Pairwise Preferences Learning for Recommender Systems

Nunung Nurul Qomariyah

Doctor of Philosophy

University of York
Computer Science

January 2018

Dedication

To my beloved daughter, Azkia, and my beloved son, Arazka,

who always be the greatest reason for not giving up on my dreams.

Abstract

Preference learning (PL) plays an important role in machine learning research and practice. PL works with an ordinal dataset, used frequently in areas such as behavioural science, medical science, education, psychology and social science. The aim of PL is to predict the preference for a new set of items based on the training data.

In the application area of Recommender Systems (RSs), PL is used as an important element to produce good recommendations. Many ideas have been developed to build better recommendation techniques. One of the challenges in RSs is how to develop systems that are proactive and unobtrusive. To address this problem, we have studied the use of pairwise comparisons in preference elicitation as a very simple way of expressing preferences. Research in PL has also discovered this kind of representation and considers it to be learning from binary relations.

There are three contributions in this thesis:

The first and the most significant contribution is a new approach based on Inductive Logic Programming (ILP) in Description Logics (DL) representation to learn the relation of order. The second contribution is a strategy based on Active Learning (AL) to support the inference process and make choices more informative for learning purposes. A third contribution is a recommender system algorithm based on the ILP in DL approach, implemented in a real-world recommender system with a large used-car dataset.

The proposed approach has been evaluated by using both offline and online experiments. The offline experiments were performed using two publicly available preference datasets, while the online experiment was conducted using 24 participants to evaluate the system. In the offline experiments, the overall accuracy of our proposed approach outperformed the other 3 baseline algorithms, SVM, Decision Tree and Aleph. In the online experiment, the user study also showed some satisfactory results in which our proposed pairwise comparisons interface in a recommender system beat a common standard list interface.

Contents

List of Figures

List of Tables

Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisor Dr. Dimitar Kazakov for the continuous support, patience, motivation, and immense knowledge he has provided. Without his guidance, this thesis would not have been possible and I shall eternally be grateful to him for his assistance.

I would like to thank my thesis assessor, Dr. Tangming Yuan, for his insightful encouragement during my study and also to my thesis examiner, Prof. Hendrik Blockeel from Katholieke Universiteit Leuven Belgium, who has read this thesis and provided valuable feedback and support, which helped to improve the quality of the thesis. I would also like to thank all the Computer Science Department research administrators and IT staff who have been very helpful in all my administrative arrangements and always responsive to all my requests.

Financially, I was mainly supported by the Indonesia Endowment Fund for Education (LPDP), Indonesian Ministry of Finance. I was delighted with their support for making it possible for me to study here.

I also appreciate all the effort from the participants who were involved in my study, even though some of them did not get an incentive they still showed huge enthusiasm in supporting my study.

Furthermore, I would like to thank my family and friends for supporting me emotionally and spiritually throughout the writing of this thesis and for my life in general. In particular, I thank my husband, Anjar, who has always accompanied me throughout my wonderful PhD adventure.

Declaration

I declare that the research described in this thesis is original work, which I undertook at the University of York during 2014 - 2018. Except where stated, all of the work contained within this thesis represents the original contribution of the author.

Some parts of this thesis have been published in conference proceedings; where items were published jointly with collaborators, the author of this thesis is responsible for the material presented here. For each published item the primary author is the first listed author.

- Nunung N. Qomariyah and Dimitar Kazakov. Learning binary preference relations: analysis of logic-based versus statistical approaches. In *Proceedings of the ACM RecSys 2017 Joint Workshop on Interfaces and Human Decision Making for Recommender System (IntRS)*, vol. 1884, pages 30–34. CEUR-WS.org, Como, Italy, 2017.

- Nunung N. Qomariyah and Dimitar Kazakov. Learning from ordinal data with inductive logic programming in description logic. In *Proceedings of the Late Breaking Papers of the 27th International Conference on Inductive Logic Programming*, vol. 2085, pages 38–50. CEUR-WS.org, Orléans, France, 2017.

- Nunung N. Qomariyah and Dimitar Kazakov. Mixed-type multi-attribute pairwise comparisons learning. In *Proceedings of the 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, Cancun, Mexico, 2017.

Chapter 1

Introduction

1.1 Background and Motivation

Today, more and more people enjoy fast internet access which is used for numerous activities, such as browsing, shopping, video communication, playing games and so on. Businesses are also taking advantage of technological improvements. They sell products and services via the internet with various attractive offers and work hard to increase sales. One strategy to get more sales is personalisation. The role of personalisation in e-commerce has been predicted to increase. In 2018, 70% of e-commerce will move from business-to-consumer (B2C) and business-to-business (B2B) models to a model that focuses on the individual customer experience [43].

