme. This screen will not display buttons for datasets that already have twenty participants.



*Figure 6.3: Screen for Selecting Dataset*

If the participant clicks the “End the Research Session” button, then they are shown a thank you screen. The participant may choose the order of datasets. The order of documents within each set is random for each participant. The first screen a user sees after clicking on an area of interest is grouped and ordered using NonLearner (Figure 6.5). The user is invited to drag and drop the data on this screen to suit their preferences. The progress bar in the top right corner relates to progress through the dataset (Figure 6.6).

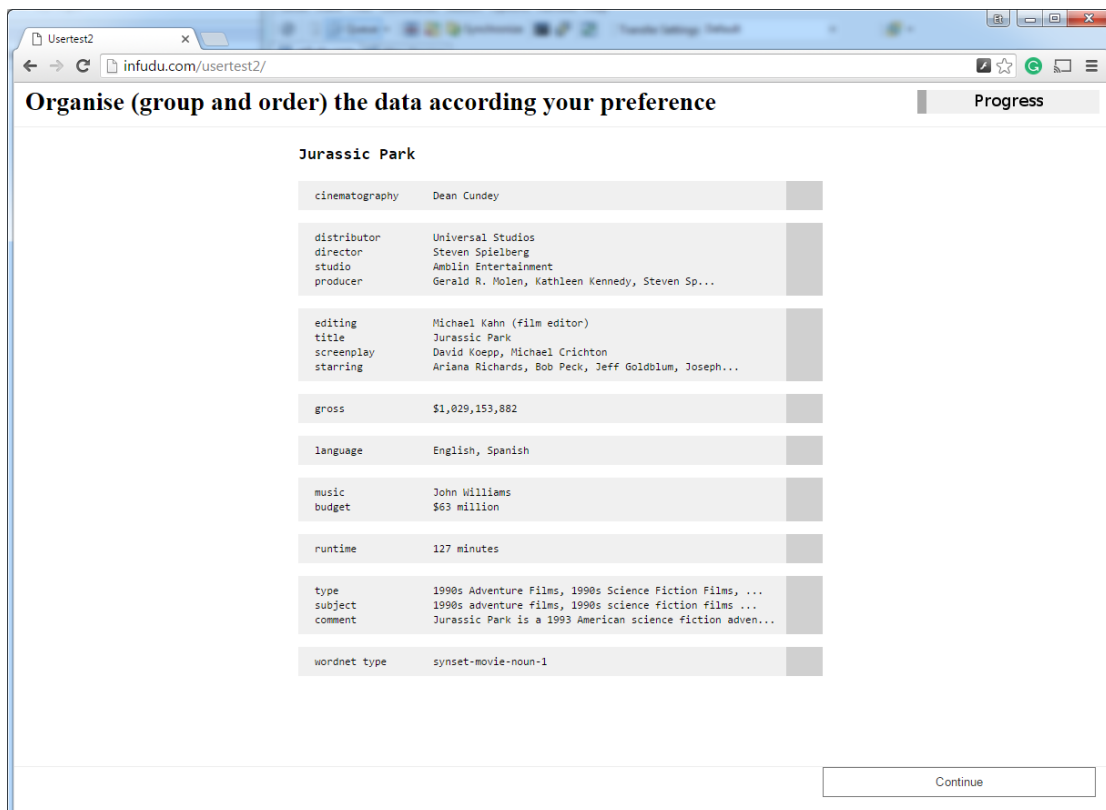


Figure 6.4: Grouping and Ordering Screen showing grouping and ordering by NonLearner

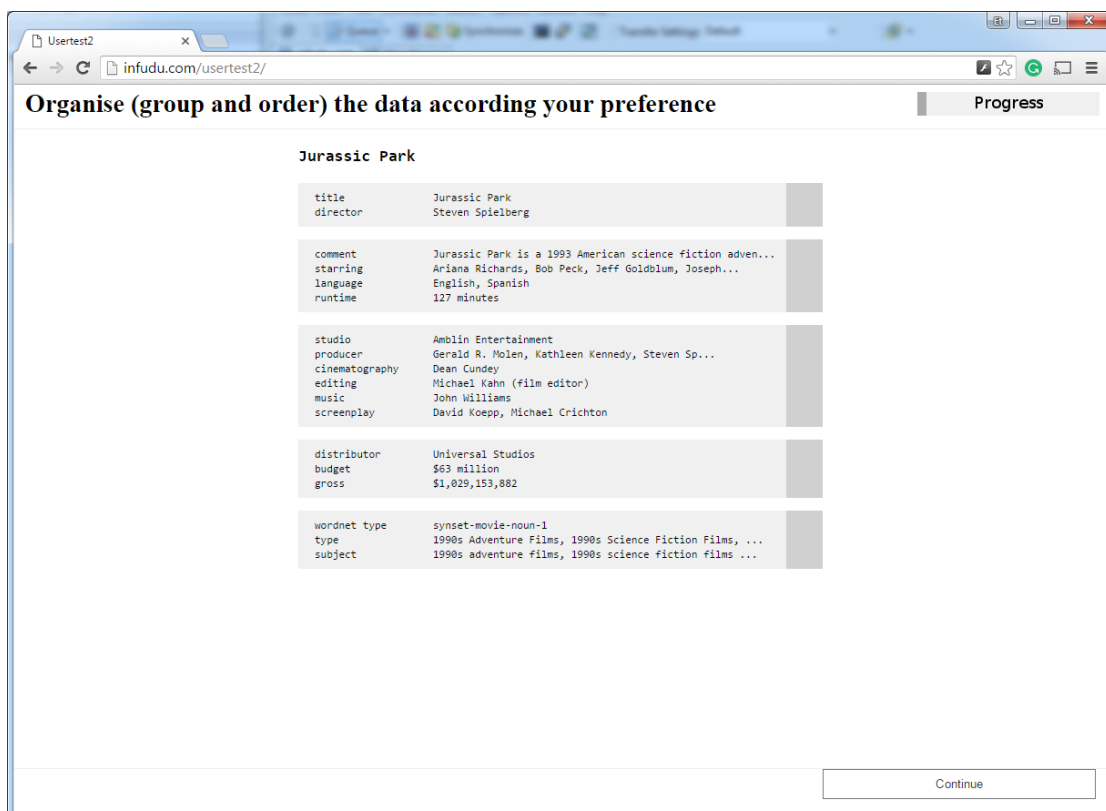


Figure 6.5: Grouping and Ordering Screen showing data as grouped and ordered by a participant

Once the participant is satisfied with the grouping and ordering, they then click the “Continue” button. In the background, ListAlg, GPRank add the user’s preferences to their user models, and then a new RDF document is randomly selected from the dataset. Each RDF document is only encountered once per participant. The RDF data is then grouped and ordered using ListAlg and GPRank and then displayed side by side in random order, as in the training (Figure 6.7).

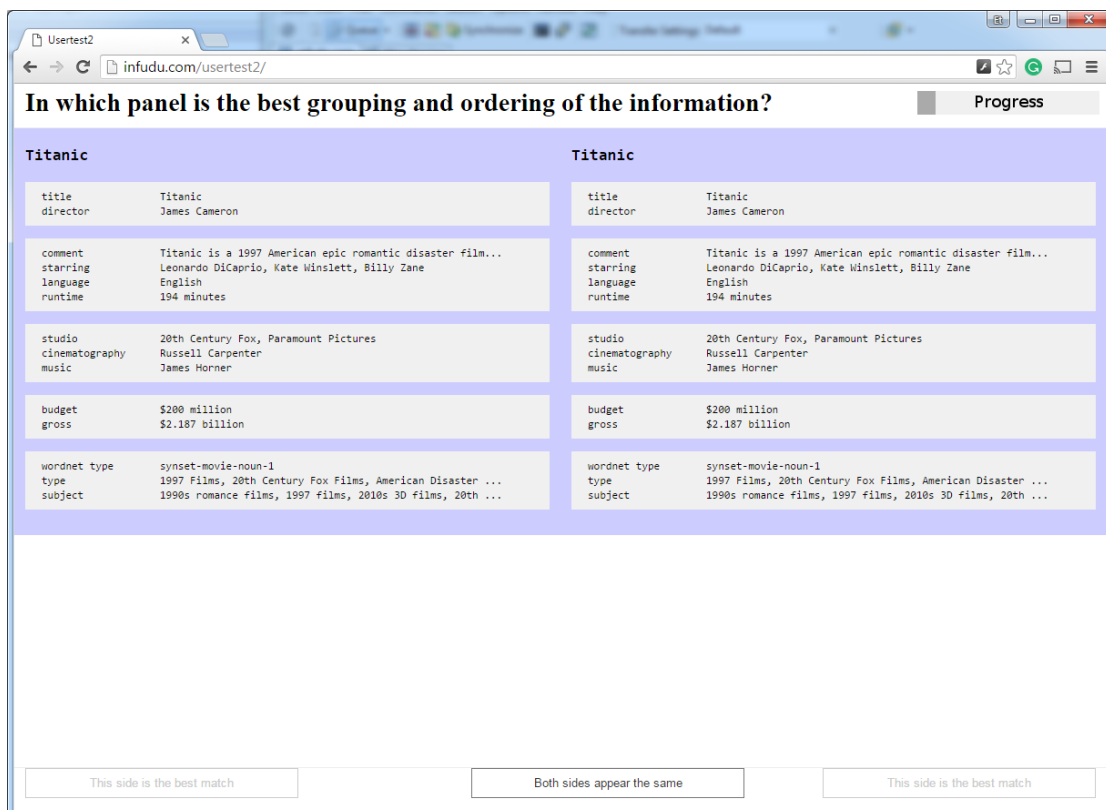


Figure 6.6: Participant selects the best Grouping/Ordering

The choice test application records the participant’s choice. The participant is then shown the drag and drop interface to fine tune the grouping and ordering (Figure 6.8). This screen has the grouping and ordering from the side the participant selected.

Predicates that have not been seen before are placed in a group at the bottom of the document. That group has a yellow background to alert the user that these predicates are new.

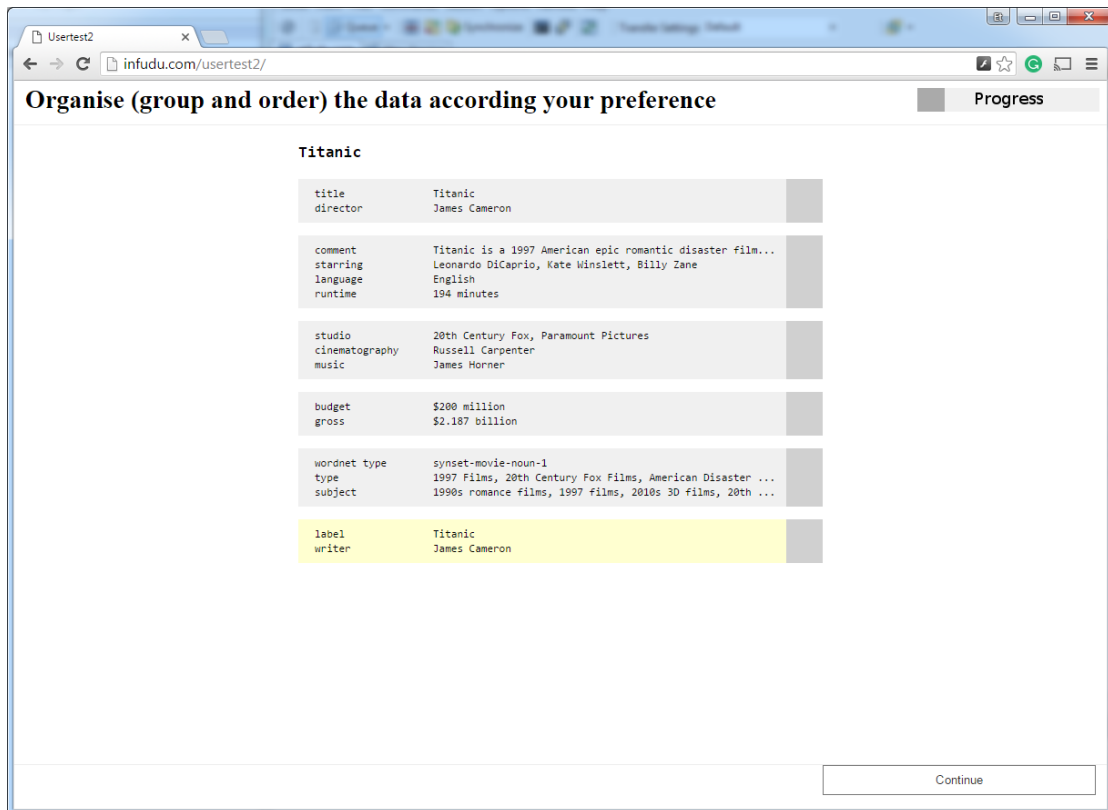


Figure 6.7: Participant can fine tune the grouping and ordering

Once the participant has finished with the drag and drag interface, they click the “Continue” button. The user’s ordering is compared using the NKTC with the outputs of ListAlg, GPRank, and NonLearner. These NKTCs for each algorithm to the participant supplied data are recorded, and the user models for ListAlg and GPRank updated with the new user preference data. Then, a new random document is selected, and the test continues repeating from the select which side screen until the participant volunteers to end or all documents in the dataset are viewed.