Following this trend, the initiative to implement *recommender systems* (RSs) is being popularized by large online marketplaces such as Amazon and eBay. The main goal of RSs is to recommend suitable items to the customer based on their preferences. RSs have been implemented in many domains, such as news, films, music, books, research articles and products in general. This field is becoming more popular with internet technology making processes faster and easier. Since the end of the 1990s when Amazon launched their Collaborative Filtering (CF) method, research in recommender systems has increased manyfold. Many ideas have been developed to build better recommendation techniques. In the recommender system handbook, Ricci et al. [106] discuss some of the current research challenges in this area, including developing a system to learn user preferences implicitly without bothering customers with a huge number of questions.

It is essential to elicit the buyers' preferences accurately and naturally, so the system can predict the best match items for them. Rating, feedback and product review are the most common methods used to express user preferences. Even though buyers have become used to giving ratings to items that they like, many people feel uncomfortable with the use of numerical values. A novel approach has come up with a scheme asking, "which one do you like better, item A or item B?" by using *pairwise comparisons* theory, proposed by Balakrishnan and Chopra [14]. Almost half of the participants in their study enjoyed answering pairwise questions compared to providing explicit ratings, while the majority of the participants liked the recommender system produced by the pairwise comparisons more than the one produced by the rating system. The finding is interesting due to the fact that they are not familiar with pairwise comparisons. It is expected that customers express their preferences more easily if the options are simplified. The pairwise comparisons method is very common in statistics, but incorporating it into recommender system still needs further investigation. Chapter 3 discusses studies of recommender systems which use pairwise comparisons. In order to make the best recommendation for the user, a very large number of pairwise choices may have to be asked. There is a trade-off between recommendation prediction accuracy and the number of questions asked before the customer gets their personal recommendation. An active learning method can choose the most informative pairs in order to capture user tastes. Thus, the number of questions can be reduced for user convenience.

Another relevant topic in recommender system research is the theory of web ontologies. There are numerous examples of research on ontologies which have been implemented in many different domains. In e-commerce systems, suppliers and vendors suffer from unstandardized ways of describing the products. Ontologies can address this problem by providing support for integrating heterogeneous and distributed information sources. They provide a more powerful technique to describe items than a conventional relational database system, especially in *semantic representation.* Allowing users to easily annotate related terms and build a graph of a whole understanding in context is an advantage, as it would be difficult to perform in a relational database system. Some ontology standards for product feature descriptions, vendor details and most e-commerce terms have been published on the web. Buyers can also exploit the use of ontologies in e-commerce to find items with a similar meaning in a given context.

RSs use *preference learning* as an important element to produce good recommendations. Preference learning (PL) is a subtopic in Machine Learning (ML) that works with an ordinal dataset, either in partial or full order. Nowadays, PL plays an important role in machine learning research and practice, because the ordinal data itself is used frequently in many areas, such as behavioural science, medicine, education, psychology and social science. For these domains, people can express their unique “value of preference”, which may differ from others. For example, some buyers may give a rating on a Likert scale to show whether they like a certain product or not; some paper submissions may be weakly accepted, rejected or accepted depending on the review results. The aim of PL is to predict preferences for a new set of items, so that the produced ranking is similar in some sense to the order provided in the given examples.

In logic-based machine learning, Inductive Logic Programming (ILP) is a robust method that can learn relations in First Order Logic (FOL). This method is suitable for use in learning user preference models from pairwise comparisons and can learn the order of preferences from a given set of examples. ILP works by finding a valid hypothesis that covers all positive examples and no negative examples (in some systems, the noise threshold in the data can be set) by using logic programs. The ILP method is most commonly associated with categorical data, but it can also handle numerical data and combinations of these two data types. An advantage of ILP, which can be beneficial for solving problems in preference learning, is its ability to learn from a limited number of instances as it bases its hypotheses on logic reasoning rather than pure statistics. ILP will usually generalise from the examples to produce a parsimonious rule using a minimum of constraints to explain the data. To the best of our knowledge, there is no work addressing the PL problem using ILP.