When a dataset is complete, the user is returned to the dataset selection screen (Figure 6.4) to choose another dataset or end the research session. Datasets that the participant has previously completed are greyed out and disabled so that no participant may repeat a completed dataset.



### 6.2.5 Study location and time

User testing was carried out during February 2014 in the Waikato area. There was a total of 24 participants who between them, went through each of the five datasets, twenty times. Almost all participants were known to the researcher before the study.

## 6.3 RESULTS

Table 6.2 summarises the mean and standard deviation of the normalised NKTC between an algorithm and the participant's grouping/ordering of Semantic Web data. The table is organised by Algorithm, Summary Statistic, and Order. The Order is the sequence in which participants encounter documents in a data set. GPRank and ListAlg have no entries for the first document because their user models are empty until after the first document is grouped and ordered by the participant.

Algorithm	Summary Stat.	Order									
		0	1	2	3	4	5	6	7	8	9
NonLearner	Mean	0.695	0.674	0.646	0.648	0.643	0.672	0.661	0.655	0.646	0.676
	StdDev	0.123	0.078	0.093	0.093	0.080	0.080	0.100	0.086	0.101	0.081
ListAlg	Mean		0.909	0.948	0.948	0.954	0.953	0.959	0.958	0.959	0.956
	StdDev		0.109	0.060	0.063	0.055	0.061	0.052	0.060	0.057	0.053
GPRank	Mean		0.909	0.957	0.961	0.963	0.964	0.966	0.968	0.965	0.973
	StdDev		0.109	0.059	0.052	0.050	0.064	0.050	0.051	0.052	0.043

Table 6.2: Mean and Standard Deviation of Normalised Kendall Tau Closeness by algorithm and sequence for the second user study

Table 6.3 shows the mean NKTC measures by data set so that it is possible to see if dataset diversity affects the ability to learn user preferences. The results are separated into Algorithm, Dataset and then in columns by the sequence in which the document is encountered. There is no value in the first column for ListAlg and GPRank because their user models are empty until after the participant has supplied grouping/ordering information by dragging and dropping triplets in the first document they encounter.

Algorithm	Data-set	Order									
		0	1	2	3	4	5	6	7	8	9
NonLearner	Movies	0.724	0.696	0.675	0.697	0.679	0.691	0.692	0.697	0.694	0.704
	Events	0.613	0.654	0.639	0.616	0.608	0.664	0.622	0.625	0.623	0.658
	Leaders	0.688	0.650	0.619	0.633	0.634	0.658	0.673	0.645	0.615	0.657
	Places	0.705	0.672	0.595	0.609	0.619	0.645	0.599	0.588	0.584	0.652
	Books	0.718	0.673	0.665	0.653	0.644	0.669	0.687	0.683	0.679	0.679
ListAlg	Movies		0.928	0.972	0.970	0.971	0.975	0.981	0.991	0.981	0.979
	Events		0.852	0.940	0.924	0.936	0.941	0.956	0.920	0.928	0.938
	Leaders		0.916	0.949	0.971	0.967	0.967	0.971	0.979	0.974	0.977
	Places		0.903	0.935	0.912	0.930	0.927	0.930	0.943	0.945	0.940
	Books		0.942	0.939	0.958	0.955	0.949	0.953	0.954	0.962	0.943
GPRank	Movies		0.928	0.985	0.976	0.971	0.982	0.979	0.993	0.987	0.982
	Events		0.851	0.940	0.950	0.951	0.930	0.969	0.943	0.939	0.967
	Leaders		0.916	0.962	0.979	0.975	0.988	0.984	0.992	0.978	0.986
	Places		0.903	0.943	0.933	0.950	0.933	0.932	0.937	0.950	0.961
	Books		0.942	0.950	0.960	0.960	0.982	0.960	0.970	0.969	0.966

Table 6.3: Mean Normalised Kendall Tau Closeness by Algorithm, Dataset, and Sequence for the second user study

The testing interface also counted the number of drag and drop operations performed by participants for each document they encountered. It is expected that user familiarity with the task of dragging and dropping to express grouping/ordering preferences will cause this number to decline over time. However, some of this decrease is also attributable to ListAlg and GPRank learning user preferences. Table 6.4 shows the mean number of drag and drop operations to transform a document to a participant's preferred grouping/ordering from an algorithm's prediction. The Order is the number of documents in a dataset encountered by the participant. The table is by dataset and order.

Algorithm	Dataset	Order									
		0	1	2	3	4	5	6	7	8	9
ALL	Movies	23.63	10.05	3.11	4.32	2.00	1.32	2.05	0.74	2.47	1.37
	Events	17.70	12.90	7.20	5.90	4.85	4.60	3.30	3.85	4.85	3.75
	Leaders	20.81	13.95	6.33	4.29	3.57	3.76	3.48	3.24	2.48	2.81
	Places	17.29	11.24	8.90	7.76	9.38	7.76	7.81	8.19	8.76	7.67
	Books	22.05	11.00	6.57	4.05	3.52	2.67	3.76	2.81	2.38	2.24
	Mean	20.30	11.83	6.42	5.26	4.66	4.02	4.08	3.77	4.19	3.57

Table 6.4: Mean count of drag and drop operations by dataset and sequence in the second user study

Table 6.5 shows the percentage of times each algorithm is selected as the best representation of a participant's grouping/ordering preferences in a blind selection. The table is organised by algorithm and sequence. The both option denotes that the grouping/ordering of ListAlg and GPRank were either the same or the user chose to say they were equally good. There is no algorithm choice before the first document is displayed because ListAlg and GPRank have empty user models.

Algorithm	Dataset	Order									
		0	1	2	3	4	5	6	7	8	9
Both	ALL		100	54.81	46.67	42.72	36.63	34.65	38.24	33.33	30.69
ListAlg			0.00	10.58	19.05	15.53	17.82	24.75	19.61	26.47	18.81
GPRank			0.00	34.62	34.29	41.75	45.54	40.59	42.16	40.20	50.50

Table 6.5: Percentage of times participant choose each algorithm by algorithm and sequence in the second user study

Table 6.6 shows the amount by which the algorithm that was not selected as best by the participant differs from the algorithm the participant selected as having the best grouping/ordering. The values are the Normalised Kendall Tau Distance multiplied by one hundred so that the values can be read as a percentage difference. The values can be interpreted as a percentage of predictive advantage compared to the other algorithm. Higher values represent an algorithm that is closer to correct when it is not selected.

Algorithm	Dataset	Order									
		0	1	2	3	4	5	6	7	8	9
ListAlg			1.0	4.5	5.4	4.1	3.9	3.8	4.5	4.4	4.7
GPRank			0.0	4.6	3.2	5.0	3.6	3.7	4.9	4.7	3.8

Table 6.6: Predictive Disadvantage for Algorithm and Sequence in the second user study

### 6.3.1 Performance of NonLearner

NonLearner is used when no user preference information is available. NonLearner's predictive performance is silently measured for NKTC alongside ListAlg and GPRank throughout all participant interactions during this study. Since NonLearner does not learn, its performance will not change as the participant provides more preference information, so overall summary statistics are shown instead of a accuracy broken into the number of SSRGs encountered.

Table 6.7 summarises all NKTC values for NonLearner compared with the participant's preferred grouping and order into quartiles.

Min-0%	Quartile 1-25%	Median-50%	Quartile 3-75%	Max-100%
0.58	0.62	0.66	0.68	0.72

*Table 6.7: Quartiles for the NonLearner's Normalised Kendall-Tau Closeness in the second user study*

Randomised groups and orders will tend towards an NKTC of 0.5 and higher values represent groupings and ordering that better reflects user preferences. NonLearner scores a median of 0.66 with a range from 0.58 to 0.72 and an inter-range gap of 0.14. The performance of NonLearner is therefore 0.66 with a tolerance of  $\pm 0.08$ . The worst performance of NonLearner is greater than random (0.5) by at least the tolerance of  $\pm 0.08$  so there is evidence to support a claim that NonLearner consistently performs better than random.

### 6.3.2 ListAlg and GPRank Learning

If ListAlg and GPRank successfully learn user preferences, then their performance should become better than both random (NKTC > 0.5) and NonLearner as participants supply more preference information to the user models. Figure 6.9 shows the change in the mean NKTC from all participants for both ListAlg and GPRank as more documents are encountered. A curved trend line is used to indicate change over time. From Figure 6.9, both GPRank and ListAlg are higher than random (NKTC > 0.5) and greater than NonLearner's maximum NKTC of 0.72). Additionally, the NKTC for GPRank and ListAlg increases from the first to second documents, the rate of increase slows until the eighth document, and from there NKTC diverges between GPRank and ListAlg. The improvements in NKTC indicate that ListAlg and GPRank are learning as more information about user preferences is available.

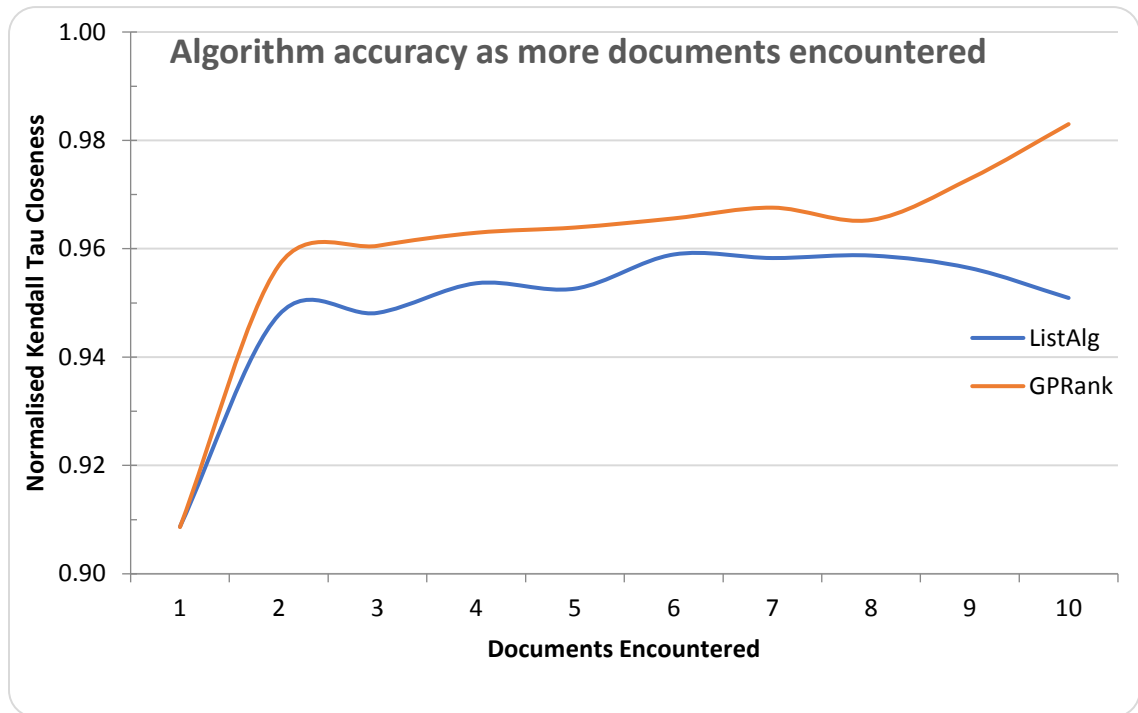


Figure 6.8: Algorithm Accuracy over time with +/- One Standard Deviation

Figure 6.10 plots the standard deviation of the accuracy of GPRank and ListAlg as the user model has data from more documents. If learning is occurring then, the standard deviation is expected to begin high and become lower. A higher standard deviation in NKTC indicates less consistent predictions while a lower standard deviation indicates that predictive accuracy is becoming more consistent. Figure 6.10 shows the standard deviations for GPRank and ListAlg begin high and become lower, and this indicates that predictive accuracy becomes more accurate as more documents are encountered. This is further indication that both GPRank and ListAlg learn.

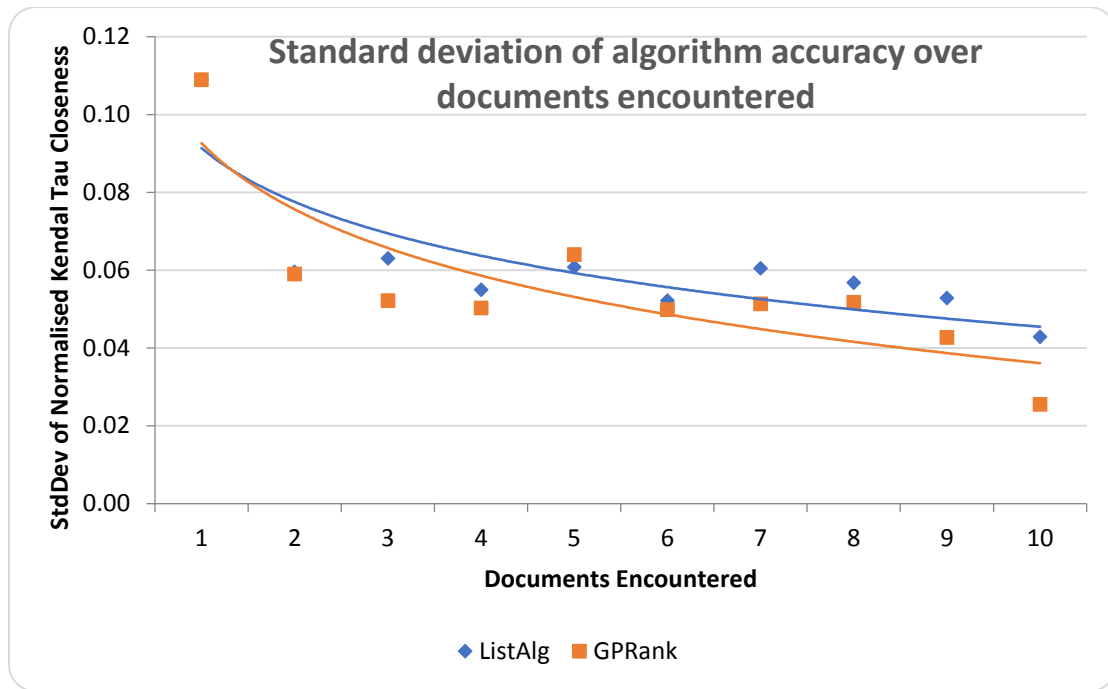


Figure 6.9: Plot showing the standard deviation of algorithm accuracy reduces as more user preference data is available

Another indicator that GPRank and ListAlg learn user preferences is that the number of drag-and-drop operations a participant uses to change from their selected closest algorithm's predicted grouping/ordering to their preferred grouping/ordering should also decrease over time. This is a less reliable indicator because the number of drag-and-drop operations might decrease because of user fatigue or the user developing more efficient strategies moving data into their preferred grouping and ordering. Figure 6.11 shows the mean number of drag and drop operations for all participants against the number of documents encountered. The chart begins with the first document for which ListAlg and GPRank propose grouping/ordering. A curved trend-line indicates that the number of drag-and-drop operations decreases over time and this decrease indicates that the algorithms have learned user preferences.

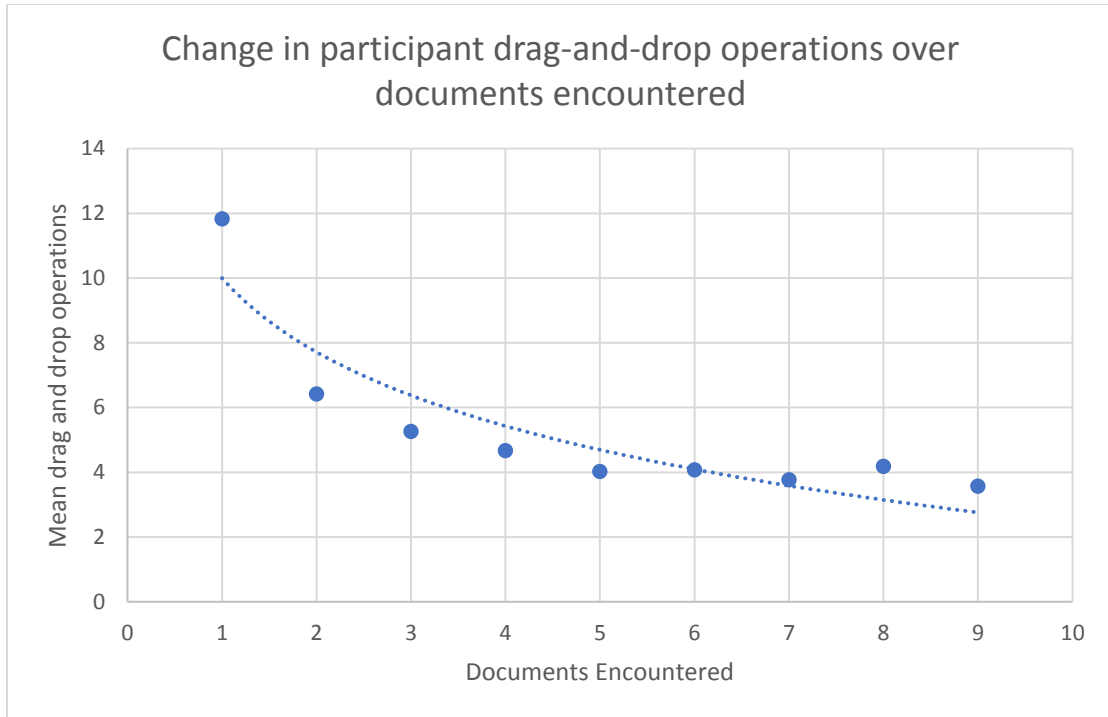


Figure 6.10: Chart of Change in Participant Drag-and-Drop Operations Over Time

ListAlg and GPRank both have NKTC values that begin above 0.9 and increase above 0.95 as more documents are encountered. The standard deviation of the NKTC also reduces over time which indicates that predictions by the algorithms become more consistent as the algorithms learn more. The overall number of user drag and drop operations also decreases over time.

Since both ListAlg and GPRank have an increase in mean NKTC, decreases in the standard deviation of the NKTC and decreases in the number of drag and drop operations over time, this indicates that ListAlg and GPRank learn user preferences for grouping/ordering Semantic Web data.



### 6.3.3 ListAlg and GPRank Performance versus NonLearner

NonLearner has a median NKTC of 0.66 and a maximum of 0.72. Once ListAlg and GPRank have data in their user models, then their lowest NKTC value (0.85 for ListAlg and GPRank) is higher than the maximum performance of NonLearner. Therefore, ListAlg and GPRank both perform better than the NonLearner.

GPRank increases NKTC accuracy quicker and levels off at a slightly higher plateau compared to ListAlg. Also, GPRank's results are more stable over time than ListAlg's because the range between  $\pm$  a single standard deviation are lower and continue to narrow.

While the performance results for ListAlg and GPRank are numerically close in terms of NKTC, participants can detect the difference between the two algorithms.

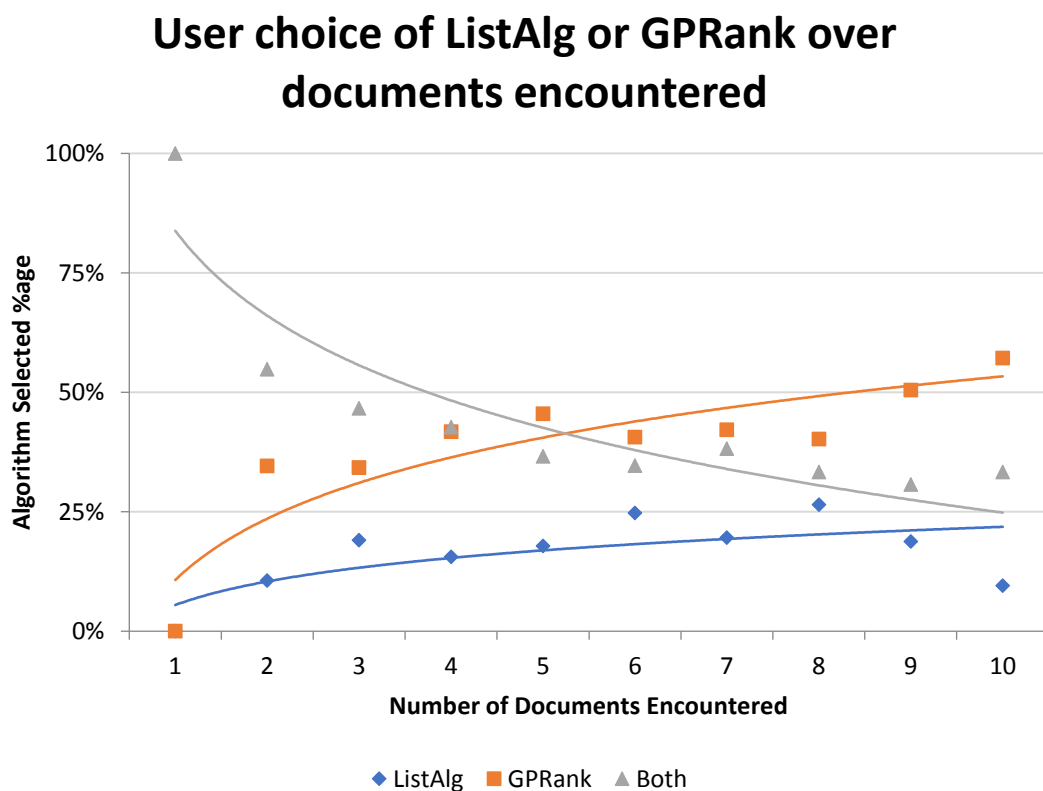


Figure 6.11: User Choice of Algorithm by Sequence in the second user study

Figure 6.12 plots the percentage of times that participants selected ListAlg, GPRank or “both are the same” (Y-axis) along with the number of documents encountered so far per participant (X-axis). An exponential trendline per algorithm is given to aid in understanding how user choice of algorithm changes as the more documents are encountered and the algorithms have therefore had more user preference data from which to learn user preferences.

On average, GPRank is favoured over ListAlg when users select algorithm in a blind, side-by-side choice. From the fourth document onwards GPRank and ListAlg are less likely to produce equivalent results, and GPRank leads above the choice of “both.” From the ninth document, this advantage has grown to the point where GPRank produces the preferred result more often than ListAlg and Both combined.

The NKTC between algorithm prediction and participant supplied data for both ListAlg and GPRank increases as participants progress through the data sets. Learning appears to follow an exponential pattern – though the test is too short to see if the algorithms reach a natural plateau in the accuracy each achieves.

The NKTC for NonLearner does not change. This is expected because NonLearner does not incorporate user preference information.

User operations are the number of drag-and-drop operations for the user to alter the best algorithm output into the user’s preferred grouping and ordering. The Trend of User Operations over time is the Log Estimate of the line slope that tracks User Operations as more documents are encountered. A smaller Trend of User Operations over time indicates that the user operations reduce much quicker as more documents are encountered. Figure 6.13 shows that datasets with higher predicate diversity per document take longer for the user operations to reduce.

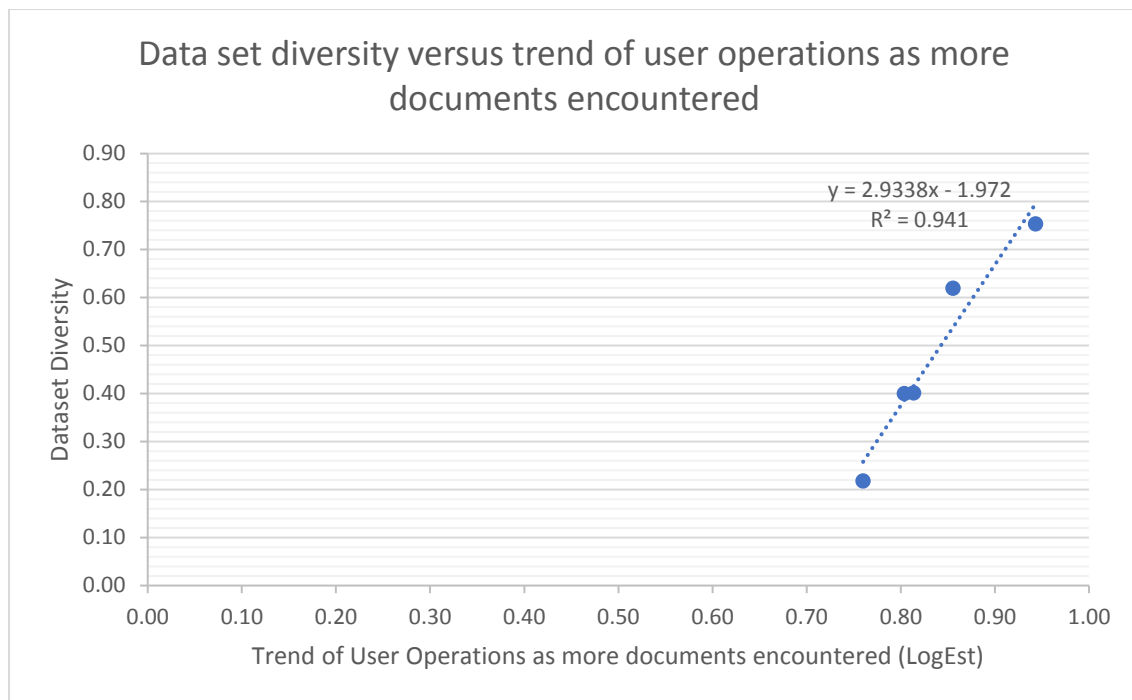


Figure 6.12: Trend of User Operations over time compared with dataset diversity in the second user study

The predictive disadvantage is the amount of error between the most accurate of ListAlg and GPRank and the least accurate. When plotted against data-set diversity (Figure 6.14) this suggests that ListAlg has a greater predictive disadvantage compared to GPRank as dataset diversity grows. In contrast, GPRank has an almost flat predictive disadvantage as dataset diversity grows. The means that GPRank is consistently more accurate than ListAlg, and GPRank is more accurate with greater dataset diversity.