Muggleton introduced the term ILP in 1991 [90] and developed an application of it in FOL, called Progol [92]. However, the nature of problems like PL may be best described in other representations such as Description Logics (DL). Using description logics in learning about preferences can be beneficial for two reasons: (1) the inference result in description logics is easier to read and understand, and (2) we can also gain the advantage of description logics implementation, such as OWL and RDFS, which provide a mechanism to interconnect with other domains. This will also be useful to enrich the inference result. With the increasing number of knowledge bases available, using the ILP in DL representation will potentially encourage this research towards the semantic web field. This trend brings great future potential to research in DL. As stated above, the field of web ontologies has attracted the attention of e-commerce research due to the ease of interconnecting it with other entities via the web. On the other hand, the implementation of ILP in DL is still quite limited and challenging, some of the challenges are presented by Lehmann [70], Iannone et al. [62] and Fanizzi et al. [35].

A contribution to this field is made by implementing an ILP in DL to learn preference relations in strict order. An experiment with the existing ILP in DL system implementation is also presented in this thesis.

1.2 Research Questions

There are three research questions to be investigated in this thesis as follows:

1. Can ILP be applied to the data/models expressed in DL in order to learn relationship of order?

2. Is it possible to use Active Learning (AL) strategy in order to reduce the number of training examples, respectively to increase the accuracy?

3. Can ILP in DL be used in an application to learn a user's relative order of preferences and to produce a set of sensible recommendations?

1.3 Aims and Objectives

The aim of this research is to propose a method which combines the principles of pairwise comparisons with those of a logic-based ML approach and AL to learn e-commerce user preferences. The new approach developed in this research is implemented in a car recommender system and has been tested online by inviting an appropriate sample of participants to evaluate the system.

The objectives of this project are:

1. Develop a new ILP learner in DL capable of learning binary relations. This is called APARELL (Active PAirwise RELation Learner).

2. Apply APARELL to learn pairwise preferences represented as ordinal data.

3. Study the benefits of using Active Learning for the task of learning from pairwise comparisons.

4. Implement a recommender system with pairwise comparisons using APARELL on a large, real-world dataset.

5. Evaluate the above implementation of a recommender system with human participants.

1.4 Research Contributions

There are three major contributions to the field of machine learning and recommender systems:

1. ILP using DL representation (Chapter 4 and 5)
   In Chapter 4, an experiment using existing systems is provided highlighting the comparisons between statistical and logic-based machine learning. A further discussion and analysis are provided to explain why the logic-based approach can be beneficial in solving the PL problem. In Chapter 5, a novel approach in machine learning to learn strict order relations by using ILP in DL representation is proposed. The method is evaluated using two preference datasets and is compared to the other three baseline machine learning algorithms.
2. Active learning strategy (Chapter 6)
   The second contribution in this thesis is proposing an active learning strategy to support the inference process from the method mentioned in Chapter 5. A novel strategy is proposed to improve the learning accuracy of the preference learning dataset by using as few examples as possible.
3. Pairwise recommender system application (Chapter 7)
   The third major contribution of this thesis is proposing the recommender system method by using pairwise preference elicitation. This system implements APARELL in a real-world recommender system application with a large dataset from a popular used car website. The major contribution in the recommender system includes the introduction of a new method which combines pairwise preference elicitation with a logic-based approach in producing the recommendation and providing explanations for the choices made by the system. An online evaluation is performed by inviting a number of participants to evaluate the system. The application is ready to be used with structured linked open data for an e-commerce website.

1.5 Thesis Structure

The remainder of the thesis is organised as follows:

Chapter 2 Theoretical foundations: this chapter demonstrates the theoretical background of the thesis. It covers: (i) the problem in preference learning and recommender systems; (ii) knowledge representation, i.e. description logics and ontologies; and, (iii) the method, i.e. the basics of machine learning, inductive logic programming and active learning.

Chapter 3 Related work: in this chapter, recent work related to the thesis is described in more detail and the state of the art in four major research areas, ILP in DL, pairwise preference learning, active learning and recommender system, is also described.

Chapter 4 Learning binary preference relations: the tasks in learning binary relations are described in detail and the results of an experiment on existing systems to solve the tasks are presented. An analysis and discussion based on the experiment are also provided to explain the pros and cons of two ML approaches: statistical and logic-based. This chapter aims to understand the baseline ML approach for designing the proposed solutions in the next chapter.

Chapter 5 Inductive learning of ordinal data in description logics: this chapter describes in detail the new proposed method in ILP using DL representation. The accuracy of the performance of the proposed approach is evaluated against other ML approaches, i.e. Aleph, SVM and Decision Tree.

Chapter 6 Active learning to support the inference process: a new AL strategy to support the learning process using the method discussed in Chapter 5, is explained.