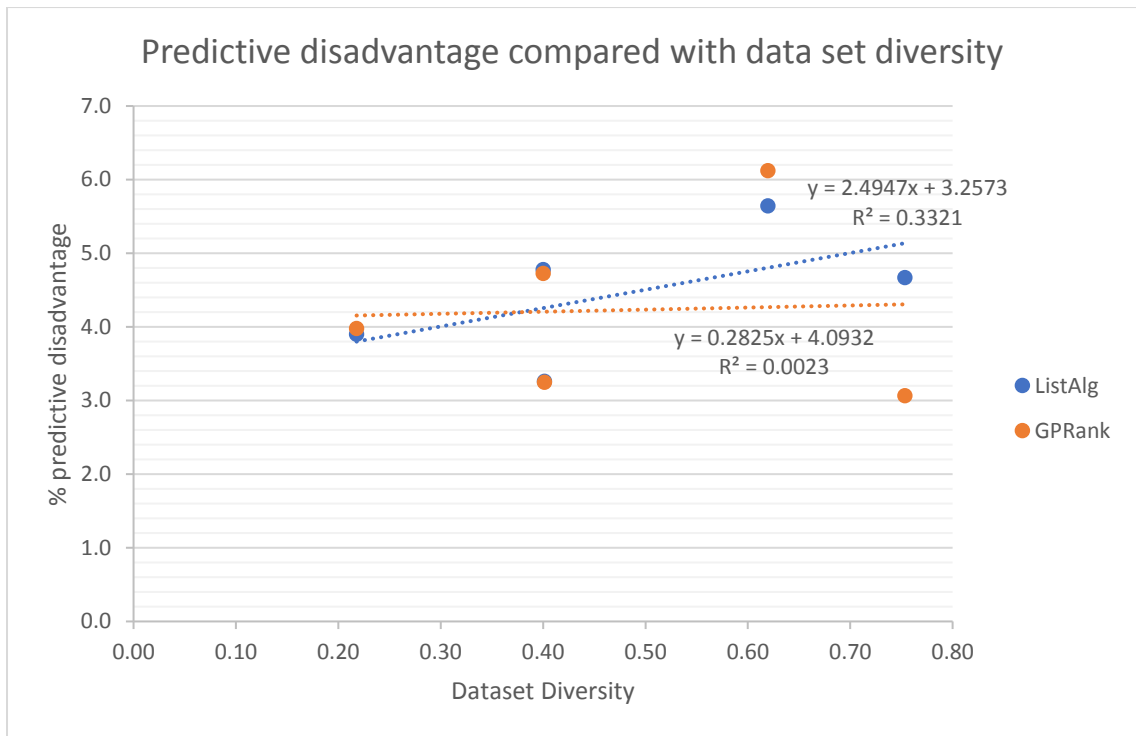


Figure 6.13: Percentage predictive disadvantage compared with dataset diversity

## 6.4 CONCLUSIONS AND SUMMARY

This section addresses the hypotheses from the start of the user study and then discusses these outcomes with reference to the first and second research questions.

NonLearner has a minimum performance (0.58) that is better than random (0.5) by at least some tolerance, here the gap between median and min/max performance. Therefore, H0 and H1 are rejected, and H2 is accepted.

H2: NonLearner is better than random grouping and ordering when compared to user's preferred grouping and ordering of Semantic Web data.

Both ListAlg and GPRank show increases in mean NKTC, decreasing in the standard deviation of the NKTC and decreasing in the number of drag and drop operations over time and this is evidence that ListAlg and GPRank learn user preferences for grouping/ordering Semantic Web data. Therefore, GL0 and GG0 are rejected, and GL1 and GG1 are accepted.

GL1: ListAlg does learn user preferences grouping/ordering Semantic Web documents.

GG1: GPRank does learn user preferences grouping/ordering Semantic Web documents.

The minimum NKTC of ListAlg and GPRank are both better than NonLearner's maximum NKTC. Therefore, JL0 and JG0 are rejected, and JL1 and JG1 are accepted.

JL1: ListAlg is more accurate than NonLearner when predicting user grouping/ordering preferences.

JG1: GPRank is more accurate than NonLearner when predicting user grouping/ordering preferences.

Although the predictive accuracy of GPRank and ListAlg are numerically close in terms of NKTC, users are quickly able to discern a difference. In a side-by-side blind selection, users quickly choose GPRank more often than ListAlg. GPRank is (on average) either equal or selected more often than ListAlg from the beginning, and GPRank's is selected more often as more documents are encountered. Therefore, there are grounds to reject K0 and K1 and accept K2.

K2: GPRank outperforms ListAlg when learning user preferences for grouping and ordering Semantic Web data.

The NonLearner, ListAlg and GPRank perform better than random (NKTC of 0.5). The two learning algorithms match user preferences to a greater degree than NonLearner (JL1, JG1).

The two learning algorithms learn user preferences and the evidence for this is that ListAlg's and GPRank's accuracy in predicting a user's preferences for grouping and ordering triples in an SSRG increases as more documents are encountered (GL1 and GG1).

GPRank more accurately predicts user preferences for grouping and ordering than ListAlg (K2). Also, GPRank has a lower predictive disadvantage meaning that GPRank predicts more consistently and GPRank's predictive disadvantage is less affected by dataset diversity than ListAlg.

The second research question is: *Can an adaptive interface learn, from a few interactions, user preferences for grouping and ordering displays of Semantic Web data?*

GPRank is an example of an adaptive user interface method that fulfils the requirements to answer this question in the affirmative. GPRank is the best choice (over NonLearner and ListAlg) for a method for an AUI that groups and orders triples from an SSRG because GPRank has the best ability to predict user preferences when encountering a new SSRG, has the best ability to learn user preferences for grouping and ordering and is less affected by dataset diversity.

This chapter presents evidence that GPRank is a good candidate for constructing grouped and ordered displays of Semantic Web data based on learning user preferences. The next chapter (below) tests whether a display of triples that are grouped and ordered by GPRank has speed and accuracy advantages over an Alphabetical ordered display for users selecting data on a screen.

## CHAPTER SEVEN USER STUDY III: GPRANK VERSUS ALPHABETICAL

### ORDERING FOR USER SPEED AND ACCURACY IN INFORMATION

#### RETRIEVAL TASKS

---

This chapter compares GPRank, an adaptive user interface method for grouping and ordering triples in an SSRG against Alphabetical ordering for information retrieval speed and accuracy.

This chapter directly addresses the third research question.

Alphabetical ordering is a suitable baseline for comparison with an AUI because Alphabetical ordering is used in many of the Semantic Web browsers reviewed in Chapter Three, and has broad usage in information systems in general as so is familiar to users.

The testing is a user study where users click on the answer to a question that is located on a single screen of data.

#### 7.1 HYPOTHESES

For this experiment, Information retrieval is the retrieval of a single data item from a screen display of data items without scrolling. This restricted definition is to reduce the timing overheads associated with searching for a Semantic Web document and navigating within that RDF Document. The hypotheses are grouped into speed and accuracy, each having a null hypothesis representing no detectable advantage to either algorithm and a hypothesis where GPRank has an advantage over Alphabetical order.

Are participants faster when carrying out information retrieval tasks from Semantic Web displays formed by Alphabetical ordering or GPRank?

H0: Grouping and ordering data for display using GPRank has no difference in information retrieval speed compared to Alphabetical ordering.

H1: Grouping and ordering data for display using GPRank has faster information retrieval than Alphabetical ordering.

Are participants more accurate when carrying out information retrieval tasks from Semantic Web displays formed by Alphabetical ordering or GPRank?

G0: Grouping and ordering data for display using GPRank has no effect on accuracy for information retrieval tasks compared to Alphabetical ordering.

G1: Grouping and ordering data for display using GPRank results in more accurate information retrieval than Alphabetical ordering.

## 7.2 METHODOLOGY

A useful test will emulate close to real-world usage scenarios. Semantic Web browsers are not commonly in use and programs that work with data (e.g. booking systems) are usually customised to suit the data schema and the workflows for their usage context. An imagined approximation of a future usage scenario is an information retrieval task in response to a question. In this situation, a user has results returned from a customised query. The unstable ontology concept means that the returned data may not follow a recognisable schema but will have a usable pattern of predicates shared across similar queries. This section discusses the experiment design and data gathering.



### 7.2.1 Experiment Design Considerations

The research assumes that scenario to be information lookup and measures this by having participants answer supplied questions. In a normal information lookup scenario, a user would expect to locate the document that held the answer. The experiment eliminates locating the document as a variable by supplying the Semantic Web document that contains the answer. Therefore, this experiment focuses on the retrieval of a triplet from a single screen display of an SSRG, without scrolling. The databank and question templates are available on Github at <https://github.com/Stormrose/GPRank>.

A test app (HTML/JS) allows participants to answer questions from supplied data by clicking on the answer. Participants are instructed to go as quickly and as accurately as they can because the time taken to find then click the answer and whether the question was answered correctly are both recorded. Participants answer enough questions so that there is sufficient data to establish reasonably accurate measurements of an individual's speed and accuracy for both Alphabetical ordered and GPRank displays.

### 7.2.2 Experiment Implementation

The testing application is built in HTML/JavaScript and delivered via a web server. The researcher briefed each participant in person. Either the researcher or the participant supplied the testing equipment, and so it is not possible to control for the differences in computer setups. Differences in computer setup will have affected the raw experiment results. The mouse moves differently depending on individual settings and screen size/resolution. However, the participants completed their contributions in single sessions. The effect is that raw speed is incomparable among participants without some form of normalisation. Timing is normalised using Wilcoxon-Mann-Whitney Rank Sum and accuracy is normalised using an accuracy difference between GPRank and Alphabetical order.

This test uses data from multiple topic-domains. The topic domains are the same as in the previous chapter; Movies, Tourist Destinations, Historical Events, Political Leaders, and Books.

This study reuses the Semantic Web data from Chapter Five (see 6.2.1) and extends the dataset with additional RDF documents from DBPedia. An HTML/JS based program is used to verify that the spread in dataset diversity. Dataset diversity is one minus the mean number of unique triplets per document divided by the total number of unique predicates in the document set. The dataset diversity in this user study (Table 7.1) is broadly the same as the data set used in Chapter Five (see Table 6.1) except that Political leaders dataset was increased in dataset diversity by 0.13 to give better coverage of the dataset diversity range.

<b>Dataset</b>	<b>Diversity</b>
Plants (training)	0.10
Movies	0.26
Historical Events	0.64
Political Leaders	0.53
Tourist Destinations	0.76
Books	0.41

*Table 7.1: Dataset Diversity for the third user study*

The experiment randomly generates questions from templates combined with the Semantic Web documents. In total, there are over 900 possible questions, of which an individual participant could encounter a maximum of 81 because participants can only encounter each SSRG once. Showing each SSRG only once means that all SSRGs remain are equally unfamiliar to the user.

Some questions include a word that matches the predicate in the answer and some questions do not. These questions are called Easy and Hard respectively. In the following example (Table

7.2), the word species is used in both the question “To which species does Banana belong?” and in the predicate label that indicates the answer “species.” The hard question does not include a word match between the question wording and the answer predicate. Solving hard questions requires more background knowledge from the participant. The analysis distinguishes between Easy and Hard questions to determine if GPRank or Alphabetical ordering have an advantage. GPRank may have an advantage if users group similar meaning predicates together.

Type	Example Question	Example Answer Predicate
Easy	To which species does Banana belong?	species
Hard	Where was Helen Clark educated?	alma mater

*Table 7.2: Easy/Hard Question examples for the third user study*

The selection of display algorithm (GPRank or Alphabetical ordering) is also randomised for each question to avoid bias that a specific order would introduce. There is no guarantee that an equal number of questions are attempted per participant and algorithm, so the results report the N sizes for both algorithms.

The results are saved to HTML5 LocalStorage in CSV format and then copied to a CSV file. The format of the save includes SetID, Participant ID and then a series of four values, one per question, representing: DocumentID, QuestionTemplateID, Algorithm (independent) and the dependent variables: Did the participant answer correctly and in what time measured in milliseconds.

The sequence of the experiment is Topic Domain Selection, Training Phase and Experimental Phase. An explanation of what occurs in each phase follows.

<i>Topic</i>	<i>Domain</i>	<i>Selection</i>
--------------	---------------	------------------

The participant selections a topic domain (Figure 7.1). This choice represents a Set of Semantic Web documents and questions.

### Choose an Area of Interest

---

Choose an area of interest from the choices below.

Movies
Historical Events
Political Leaders
Tourist Destinations
Books
End the Research Session

---

*Figure 7.1: Participant selects a topic domain*

<i>Training</i>	<i>Phase</i>
-----------------	--------------

The participant trains their group-order preferences for five documents so that GPRank is primed with some initial user preference data (Figure 7.2). Testing in Chapter Five showed that GPRank achieves good user preference tracking after five documents.

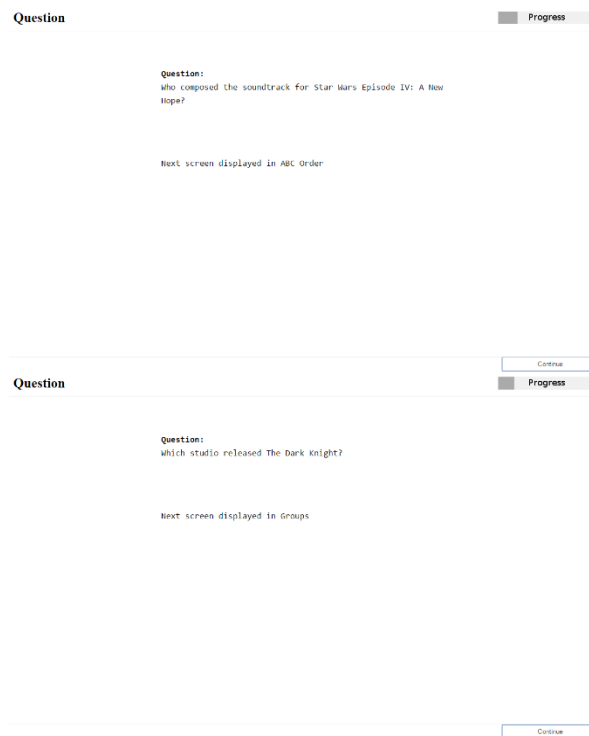
47 Ronin		
label	47 Ronin	
starring	Keanu Reeves, Hiroyuki Sanada	
director	Carl Rinsch	
comment	47 Ronin is a Japanese-American fantasy action ...	
released	6 December 2013	
language	English, Japanese	
runtime	118 minutes	
producer	Pamela Abdy, Eric McLeod	
cinematography	John Mathieson	
music	Tyler Bates	
writer	Chris Morgan, Hossein Amini	
editing	Stuart Baird	
distributor	Universal Pictures	
gross	\$151 million	
budget	\$175 million	
wordnet type	synset-movie-noun-1	
type	Film, Work, Creative Work, Thing, Movie	
subject	Japanese Fantasy Films, 2010s 3D Films, Samurai ...	

Continue

Figure 7.2: Training GPRank with User's preferences for grouping and ordering

7.2.2.1 Experimental Phase

First participants see the question screen (Figure 7.3). Participants are asked a question and told whether the next screen will be ABC of Learned (GPRank) orders. The participant clicks the next button when they are ready.



*Figure 7.3: Question screens with advance indication of the display format for the answering screen (Alphabetical ordering and GPRank)*

Then participants see the answer screen. The experiment repeats the question at the top of the screen. The data is shown in either Alphabetical ordering (Figure 7.4) or GPRank (Figure 7.5). Participants click what they believe is the correct answer. Timing and accuracy (in/correct answer) are recorded from the time the Answer Screen is shown until the click occurs.

## Who composed the soundtrack for Star Wars Episode IV: A New Hope?

Progress

### Star Wars Episode IV: A New Hope

budget	\$11 million
cinematography	British Society of Cinematographers, Gilbert Taylor
comment	Star Wars Episode IV: A New Hope, originally released ...
director	George Lucas
distributor	20th Century Fox
editing	Marcia Lucas, Paul Hirsch (film editor), Richard Chew
gross	\$775,398,007
label	Star Wars Episode IV: A New Hope
language	English
music	John Williams
producer	Gary Kurtz
runtime	121 minutes
starring	Alec Guinness, Carrie Fisher, Harrison Ford, Mark...
studio	Lucasfilm
subject	1970s science fiction films, 1977 films, 20th Century...
title	Star Wars Episode IV: A New Hope
type	Creative Work, Film, Movie, Movie CW, Thing, Work
wordnet type	synset-movie-noun-1
writer	George Lucas

Figure 7.4: Answering screen for ABC Ordering

## Which studio released The Dark Knight?

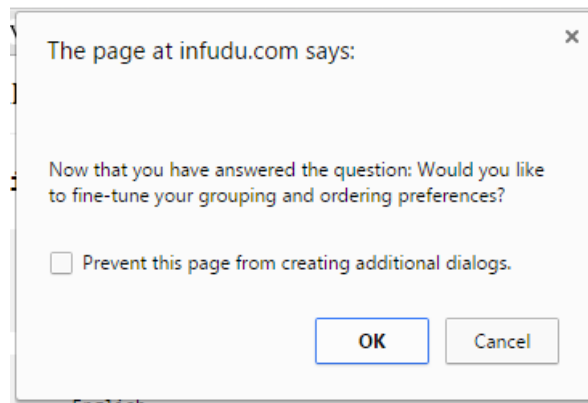
Progress

### The Dark Knight

label	The Dark Knight (film)
starring	Aaron Eckhart, Christian Bale, Gary Oldman, Heath Led...
director	Christopher Nolan
comment	The Dark Knight is a 2008 superhero film directed, prod...
language	English
country	United States
runtime	152 minutes
studio	DC Comics, Legendary Pictures, Syncopy Films
producer	Charles Roven, Christopher Nolan, Emma Thomas
music	Hans Zimmer, James Newton Howard
screenplay	Christopher Nolan, Jonathan Nolan
writer	Christopher Nolan, Jonathan Nolan
editing	Lee Smith (editor)
distributor	Warner Bros.
gross	\$1,004,558,444
budget	\$185 million
wordnet type	synset-movie-noun-1
type	2000s Action Films, 2008Films, American Action Films, ...
subject	2000s action films, 2008 films, American action films...
story	Christopher Nolan, David S. Goyer

Figure 7.5: Answering screen for GPRank format

If the answer screen was GPRank, then the participant is given an opportunity to alter grouping and orderings (Figure 7.6).



*Figure 7.6: After answering from a GPRank formatted screen, the participant is given the opportunity to adjust their group-order preferences.*

There is forced rest of a few seconds after several questions. The Rest Screen reminds participants that they can pause on any Question Screen, but not on Answer Screens.

The participant continues until they volunteer to end the set early or the set runs out of documents from which to generate questions. The experiment then shows the Topic Domain Selection Screen.

The experiment shows a progress bar always at the top right of the screen. Like previous experiments (see Chapters 4 and 6), participants have the flexibility on how long they spend doing this user test. Participants have the option to end their session at any time, after or during a set. Participants do not have to complete a set for their data to be recorded and useful: partial completions still yield useful data.



The data gathering was carried out in the third quarter of 2015. The researcher approached participants from among personal contacts. There were thirty participants.

A summary of the participant demographic attributes follows to illuminate any sampling biases that may have occurred. The study results are not analysed demographically. The demographic information includes Gender, Age bands, Ethnicity/s, First Languages, Profession, and Education. Participants are not required to respond to any questions and could supply multiple answers so the numbers of responses for each demographic question may not equal the number of participants. Ethnicity is categorised according to labels nominated by the participants themselves. The summary omits ethnicities with one response to protect the identity of the participants.

There were 13 female and 17 male participants. There is one participant aged 15-19, 17 participants aged 20-24, one participant aged 25-29, 4 participants aged 30-39, 2 participants aged 40-49, 2 participants aged 50-59, one participant aged 60-64 and one participant aged 65 or older. Participant ethnicity includes NZ European (8), European (5), New Zealander (4), Chinese (4), Maori (3), Pakeha (2), and three other participants, each from different ethnicities.

Twenty-four participants gave English as their first language. Three participants are first language Cantonese speakers and there are three participants each giving Chinese, French, or Spanish as their first language.

Participant professions include seventeen graphic designers, five educators, four computer scientists, twelve students and one retired person. Participant education levels included twelve students, four participants holding bachelors and six holding masters.

### 7.3 RESULTS

The results are processed in a combination of Sci-Py Python, R (R-Studio) and Excel because each made certain aspects of the analysis easier. The raw data has timing and correctness data for nearly 2000 questions answered, so this chapter shows only the aggregated results. Due to differences in testing computers, the results are not directly comparable between participants. Normalised measures by participant allow for comparison.

Wilcoxon-Mann-Whitney Rank Sum gives a p-value in the range [0.0 – 1.0] is used to compare the probability that timings with one algorithm are quicker than the other for each participant. High p-values (1.0) represent quicker times with Alphabetical ordering, while lower p-values (0.0) represent quicker times with GPRank. The amount the p-value diverges from 0.5 indicates certainty.

Accuracy Delta is the difference between an individual participant's accuracy under GPRank and Alphabetical ordering. The experiment calculates accuracy as the percentage of correct answers for each algorithm, and then the Accuracy Delta is calculated by subtracting the accuracy percentage for GPRank from the accuracy percentage for Alphabetical ordering.

The Participant IDs used here are not the same as the participant numbers used during data capture to protect participant privacy. The order of participants below is randomised so that it is different to the order of participation during data collection.

The following table (Table 7.3) shows timing data by participant and algorithm. Timing data is skewed to the right, so the median is a better measure of centrality than the arithmetic mean. The median is indicative only because centrality does not measure spread. The p-value column (Wilcoxon-Mann-Whitney p-value) is the measure of the probability that the participant performs faster with one algorithm than the other. The N column states the

numbers of questions answered by the participant given as the number for each algorithm and then the total.

Participant	ABC Median (ms)	Timing	GPRank Median (ms)	Timing	p-value for H0/H1	N (nABC nGPRank = N)	+
u01	2512		2275		0.077	40+35 = 75	
u02	4284		3523		0.340	20+25 = 45	
u03	2465		1793		0.010	36+30 = 66	
u04	3540		3431		0.073	43+38 = 81	
u05	4705		3377		0.069	13+18 = 31	
u06	2296		2924		0.843	34+47 = 81	
u07	7720		15945		0.991	38+28 = 66	
u08	2343		1886		0.104	11+14 = 25	
u09	2152		1963		0.386	33+27 = 60	
u10	3352		2446		0.072	21+20 = 41	
u11	2924		2508		0.292	42+39 = 81	
u12	3112		3681		0.932	42+39 = 81	
u13	2052		2200		0.196	28+12 = 40	
u14	1760		1820		0.440	45+36 = 81	
u15	3408		3205		0.346	27+24 = 51	
u16	2048		1889		0.266	14+26 = 40	
u17	2624		1943		0.076	43+38 = 81	
u18	2968		3066		0.436	45+36 = 81	
u19	2454		3277		0.927	26+20 = 46	
u20	6607		4228		0.024	30+21 = 51	
u21	2480		2333		0.236	41+40 = 81	
u22	3603		2525		0.000	34+47 = 81	
u23	3599		2770		0.007	43+38 = 81	
u24	1693		1669		0.468	40+41 = 81	
u25	3291		2515		0.024	40+41 = 81	
u26	3183		2618		0.214	42+39 = 81	
u27	2952		3122		0.468	29+31 = 60	
u28	1511		2065		0.761	41+40 = 81	
u29	1791		1974		0.447	36+45 = 81	
u30	1400		1766		0.955	43+38 = 81	

Table 7.3: Time to answer question by Participant and Algorithm in the third user study

Table 7.4 shows the per participant, per algorithm accuracy when retrieving information displayed in Alphabetical order or GPRank. The Delta column is the GPRank accuracy minus the Alphabetical order accuracy. The N column is the number of questions answered by each participant. The N column contains the Alphabetical order N, the GPRank N, and a total N.

Participant	ABC Accuracy (%)	GPRank Accuracy (%)	Accuracy Delta for G0/G1	N (nABC + nGPRank = N)
u01	100%	97%	-3%	40+35 = 75
u02	95%	88%	-7%	20+25 = 45
u03	83%	97%	13%	36+30 = 66
u04	91%	92%	1%	43+38 = 81
u05	69%	94%	25%	13+18 = 31
u06	85%	77%	-9%	34+47 = 81
u07	92%	93%	1%	38+28 = 66
u08	82%	64%	-18%	11+14 = 25
u09	85%	96%	11%	33+27 = 60
u10	90%	100%	10%	21+20 = 41
u11	90%	95%	4%	42+39 = 81
u12	95%	92%	-3%	42+39 = 81
u13	89%	100%	11%	28+12 = 40
u14	89%	97%	8%	45+36 = 81
u15	93%	88%	-5%	27+24 = 51
u16	93%	88%	-4%	14+26 = 40
u17	84%	95%	11%	43+38 = 81
u18	82%	83%	1%	45+36 = 81
u19	96%	90%	-6%	26+20 = 46
u20	83%	86%	2%	30+21 = 51
u21	95%	98%	2%	41+40 = 81
u22	88%	91%	3%	34+47 = 81
u23	95%	92%	-3%	43+38 = 81
u24	98%	90%	-7%	40+41 = 81
u25	95%	90%	-5%	40+41 = 81
u26	83%	77%	-6%	42+39 = 81
u27	97%	100%	3%	29+31 = 60
u28	85%	90%	5%	41+40 = 81
u29	94%	93%	-1%	36+45 = 81
u30	98%	95%	-3%	43+38 = 81

Table 7.4: Accuracy by Participant and Algorithm in the third user study

## 7.4 FINDINGS

The following graph (Figure 7.7) plots participants on with normalised values for which algorithm has an advantage for timing (Y-Axis) versus Accuracy (X-Axis). The p-value of ABC vs GPRank timings is used to provide a normalised measure of which algorithm has a speed advantage. A lower y-axis value represents timings in favour of GPRank while a high y-axis value represents timings in favour of Alphabetical ordering. The middle ground means that no determination of timing advantage can be made. The absolute middle (0.5) means that the data is indistinguishable from a coin-toss.