Chapter 7 Pairwise recommender system implementation: this chapter describes the recommender system application which is based on the novel approach in Chapter 5. An online experiment with participants is reported.

Chapter 8 Conclusions and future work: the work is summarised and some suggested future directions are discussed based on issues and points of interest arising in this research.

Chapter 2

Theoretical Foundations

2.1 Recommender Systems

2.1.1 Definition and history of recommender systems

RSs (RSs) is an emerging research field that has grown fast and become popular. The increase of interest in this research topic has also been driven by great improvements in internet technology and e-commerce. RSs have many advantages for e-commerce. Schafer et al. [114] define three ways in which RSs enhance an e-commerce system, by helping buyers with no experience in online shopping, by cross-selling the products and by improving customer loyalty. The peak explosion of research in RSs occured when Amazon launched their Collaborative Filtering (CF) method at the end of the 1990s, successfully increasing their sales [33]. The successful Amazon became popular and other online businesses started to implement RSs on their website. Amazon has patented their CF method as a United States Patent [76]. Due to the fact that the main goal of an RS is to find the preferred information and eliminate information which is not liked by a user, the RS field can be considered as a subset of information filtering [105]. The process of exploring a user's preferences from their historical data is followed by processing it using machine learning algorithms to build a ranked list of recommended items, as preferred by the user [106].

The idea of exploiting computers to recommend the best item for the user has been around since the beginning of computing. The first implementation of the RS concept appeared in 1979, in a system called Grundy [107], a computer-based librarian that provided suggestions to the user on what books to read. This followed in the early 1990s with the launch of Tapestry [44], the first commercial RS. Another RS implementation for helping people find their preferred articles was launched in the early 1990s by GroupLens, a research lab at the University of Minnesota, USA [105]. They named the system after the group, GroupLens Recommender System. This system claims to have a similar spirit to that of Tapestry, Ringo, BellCore and Jester. A further development of RSs in the late 1990s was the implementation of Amazon Collaborative Filtering [76], one of the most widely known RS technologies. Since this era, RSs based on Collaborative Filtering have become very popular and has been implemented by many e-commerce and online systems. Many toolboxes for RSs have also been developed. The success story of Amazon also gave rise to the development of many RS algorithms known as hybrid approaches, which combine multiple approaches.

Following the successful era at the end of the 1990s, industry offered generous funding to implement RSs research. The most popular competition in RSs was held by Netflix, a provider of internet streaming media. They launched the Netflix Prize[1] in 2006 and give 1 million US Dollars to the winner of the competition who provided the best RS movie recommendation. They announced the winning team in 2009. In 2010, YouTube also implemented an RS on their website [29].

2.1.2 Recent RS research and challenges

Recent RS research has become more specific, attracting scientists to specific recommender system conferences. The first ACM (Association for Computing Machinery) Conference on RSs (RecSys) was held in Minneapolis, Minnesota, USA. It successfully attracted 35 long paper submissions and 23 short paper submissions from 15 countries. Following that successful event, the ACM RecSyS Programme Committee decided to hold the event annually. Along with the event, the ACM RecSys Conference Proceedings, published since 2007, have become a well-cited source in RS research.

To date, more than 200 articles have been published in the RS field answering many challenges. RS research has not only become of interest in the computer science field, but also in a number of different fields like marketing, information technology, information science, economy and management. In 2012, Park et al. [100] classified 210 RS research papers into eight categories using data-mining approaches to process historical user data. They are, association rule, clustering, decision tree, k-nearest neighbour, link analysis,

[1]www.netflixprize.com

neural networks, regression, and other heuristic methods. They found that clustering and association rule were the most popular techniques used in business.

There are also studies in the RS field on incorporating emerged technologies such as web ontologies [88,115,131], multi agent systems [8,19,81,82,89,127] and PCs [4,14,34,63,109] to enhance the RS from different points of interest.

A number of sources state that there are a lot of issues in RS that still need to be improved. Some of the problems which will become important RS challenges [106] are discussed below:

- Scalability

  Internet technology has become increasingly better. Therefore, more people feel happy accessing the internet and interacting with online applications regularly. This has caused the real data of both items and users to grow more quickly. Therefore, there are opportunities to develop an approach to building RSs with greater capability of handling large scale datasets without interfering with system performance.

- Proactive and unobtrusive RSs

  Some users may find it hard to articulate their preferences and needs, but from the other side, they do not want to be bothered by questions or tasks when interacting with the system. The interesting challenge is how to make a recommender system able to learn a user's preferences implicitly. It has also become a challenging issue to develop a recommender system that can proactively recommend items which users may need at a specific time.

- User privacy

  Accessing users' personal data may cause users to feel insecure about their privacy. A recommender system which keeps users' private data safe and protects against malicious use will therefore be highly favoured.

- Diversity of the recommended items

  Recommender systems can be used as knowledge discovery tools. Users may want to explore the available options within the preferred items listed on the page. It will become an interesting challenge to define the diversity level and balance the diversity of recommendation results with accuracy.

- Generic user models and cross domain RSs

  It will be interesting to develop a recommender system which can build a user profile and use it to recommend diverse item categories in more than two domains.

- Distributed RSs

  Following the emergence of cloud computing technology, a recommender system will be more interesting if it is placed on open networks to maximise the robustness and flexibility of the system.

- Mobile RSs

  Developing the recommender system in the mobile platform will improve the user's interaction. The user will be easier to get local information as they move to another location, such as recommend the interesting places or trip destinations.

In addition to the challenges above, Felfernig et al. [36] also list challenges for further RS research:

- Focusing on the user perspective
- Sharing recommendation knowledge
- Context awareness
- Psychological aspects of RSs

Middleton [85] also mentions challenges in building navigation history and profile representations, which is an interesting topic in the Human Computer Interaction (HCI) field.

2.1.3 Recommender system techniques and applications

Not long after the RS concept was introduced, many researchers developed techniques to implement it in real online systems. Businesses were interested in implementing the concept to increase sales by recommending suitable products for their customers. Researchers in this field worked more to find the best method to learn user preferences and collect them as historical data. The more they knew about a user's preferences, the more accurate predictions of recommended items they could produce. Machine learning or data mining techniques were then used to explore users' historical data.

How techniques in RS are categorised differs from one source to another, and sometimes they use different names for the same concepts. The most common techniques are *Collaborative Filtering (CF)*, *Content-Based Filtering (CBF)* and *Hybrid* [5]. How the techniques work is described below:

- Collaborative Filtering (CF)

  This technique recommends items by looking at other users who have similar preferences.

- Content-Based Filtering (CBF)

  This technique recommends items by looking at how the user rates items.

- Hybrid

  This technique recommends items by combining two or more techniques.

Other techniques mentioned in the literature include community-based, demographic, knowledge-based [106], context-aware, rule filtering [42], stereotypes [107], item-centric/co-occurrence based [123], graph-based [60] and global relevance. Each e-commerce site will need different techniques or approaches, depending on their specific characteristics and goals. Each of these techniques has its own advantages and disadvantages.

The above-mentioned techniques have been implemented in real-world systems and have successfully helped people to interact and use online systems. Ghazanfar [42] and Middleton [85] have collected the literature on RS applications, as can be seen in Table 2.1.

2.1.4 Evaluation of recommender system

Recent research in RSs focuses on building the best algorithms to find the most suitable items for users. Success in any kind of computer systems, including RSs, should be measured using a suitable evaluation method. An achievement of overall goals can also be used as an important measure to evaluate how well a system performs [119].

Evaluating RSs can be difficult because each algorithm has its own particular focus. Some work better with a large dataset, whilst others work better with a smaller dataset. Furthermore, the different types of dataset used in some RS algorithms make them difficult to compare with others. Some work better in a specific domain and are not suitable for other domains [52]. Choosing an appropriate method to evaluate an RS will become an important issue. An appropriate method of evaluation ensures the system is confidently implemented in the market or in making a novelty system for academic purposes [119].

Table 2.1: Applications of RSs

Domain Category	Application Name	Commercial Site URL
Films	MovieLens	movielens.org
	MovieFinder	moviefinderonline.com
	Netflix	netflix.com
	Reel	reel.com
	Recommender Explorer	–
	Virtual Reviewers	–
	FilmTrust	trust.midswap.org/FilmTrust
Video	YouTube	youtube.com
Music and film	CDNOW	CDNOW.com
	Ringo	–
	LastFm	last.fm
	CoCoA	–
	Pandora	pandora.com
	MyStrands	mystrands.com
	iTunes	itunes.com
	Flickr	flickr.com
E-commerce	Amazon	amazon.com
	eBay	eBay.com
	Dietorecs	–
	EFOL	–
	Entrée	–
	FAIRWIS	–
	Ghani	–
	Levis	–
	LIBRA	–
	MIAU	–
	RIND	–
	Ski-europe	ski-europe.com
	Choicestream	choicestream.com
Restaurant	Entrée	–
Expertise finder	ReferralWeb	referralweb.net
	Linkedin	linkedin.com
News filtering	GroupLens	–
	PHOAKS	–
	P-Tango	–
	Google news	–
Email filtering	Tapestry	–
Web	Citeseer	citeseerx.ist.psu.edu
	Fab	–
	QuickStep	–
	Foxtrot	–
Books	Which Book	whichbook.net
	What Should I Read Next	whatshouldireadnext.com
	Library Thing	librarything.com
	Libra	–
Holidays and travel	Trip Advisor	tripadvisor.co.uk
Electronic program guides	Electronic program guides	–
Other	Campiello	–
	ELFI	–
	OWL	–