The X-Axis represents the relative accuracy advantage of GPRank compared to Alphabetical ordering. Sub-zero scores represent that Alphabetical ordering has an accuracy advantage for that particular participant, while positive X-Axis values represent an accuracy advantage for GPRank. The band between -10% is considered noise.

The trend line is the linear correlation between Algorithm timing advantage and Accuracy. The right-downwards slope indicates that participants are both more accurate and quicker with their preferred algorithm.

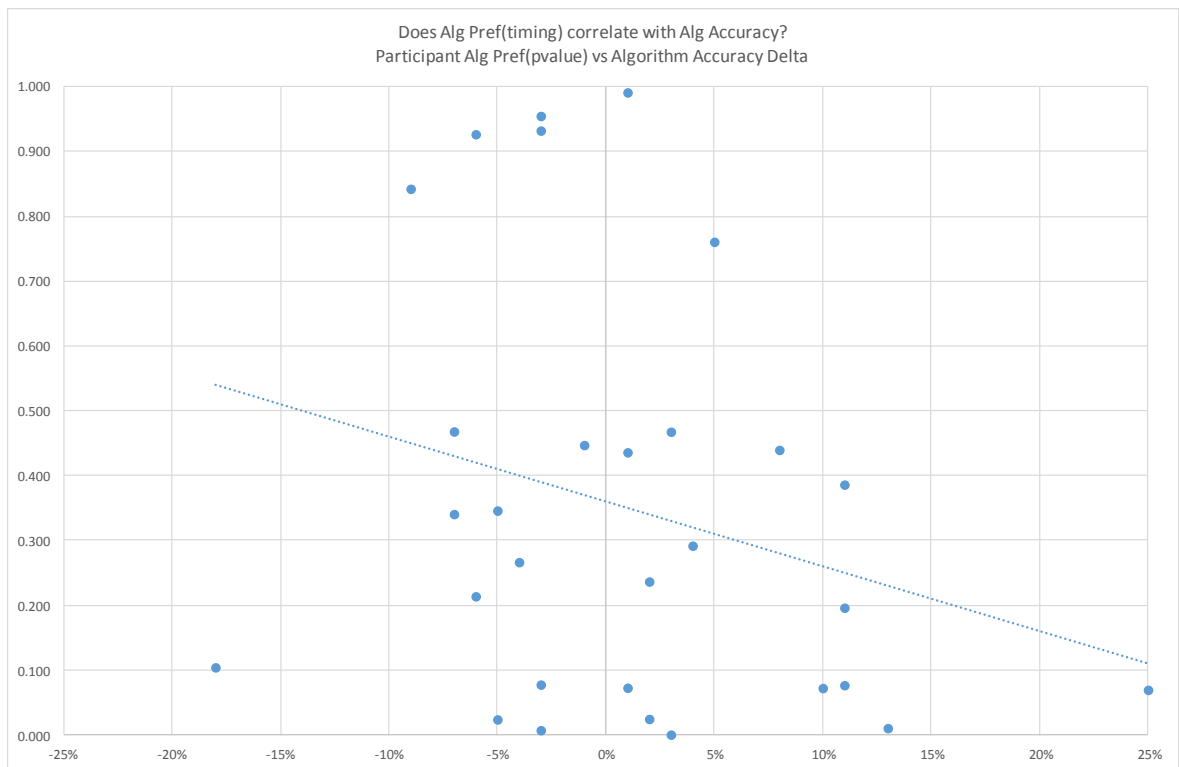


Figure 7.7: Chart plotting normalised speed advantage by Algorithm and Accuracy

From 30 participants, ten are faster with GPRank, and four are faster with Alphabetical ordering (10% p-values).

There is a band with no participants that lies between 0.5 to 0.75. The sample size is too small to investigate what this means without further research. There are 6 participants with scores above 0.75.

There are 24 participants lower than 0.5, of which 10 have a strong advantage with GPRank. This indicates that the remaining 14 participants may have a speed advantage with GPRank, but the speeds with either algorithm are close enough not to declare a winner. Participants in the 0.10 to 0.5 p-value range have close median times with either algorithm, so it is unlikely that display algorithm affects timing much for these participants.

Participants who have a timing advantage with GPRank have lower accuracy with Alphabetical Order than the converse: those who have a timing advantage with Alphabetical Order have accuracy less affected when encountering GPRank.

#### 7.4.1 GPRank may be more Accurate for Some Participants

The following graph (Figure 7.8) plots only the participants with a p-value of less than 0.1 or greater than 0.9 and removes the participant with a 25% accuracy advantage with GPRank as an outlier (25% is higher than the third quartile boundary plus 1.5 times the interquartile range of Accuracy Deltas).

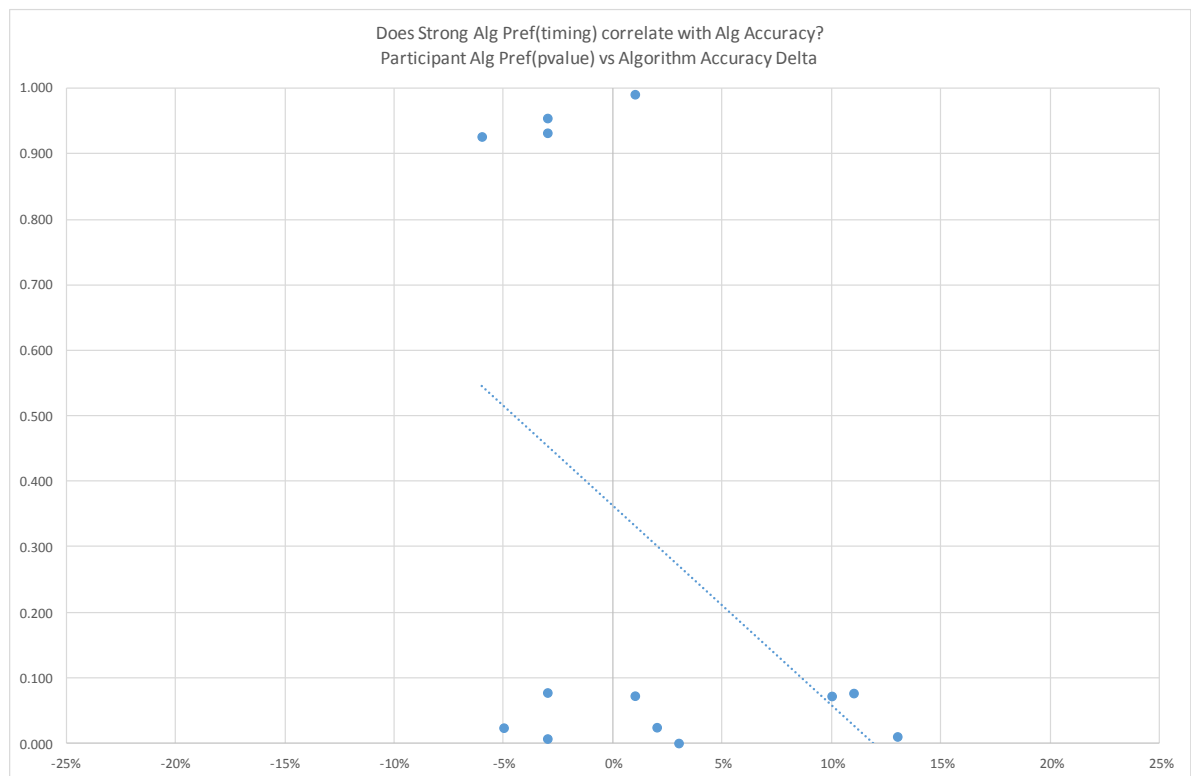


Figure 7.8: Chart plotting normalised speed and accuracy advantage. p-values <0.1 and >0.9. Accuracy outlier removed.

Ignoring the accuracies between -10% and +10% as noise, there is a cluster of three participants who have a significantly higher accuracy advantage with GPRank. However, the N value is too small to read much into this without further research.

## 7.4.2 Other Findings

While not part of the main research aims for this chapter, a large dataset contains other opportunities for quantitative exploration. This section contains analysis that is interesting but does not directly answer the hypothesis.

### 7.4.2.1 *The Effect of Set Diversity*

Set diversity is a measure of how diverse the predicate labels are within a document set. The dataset diversity formula is the same as that in the previous chapter. The calculation for set diversity is one minus the mean number of unique predicates per document divided by the total number of unique predicates in a set. The number is in the range [0.0 – 1.0) with 0.0 representing the least diverse set (the same predicates in all documents) and 1.0 representing the most diverse set (no predicates are repeated in between documents).

There is no linear correlation (Pearson) between set diversity and time to answer. There was a moderate correlation (0.49) in incorrect answers as set diversity increased. There was no significant difference between GPRank and Alphabetical ordering when correlating time to answer with set diversity. Set diversity has a small (-0.27) negative correlation with accuracy as set diversity increased, but there was no difference between the accuracy of GPRank and Alphabetical ordering. (see Table 7.5)



Set Diversity Pearson Correlation vs	ABC Ordering	GPRank	Combined
Time to answer correctly	*	*	+0.04
Time to answer incorrectly	*	*	+0.49
All Times to Answer	+0.06	+0.12	+0.09
Accuracy	-0.27	-0.27	-0.27

\* times to answer in/correctly not calculated per algorithm

Table 7.5: Correlation between Dataset Diversity and answering times in the third user study

#### 7.4.2.2 The Effect of Easy / Hard Questions

Easy questions contain the answer's predicate label in the question text. Hard questions do not contain the answer's predicate label in the question text. Hard questions take longer to answer and have approximately 5% lower accuracy, but the results are similar for both GPRank and Alphabetical ordering (Table 7.6).

Question Type	ABC Ordering (timing mean ms)	GPRank (timing mean ms)	ABC Ordering (accuracy %)	GPRank (accuracy %)	N (nABC + nGPRank = N)
Easy	3747	3624	92%	93%	636 + 627 = 1263
Hard	5049	5285	86%	87%	370 + 341 = 711

Table 7.6: The effect of Easy/Hard question on Answer Time and Accuracy by Algorithm in the third user study

#### 7.4.2.3 The Effect of Participants Learning the Task

Participants may become more accustomed to a task as the user test progresses. Document order, question order, and the ordering in with the two display algorithms were used are all

randomised to reduce any bias that becoming accustomed to the task may introduce. However, viewing this information by algorithm may show whether GPRank or Alphabetical ordering become more learnable for users.

The following graph (see Figure 7.9) shows that participant response times for correct answers improve the more questions a participant answers. GPRank and Alphabetical ordering have a similar rate of improvement.

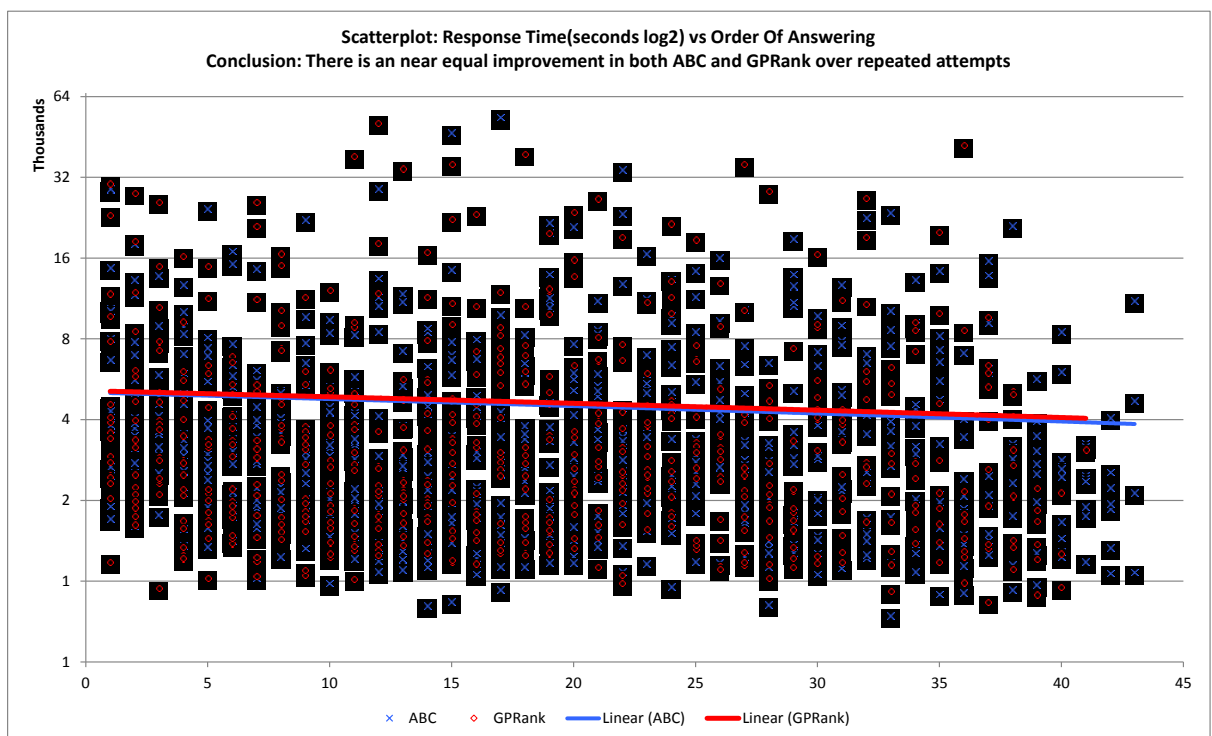


Figure 7.9: Response Times vs Order of Answering by Algorithm

There was no effect on accuracy as users progressed through the questions for either GPRank or Alphabetical ordering (Table 7.7). Accuracy overall appears to be unaffected by the length of the study (up to 81 questions per participant).