The evaluation of an RS can be conducted using several methods and can be an offline test or online test. The test can be combined with a user study to measure how satisfied the user is with the system. An offline test will not cost too much because it does not need user involvement. An offline test is usually performed to make sure that all algorithms and environments work well before they are given an online test. According to [33], the offline test only looks at users' historical data. An online test can be more useful, because it can discover the real taste of the users.

Some important properties to be considered in deciding the best recommender algorithms include ([119] and [9]):

1. Prediction accuracy and coverage

   This is the most discussed property in many sources. This property follows the basic assumption in RSs that users prefer more accurate prediction results which cover more items.

2. Cold-start

   In many cases, when a new item or a new user is added to the system, the recommender algorithm can have difficulties in making a recommendation because it lacks sufficient information. This problem is called *cold-start*.

3. Novelty and serendipity

   Novelty is used to measure how the RS shows items that are new or unknown to the user. Serendipity is used to measure how the recommender system provides surprising yet beneficial items to the users.

4. Diversity

   Some users of a certain type of application may like diverse recommendations rather than items which are too similar.

5. Utility

   A utility score is calculated from values that the system gives to the users.

6. Trust, risk and privacy

   This property refers to user risk when accepting the recommendation. For example, in stock purchasing recommendations, users may deal with a higher risk compared to movie recommendations. The user also needs to feel secure in using the system. This includes privacy and trust.

7. Robustness, adaptivity and scalability

   The general quality measurements of the system include these three properties. Robustness is the ability of the system to protect itself from an undesired attack that could cause the system to produce false recommendations, while adaptivity means how fast the algorithm can be adjusted when there are changes in the user's preferences. Scalability can also be considered because real data may go faster than the developer predicted.

8. Usability

   Users can judge how easy the recommender system is to use. The easiest way to evaluate is with user studies (e.g. survey, observation and monitoring). Pu and Chen [102] propose a framework to evaluate RSs from the user's perspective, called ResQue (the RS's Quality of user experience). The framework is used in this thesis to evaluate our RS. It consists of 13 constructs and a total of 60 questions. The framework is built to assess the perceived qualities of recommenders such as their usability, usefulness, interface and interaction qualities, users' satisfaction with the system. It also looks at the influence of these qualities on users' behavioural intentions, including their intention to purchase the products recommended to them, return to the system in the future and tell their friend about the system.

2.2 Pairwise Comparison

The term Pairwise Comparison (PC) originally comes from *psychometrics*, the field that concerning psychological measurements. Thurstone introduced this theory in 1927 [129]. Bradley-Terry-Luce [23] then proposed a model for scaling paired comparisons in 1952. The basic concept of PCs generally refers to the process of making a decision by comparing items in pairs or choosing which item has a greater quantitative property than another. PC is popular in a number of application areas, such as marketing, voting, multi-agent systems and ranking players.

Thurstone's law of comparative judgement [129] is defined as:

$$S_1 - S_2 = x_{12} \times \sqrt{\sigma_1^2 + \sigma_2^2 - 2r\sigma_1\sigma_2} \tag{2.1}$$

in which:

S_1, S_2 = the psychological scale values of the two compared stimuli.

x_{12} = the sigma value corresponding to the proportion of judgements $p_{1\succ 2}$.

When $p_{1\succ 2}$ is greater than 0.5 the numerical value of x_{12} is positive.

When $p_{1\succ 2}$ is less than 0.5 the numerical value of x_{12} is negative.

σ_1 = discriminal dispersion of stimulus R_1

σ_2 = discriminal dispersion of stimulus R_2

r = correlation between the discriminal deviations of R_1 and R_2 in the same judgement.