Alphabetical Order	-0.02
GPRank	-0.03

*Table 7.7: Correlation [-1,+1] of Algorithm Accuracy to Cumulative Questions Answered by Algorithm in the third user study*

## 7.5 CONCLUSIONS

From thirty participants, one-third show a speed advantage with GPRank, and four have a speed advantage with Alphabetical ordering. There are 20% ( $10 - 4 = 6$ ) more participants that have a speed advantage with GPRank compared to Alphabetical ordering. Therefore, H0 is rejected, and H1 is accepted.

H1: Grouping and ordering data for display using GPRank results in faster information retrieval than Alphabetical ordering.

From thirty participants there are four (inc. one outlier) that show an accuracy advantage with GPRank over Alphabetical ordering. Participants tended to be more accurate with the algorithm with which they were also fastest. Therefore, there are no adequate grounds to reject the null hypothesis G0 and accept G1.

G0: Grouping and ordering data for display using GPRank has no effect on accuracy for information retrieval tasks compared to Alphabetical ordering.

As a set becomes more complex then times to answer incorrectly increase. There is a weak correlation that indicates accuracy decreases as set diversity increases. Both GPRank and Alphabetical ordering perform similarly in this regard.

Hard questions take longer to answer and are have ~5% lower accuracy. GPRank and Alphabetical ordering perform similarly for timing and accuracy under Easy/Hard questions.

Participants become quicker at answering questions, and their accuracy is unaffected over the duration of an 81 questions test.

The accepted hypotheses are:

H1: Grouping and ordering data for display using GPRank has faster information retrieval than Alphabetical ordering.

G0: Grouping and ordering data for display using GPRank has no effect on accuracy for information retrieval tasks compared to Alphabetical ordering.

Since GPRank, compared to Alphabetical ordering, enables more participants to be faster with equivalent accuracy, then GPRank is a better alternative than Alphabetical ordering for forming displays of Semantic Web data.

The next chapter examines the conclusions from this chapter in the context of the wider thesis.

## CHAPTER EIGHT SUMMARY AND CONCLUSIONS

---

This chapter discusses the research considering the overall research aims. The chapter begins with a restatement of the research questions and moves onto a summary of the research. After the summary is a discussion of the findings and implication of the research. The chapter ends with recommendations for application of the findings and suggestions for future research.

This thesis examines if a Semantic Web browser with an adaptive user interface may improve speed and accuracy information retrieval tasks. The Semantic Web presents a unique challenge for the display of data because the Semantic Web does not have fixed ontologies and users may have different preferences for how data is grouped and ordering when displayed. The hypothesis is that:

*A Semantic Web browser with an adaptive user interface that groups and orders data has speed and accuracy advantages in information retrieval tasks.*

We address the thesis by answering the following questions:

**Question 1.** *Is there sufficient diversity in user preferences for displaying Semantic Web data to justify the overhead of an adaptive user interface that learns how to group and order?*

The first user study (see Chapter Four) investigates the inter-rater agreement for participants who are asked to rank the relatedness of pairs of Semantic Web triples. The results are that participants have a moderate amount of agreement about the relatedness of pairs of triples and this indicates that user preferences for grouping and ordering data are diverse enough to justify the overhead of an adaptive user interface.

**Question 2.** *Can an adaptive interface learn, from a few interactions, user preferences for grouping and ordering displays of Semantic Web data?*

Chapter Five describes two adaptive user interface methods, ListAlg and GPRank, that are subsequently tested in the second user study in Chapter Six. Both ListAlg and GPRank are capable of learning user preferences for grouping and ordering Semantic Web data. GPRank is more accurate at predicting user preference than GPRank. Although the absolute difference is slight, participants reliably select GPRank's predictions over ListAlg in a blind selection.

**Question 3.** *Do users perform single screen search tasks quicker and more accurately with an adaptive user interface that groups and orders Semantic Web data or with data in alphabetical order?*

The third user study in Chapter Seven tests whether users are faster and more accurate finding the answer to a question in a screen of data arranged using an adaptive user interface based upon GPRank or with Alphabetical ordering. Some participants are faster with GPRank, and the users that are faster with Alphabetical ordering are not much slower when using GPRank. There are a smaller group of users who perform information retrieval tasks more quickly with Alphabetical ordering. There is no difference in the accuracy of users performing information retrieval tasks when using GPRank or Alphabetical ordering.

## 8.1 CONCLUSIONS

For about a quarter of users, GPRank (an adaptive user interface algorithm) resulted in increased speed in information retrieval tasks compared to Alphabetical ordering. For most users, the performance was about the same. Users were no less accurate using GPRank than when using Alphabetical ordering.

The performance of GPRank is then evidence that an adaptive user interface that groups and orders data can improve speed in information retrieval tasks. Some users are faster with GPRank and no less accurate than Alphabetical ordering. Therefore, the hypothesis is accepted: A Semantic Web browser with an adaptive user interface that groups and orders data has improves speed in information retrieval tasks.

The user tests (Chapters Six and Seven) use single screens of data that did not scroll. It is not known how GPRank and Alphabetical ordering will perform relative to each other if the display includes scrolling.

The second and third user studies use Semantic Web data from five topic areas, and the results might not be generalisable outside those topics.

There are advantages for more users in using AUIs based upon GPRank and using GPRank has few disadvantages for those more used to Alphabetical Ordering. For those that prefer alphabetical order, the drag-and-drop system used to express user preferences for grouping and ordering can detect when a user is consistently arranging data in alphabetical order and offer to always to do alphabetical ordering.

GPRank makes decisions based on pairs of triplets. GPRank works when the assumptions underlying the unstable ontologies hold. The key assumption is that predicates within an SSRG are not reliably predictable until the SSRG is retrieved. Schema information – such as that contained in `rdf:type`, RDFS, and OWL – cannot be relied upon to be present or accurate. Resolving schema information may take several slow network transactions.

This research views unstable ontologies as a fundamental property of the Semantic Web, and therefore GPRank is suitable within this scope. If there is a more limited situation where Semantic Web technology is used for a dataset with high data quality, enforced schema and a narrow set of user goals, then other user interfaces may be more appropriate than an AUI

based upon GPRank. In that situation, the ontologies are stable, and the dataset is more like a traditional database though delivered on the Semantic Web platform. In this situation, templating or traditional database forms are probably a better approach. The research places this situation out of scope, but it is discussed here so that the limitations of GPRank are properly defined.

## 8.2 RECOMMENDATIONS FOR FUTURE RESEARCH

During this research area for further investigation were discovered. This section summarises avenues for future research.

GPRank might be generalisable to any situation with supervised learning for grouping and ordering with pairwise, partial orders. The implementation in this research focuses on grouping and ordering predicates for forming displays of Semantic Web data. Whether GPRank is generalizable can be the subject of further research.

GPRank was not user tested on scrolling displays. A future investigation could test the effects of scrolling displays on speed and accuracy for information retrieval tasks.

GPRank has a higher computational cost per iteration, and its user model is larger than ListAlg. It is possible to improve GPRank's efficiency by looking for fragments for which ListAlg's assumptions hold (the user prefers the same grouping and order all the time) and then treating those fragments as if they were a single item within GPRank. This should be a simple extension for ordered predicates within a single group. However, the approach could be extended to include groups and their members also. A successful implementation would improve the efficiency (space and computation) of GPRank.

If ListAlg's assumptions hold for parts of the group/ordering preferences for individual users, then it follows that the predicate patterns might be stable enough to support a partial-



templating system. Further research could look at mixing partial-templates into a GPRank created display. Since templates represent a fixed perspective on how data should be displayed, then there is the danger that the template enforces its form of life on the user. Research in this direction should carefully balance the benefits of templates versus the potential imposition of a dominant form of life: user preferences should be easily expressible and take priority.

GPRank weights group affinity by the number of confirmations. The strength of the weighting becomes stronger as more confirmations are received. GPRank currently caps the strength at six confirmations based on a preliminary investigation by the researcher. However, the exact tuning of confirmation based weighting should be the subject of more robust investigation.

ListAlg and GPRank appear to plateau following an exponential trend. However, the user study in Chapter Five (above) was too short to determine if the trend continues. A user test with more documents per participant will show if GPRank outperforms ListAlg in terms of the predictive accuracy of the algorithm as measured by Normalised Kendall Tau Distance.

There is a missing third quartile in the p-values for algorithm timing in the third user study (7 above). Care should be taken not to infer too much from this, but further research could determine if this indicates that preference for Alphabetical ordering is a learned behaviour.

There was a cluster of three participants in the third user study (Chapter 7 above) for whom GPRank gives a significant accuracy advantage over Alphabetical ordering. Further research with more participants could identify if this is a distinct group of users and look at the composition of that group of participants.

GPRank's user model has the same boot-strapping problem common to all algorithms that learn from user preferences. The boot-strapping problem is because user preferences are unknown in the beginning. Addressing the boot-strapping problem could be the subject of

further exploration. This research proposed uses the NonLearner algorithm when the user model is empty. NonLearner performs at 0.6, which is better than random (0.5) and so any bootstrapping approach has a low bar to outperform NonLearner.

The field of social recommenders/collaborative filtering could provide bootstrapping information based on the preferences of other users. Incorporating the preferences of others can become implied templating by the masses, and so this researcher urges caution that social filters are sophisticated enough that grouping/ordering decisions respect plurality in the forms of life that these decisions embody.

GPRank tracks grouping and ordering decisions only. The research assumed that filtering decisions – deciding which triplets do not get shown – is made before GPRank is activated. Filtering may also remove triplets that are redundant; that is, the triplet contains information that is already in another triplet. Filtering redundant triplets can result in a situation where the filtering process removes triplet-A because it is redundant compared to triplet-B and the user model has no knowledge of triplet-B but does have a knowledge of triplet-A. More research could investigate ways to allow the display algorithm to utilise its knowledge of triplet-A to inform display decisions about triplet-B until there is explicit information for triplet-B. This specific situation is distinct to the partial bootstrapping problem because there is already potentially useful information in the user model.

GPRank incorporates new user preference information from a grouped and ordered display without any regard for the actions a user took to group and order that display. The effect of this is that Group/Ordering acts of commission are weighted equally to acts of omission. If a user is unsatisfied with a grouping and ordering but does not act to correct it, then GPRank will learn an incorrect grouping and ordering. Further research could be done to see if user actions that are explicit (e.g. reordering two triplets) should have more effect on GPRank's learning than when the user has not acted.

## REFERENCES

---

- Alshukaili, D., Fernandes, A. A., & Paton, N. W. (2016). Structuring Linked Data Search Results Using Probabilistic Soft Logic. In *International Semantic Web Conference* (pp. 3–19). Springer.
- Bayes, T., & Price, M. (1763). An Essay towards Solving a Problem in the Doctrine of Chances. *Philosophical Transactions of the Royal Society of London*, 53(0), 370–418.  
<https://doi.org/10.1098/rstl.1763.0053>
- Becker, C., & Bizer, C. (2009). Marbles linked data browser. Retrieved 23 November 2010, from <http://marbles.sourceforge.net/>
- Berners-Lee, T., Chen, Y., Chilton, L., Connolly, D., Dhanaraj, R., Hollenbach, J., ... Sheets, D. (2006). Tabulator: Exploring and analyzing linked data on the Semantic Web. In *3rd International Semantic Web User Interaction Workshop*. Retrieved from <http://swui.semanticweb.org/swui06/papers/Berners-Lee/Berners-Lee.pdf>
- Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web. *Scientific American*, May. Retrieved from <http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21>
- Bizer, C., & Gauß, T. (2007, January 15). Disco - Hyperdata Browser. Retrieved 23 November 2010, from <http://www4.wiwiss.fu-berlin.de/bizer/ng4j/disco/>
- Bréal, M. (1904). *Essai de sémantique:(science des significations)*. Hachette.
- Camarda, D., & Mazzini, S. (2012). *LodLive*. Javascript, HTML. Retrieved from <http://en.lodlive.it>
- Cheng, G., & Qu, Y. (2009). Searching linked objects with falcons: Approach, implementation and evaluation. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 5(3), 49–70.

- Dadzie, A.-S., & Rowe, M. (2011). Approaches to visualising linked data: A survey. *Semantic Web*, 2(2), 89–124.
- de la Flor, G. (2004). *User Modeling & Adaptive User Interfaces* (No. 1085). Institute for Learning & Research Technology (ILRT). Retrieved from [http://web.archive.org/web/20060304225843/http://www.ilrt.bris.ac.uk/publications/researchreport/rr1085/report\\_html](http://web.archive.org/web/20060304225843/http://www.ilrt.bris.ac.uk/publications/researchreport/rr1085/report_html)
- Dice, L. R. (1945). Measures of the Amount of Ecologic Association Between Species. *Ecology*, 26(3), 297–302. <https://doi.org/10.2307/1932409>
- Gutteridge, C. (2012). Quick and Dirty RDF browser (Version 1.5). Retrieved from <http://graphite.ecs.soton.ac.uk/browser/>
- Hu, P. J.-H., Ma, P.-C., & Chau, P. Y. (1999). Evaluation of user interface designs for information retrieval systems: a computer-based experiment. *Decision Support Systems*, 27(1), 125–143.
- Huynh, D. F., Karger, D. R., & Miller, R. C. (2007). Exhibit: lightweight structured data publishing. In *Proceedings of the 16th international conference on World Wide Web - WWW '07* (pp. 737–746). Banff, Alberta, Canada: ACM. <https://doi.org/10.1145/1242572.1242672>
- Johnson, C. M., Johnson, T. R., & Zhang, J. (2005). A user-centered framework for redesigning health care interfaces. *Journal of Biomedical Informatics*, 38(1), 75–87.
- Kaufmann, O., Lorenz, A., Oppermann, R., Schneider, A., Eisenhauer, M., & Zimmermann, A. (2007). Implicit interaction for pro-active assistance in a context-adaptive warehouse application. In *Proceedings of the 4th international conference on mobile technology, applications, and systems and the 1st international symposium on Computer human interaction in mobile technology - Mobility '07* (p. 729). Singapore. <https://doi.org/10.1145/1378063.1378187>

- Kendall, M. G. (1938). A New Measure of Rank Correlation. *Biometrika*, 30(1/2), 81.  
<https://doi.org/10.2307/2332226>
- Kirkpatrick, S. (1984). Optimization by simulated annealing: Quantitative studies. *Journal of Statistical Physics*, 34(5–6), 975–986.
- Krippendorff, K. (2004). *Content analysis: An introduction to its methodology*. Sage.
- Kules, B. (2000). User Modeling for Adaptive and Adaptable Software Systems. Retrieved 8 December 2010, from <http://www.otal.umd.edu/UUGuide/wmk/>
- Landis, J. R., & Koch, G. G. (1977). The measurement of observer agreement for categorical data. *Biometrics*, 159–174.
- Langley, P. (1999). User modeling in adaptive interfaces. In *UM '99: Proceedings of the seventh international conference on User modeling* (pp. 357–370). Secaucus, NJ, USA: Springer-Verlag New York, Inc.
- MIT. (2005). SIMILIE: Longwell RDF Browser. Retrieved 23 November 2010, from <http://simile.mit.edu/wiki/Longwell>
- Morris, C. (1946). Signs, language and behavior.
- OpenLink Software. (2009). Zitgist DataViewer. Retrieved 23 November 2010, from <http://zitgist.com/products/dataviewer/dataviewer.html>
- Oppermann, R. (1994). *Adaptive user support: ergonomic design of manually and automatically adaptable software*. CRC Press.
- Pietriga, E., Bizer, C., Karger, D., & Lee, R. (2006). Fresnel: A browser-independent presentation vocabulary for RDF. *The Semantic Web-ISWC 2006*, 158–171.
- Quan, D. A., & Karger, R. (2004). How to make a semantic web browser. In *Proceedings of the 13th conference on World Wide Web - WWW '04* (p. 255). New York, NY, USA.  
<https://doi.org/10.1145/988672.988707>

- Quan, D., Huynh, D., & Karger, D. R. (2003). Haystack: A platform for authoring end user semantic web applications. In *International Semantic Web Conference* (pp. 738–753). Springer.
- Quan, D., & Karger, D. R. (2004). Xenon: an Rdf Stylesheet Ontology.
- Rosenholtz, R., Twarog, N. R., Schinkel-Bielefeld, N., & Wattenberg, M. (2009). An intuitive model of perceptual grouping for HCI design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1331–1340). ACM. Retrieved from <http://web.mit.edu/rruth/www/Papers/RosenholtzEtAlCHI2009PO.pdf>
- Schneider-Hufschmidt, M., Malinowski, U., & Kuhme, T. (1993). *Adaptive user interfaces: Principles and practice*. Elsevier Science Inc.
- Seeliger, A., & Paulheim, H. (2012). A semantic browser for linked open data.
- Shadbolt, N., Berners-Lee, T., & Hall, W. (2006). The semantic web revisited. *IEEE Intelligent Systems*, 21(3), 96–101.
- Sørensen, T. (1948). A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons. *Kongelige Danske Videnskabernes Selskab*, 5(4), 1–34.
- Steer, D. (2003, January 28). BrownSauce: an introduction. HP Laboratory. Retrieved from <http://www.hpl.hp.com/techreports/2003/HPL-2003-10.pdf>
- Stegemann, T., Ziegler, J., Hussein, T., & Gaulke, W. (2012). Interactive construction of semantic widgets for visualizing semantic web data. In *Proceedings of the 4th ACM SIGCHI symposium on Engineering interactive computing systems* (pp. 157–162). ACM.
- Tummarello, G., Cyganiak, R., Catasta, M., Danielczyk, S., Delbru, R., & Decker, S. (2010). Sig. ma: Live views on the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8(4), 355–364.
- W3C. (2007, October 21). IsaViz: A Visual Authoring Tool for RDF. Retrieved 23 November 2010, from <http://www.w3.org/2001/11/IsaViz/>



## APPENDICES

---



## APPENDIX ONE: DEMOGRAPHIC QUESTIONNAIRE

All three user studies use the same demographic questionnaire. The questionnaire is on the next page.

# Demographic Questionnaire

**Project Title**

*<The title of the user study>.*

**Participant Number:**

Supplying any demographic detail is optional.

**Gender:**  
(Circle one)

Female

Male

**Age:**  
(Circle one)

15–19 20–24 25–29 30–39 40–49 50–59 60–64 65+

**Ethnicity:**  
(List all that apply)

---

**First Languages:**  
(List all that apply)

---

**Profession/Education:**  
(List all that apply)

---

Researcher: Emmanuel King Turner / [eturner@waikato.ac.nz](mailto:eturner@waikato.ac.nz) / +64 7 838 4627

Supervisor: <name> / <email> / <telephone>

## APPENDIX TWO: PAIRS OF TRIPLES USED TO FORM QUESTIONS IN THE FIRST USER STUDY (CHAPTER FOUR)

Question #	Subject (both triples)	Left Triple Predicate	Left Triple Object	Right Triple Predicate	Right Triple Object
0	Patea (town)	comment	Patea is the third-largest town in South Taranaki, New Zealand...	thumbnail	<a href="http://upload.wikimedia.org/wikipedia/commons/thumb/7/76/Patea,_Taranaki,_New_Zealand...">http://upload.wikimedia.org/wikipedia/commons/thumb/7/76/Patea,_Taranaki,_New_Zealand...</a>
1	Patea (town)	latitudeMinutes	45	name	Patea
2	Patea (town)	subject	Category:South Taranaki District	latitudeSeconds	26
3	Steven Spielberg (filmmaker)	birthYear	1946-01-01 00:00:00	birthDate	1946-12-18
4	Steven Spielberg (filmmaker)	alternativeNames	Steven Allan Spielberg, Stephen Spielberg	surname	Spielberg
5	Steven Spielberg (filmmaker)	shortDescription	Academy Award winning American film director and producer	occupation	Film director, producer, screenwriter
6	Steven Spielberg (filmmaker)	description	Academy Award-winning American film director and producer	shortDescription	Academy Award winning American film director and producer
7	Steven Spielberg (filmmaker)	name	Steven Spielberg	birthName	Steven Spielberg
8	Steven Spielberg (filmmaker)	name	Steven Spielberg	birthName	Steven Allan Spielberg

9	Steven Spielberg (filmmaker)	name	Steven Spielberg	surname	Spielberg
10	Athens (city)	country	Greece	aprLowC	10
11	The Beatles	activeYearsEndYear	1970-01-01 00:00:00	pastMembers	Pete Best
12	Steven Spielberg (filmmaker)	wikiPageExternalLink	<a href="http://www.empireonline.com/features/spielbergat60/60.asp">http://www.empireonline.com/features/spielbergat60/60.asp</a>	birthYear	1946-01-01 00:00:00
13	Patea (town)	subdivisionType	District	subdivisionType	Country
14	Patea (town)	longitude	174.4766693115234	longitudeDegrees	174
15	Barbie (doll)	name	Barbara Milli Roberts	surname	Roberts
16	Peter Jackson (filmmaker)	name	Sir Peter Jackson	name	Jackson, Peter
17	Peter Jackson (filmmaker)	name	Peter Jackson	name	Jackson, Peter
18	Steven Spielberg (filmmaker)	name	Steven Spielberg	name	Spielberg, Steven
19	Peter Jackson (filmmaker)	name	Peter Jackson	name	Sir Peter Jackson
20	Patea (town)	subject	Category:South Taranaki District	longitudeDegrees	174
21	Patea (town)	longitude	174.4766693115234	subdivisionName	Taranaki Region
22	Steven Spielberg (filmmaker)	networth	3.0E9	religion	Judaism

23	Patea (town)	Longitude East-West	E	longitudeSeconds	36
24	Patea (town)	coordinatesDisplay	inline,title	coordinatesRegion	NZ
25	Steven Spielberg (filmmaker)	label	Steven Spielberg	alternativeNames	Steven Allan Spielberg, Stephen Spielberg
26	Patea (town)	subdivisionType	District	subdivisionType	Region
27	Patea (town)	subdivisionName	Taranaki Region	subdivisionName	New Zealand
28	Patea (town)	subject	Category:Populated places in New Zealand	subject	Category:South Taranaki District
29	Patea (town)	subdivisionName	South Taranaki District	subdivisionName	Taranaki Region
30	Patea (town)	comment	Patea is the third-largest town in South Taranaki, New Zealand...	populationTotal	1143
31	Patea (town)	Longitude East-West	E	subdivisionName	Taranaki Region
32	Patea (town)	country	New Zealand	longitudeSeconds	36
33	Patea (town)	wikiPageExternalLink	<a href="http://www.stjospatea.school.nz">http://www.stjospatea.school.nz</a>	wikiPageExternalLink	<a href="http://72.14.253.104/search?q=cache:Ogv_p73ng-IJ:www.stdc.co.nz/pdf/patea...">http://72.14.253.104/search?q=cache:Ogv_p73ng-IJ:www.stdc.co.nz/pdf/patea...</a>
34	Athens (city)	areaCode	21	areaUrban	412
35	Patea (town)	subdivisionName	South Taranaki District	subdivisionType	Country
36	Peter Jackson (filmmaker)	name	Sir Peter Jackson	name	Peter Jackson
37	Athens (city)	capitalOf	Greece	capital	Greece

38	Oregon (state)	country	USA	incountry	USA
39	Barbie (doll)	name	Barbara Millicent Roberts	name	Barbara Milli Roberts
40	Patea (town)	wikiPageUsesTemplate	<a href="http://dbpedia.org/resource/Template:Infobox_settlement">http://dbpedia.org/resource/Template:Infobox_settlement</a>	longitudeSeconds	36
41	Patea (town)	country	New Zealand	wikiPageUsesTemplate	<a href="http://dbpedia.org/resource/Template:Infobox_settlement">http://dbpedia.org/resource/Template:Infobox_settlement</a>
42	Patea (town)	comment	Patea is the third-largest town in South Taranaki, New Zealand...	longitude	174.4766693115234
43	Athens (city)	aprPrecipitationMm	31	areaMunicipality	39
44	Steven Spielberg (filmmaker)	birthName	Steven Allan Spielberg	birthYear	1946-01-01 00:00:00
45	Patea (town)	country	New Zealand	geometry	POINT(174.477 -39.7572)
46	Patea (town)	label	Patea	name	Patea
47	Patea (town)	country	New Zealand	subdivisionName	New Zealand
48	Patea (town)	comment	Patea is the third-largest town in South Taranaki, New Zealand...	abstract	Patea is the third-largest town in South Taranaki, New Zealand...
49	Patea (town)	label	Patea	englishName	Patea