This formula determines whether there is a different reaction when the subject of an experiment is given two or more stimuli. The original formula for pairwise comparison is adjusted in some applications, following different main goals and application purposes.

The Bradley-Terry model (BTM) [23] is also commonly used as a competitive relationship probability model, the contestant's win rate is proportional to his or her competitiveness. As an example of how BTM is used, we can consider only sports whose rules do not allow for ties. Suppose there are 30 basketball teams in the NBA, each playing 82 games in a regular season (so there are 1,230 total games). The simplest strategy to predict the overall team ranking is by comparing the number of games won by each team. An observation is made of each game played by two teams (i, j) and whether team i or team j wins. Suppose team i beats j $x(i, j)$ times and loses to team j $x(j, i)$ times. Each team i has some 'strength' β_i which is represented in the form of a real number. Within the basic probability model we can calculate the Maximum Likelihood Estimations (MLEs) of strengths β_i, β_j,... which imply a ranking order.

The probability model of team i beating j is defined by: $P(i \succ j) = \beta_i - \beta_j$. BTM treats this outcome as an independent Bernoulli random variable with Bernoulli distribution (p_{ij}), where the log-odds corresponding to the probability p_{ij} that team i beats team j is modelled as:

$$log \frac{p_{ij}}{1 - p_{ij}} = \beta_i - \beta_j \tag{2.2}$$

Equivalently, solving for p_{ij} yields

$$p_{ij} = \frac{e^{\beta_i - \beta_j}}{1 + e^{\beta_i - \beta_j}} = \frac{e^{\beta_i}}{e^{\beta_i} + e^{\beta_i}} \tag{2.3}$$

Although there are some extensions of BTM to accommodate a real situation, the basic probability model in BTM assumes the conditions mentioned below hold:

1. each game has a definite winner (no ties)

2. no home field advantage

3. not considering more elaborate modelling of point difference

4. strengths do not change with time.

If the competition level (strength) of i is higher than j, then the probability that team i wins against j is also high. Maximum likelihood is then calculated to estimate the individual competition level and then use this for ranking all individuals. This ranking can be used to predict future win probabilities. MLE for this BTM can be performed using standard software for generalised linear models or using specialised programs like R [128] package BradleyTerry2 [132]. This competition perspective can also be used for mining user preferences in RS by considering item features as competition levels.

2.3 Description Logics

Description Logics (DLs) are a family of languages that can be used to represent knowledge in a structured, formal, yet understandable way. The name *description logics* was taken from the fact that they provide a formal way to represent the important notions of an application domain as concept *descriptions*. The term *logics* means that they are equipped with a formal, *logic*-based semantics [12]. DLs are fragments of First Order Predicate Logic (FOL) that have less expressive power. Many DLs are more decidable for inference problems than FOL. Another feature that makes DLs more successful is a more readable variable-free representation. However, the main reason for using DLs rather than predicate logic is that DLs are carefully tailored, such that they combine interesting means of expressiveness with the decidability of important reasoning problems. In this section, a brief explanation of basic DLs and their equivalent syntax in FOL is provided.

2.3.1 Representing knowledge in DLs

DLs represent the world in terms of *concepts*, *objects* and *roles*. Concepts can be seen as a formal definition of classes in an application domain, e.g. one can define a father as a man having a child. They have two functions: (1) to describe a collection of objects and (2) to describe the properties that a class should hold. Objects are individuals that belong to one or more concepts. Roles represent a binary relationship between objects, e.g. John is Anna's father. In FOL, objects correspond to "individual constants", concepts correspond to “unary predicates” and roles correspond to “binary predicates”. All this information is stored in the form of a *knowledge base* which comprises two components, an assertional part (*ABox*) and a terminological part (*TBox*). In more expressive DLs, the latter is further subdivided into *TBox* and *RBox*, which contain knowledge about roles.

The notation used to represent knowledge does not include variables and is inspired by set theory. Therefore, the following Boolean constructors are used, conjunction ($\sqcap$), which is interpreted as set intersection, disjunction ($\sqcup$), which is interpreted as set union, and negation ($\neg$), which is interpreted as set complement. Some quantifier constructors are also used, the existential restriction ($\exists R.C$) and the universal restriction ($\forall R.C$), as well as the number restriction constructor ($\geq nR$). DL knowledge base representation is discussed further in the following sections.

Terminological knowledge. DLs provide the declaration of universal statements of a domain in the *Terminology Box (TBox).* The TBox is a finite set of General Concepts Inclusions (GCI) and role inclusions. It corresponds to the schema in a relational database. The GCI is of the form $C \sqsubseteq D$, where C and D are (complex) *concepts* and a role inclusions are of the form $R \sqsubseteq S$, where R and S are *roles*. It is also allowed to use the concept equivalence ($C \equiv D$) as an abbreviation of two GCIs: $C \sqsubseteq D$ and $D \sqsubseteq C$, as well as the role equivalence ($R \equiv S$) as an abbreviation for: $R \sqsubseteq S$ and $S \sqsubseteq R$. As an example, a woman can be defined as a female person by writing this declaration:

$$\text{Woman} \equiv \text{Person} \sqcap \text{Female}$$

In FOL, it is expressed: $\forall x(\text{Woman}(x) \rightarrow \text{Person}(x) \wedge \text{Female}(x))$.

In a more expressive terminological formalism, a constraint such as ‘only humans can have human children’ is allowed and written:

$$\exists \text{hasChild.Human} \sqsubseteq \text{Human}$$

In FOL, it is expressed as: $\forall x \exists y$ (hasChild$(x, y) \wedge$ Human$(y) \rightarrow$ Human (x)).

Assertional knowledge. The assertional set in the knowledge base is called the ABox. It is used to state the properties of individuals. In a relational database it is called ‘data’, while in the FOL it is known as ‘ground terms/facts’. For example, ‘Anna is a woman’ is declared as:

$$\text{Woman(anna)},$$

'John has a child which is called Anna' or 'Anna is a child of John' is declared as:

$$\text{hasChild(john,anna)}.$$

Modelling relationships between roles. Knowledge of roles can be expressed in RBox axioms, which refers to properties of roles. For example, the role parentOf is a sub-role of ancestorOf is declared as:

$$\text{parentOf} \sqsubseteq \text{ancestorOf}$$

In FOL, the above statement is written as: $\forall x \forall y(\text{parentOf}(x, y) \rightarrow \text{ancestorOf}(x, y))$, which states that being a parent of somebody implies being an ancestor of them.

Complex role inclusion axioms can include role composition, which is used to explain a certain role as a composition of two or more roles. Note that role compositions can only appear on the left-hand side of complex role inclusions. For example, a composition of role brotherOf and parentOf can be used to describe role uncleOf, as below:

$$\text{brotherOf} \circ \text{parentOf} \sqsubseteq \text{uncleOf}$$

In FOL, it can be expressed as: $\forall x \forall y \forall z(\text{brotherOf}(x, y) \wedge \text{parentOf}(y, z) \rightarrow \text{uncleOf}(x, z))$, states that the brother of someone's parent is his/her uncle.

In general, DLs offer more convenient constructs than the corresponding FOL but do not extend its expressivity. A general translation between DLs and FOL terms is provided in Table 2.2.

Table 2.2: Translation between DLs and FOL terms

DLs	FOL
concept	unary relationship/predicate
role	binary relationship/predicate
assertions or facts	predicate with no variable (ground facts)
inclusion/subsumption	implication
equivalence	bi-implication

2.3.2 DL reasoning

As mentioned above, knowledge representation is performed in such way that a machine can understand and automatically reason with the given knowledge. One of the advantages of logic-based knowledge representation, including DLs, is that once a body of knowledge has been transferred into logical representation, queries can be performed in an intelligent way which goes well beyond traditional databases. Some of the typical tasks for DL knowledge-bases which require inferencing can include KB satisfiability, axiom entailment, concept satisfiability, instance retrieval and classification.

There are several other reasoning tasks beyond those already mentioned which can also be performed in DL KBs, such as abduction, induction, explanation and module extraction. Many different techniques for DL reasoning are well studied, such as Tableau, Automata, Consequence-based reasoning and Resolution. Most of them originate from well-known approaches for theorem proving in a FOL setting. The aims of DL reasoning are soundness and completeness of decision procedures to guarantee termination. More details on DL reasoning can be found in [112]. Reasoning in DL follows the Open World Assumption (OWA) which makes it different from reasoning in other logics. Unlike working under the Closed World Assumption (CWA), with the OWA it is assumed that the knowledge base is incomplete; therefore, any missing information is treated as *unknown* rather than just *false*.

2.3.3 DL languages family

There is always a trade-off between expressivity and complexity of reasoning when choosing a language. The expressiveness of a description logic is determined by the operators allowed in the language. Higher expressiveness implies higher complexity. There is a well-established naming convention for DLs which depends on the operators used. DL languages are named by using a label starting with one of the following basic logics:

- $\mathcal{ALC}$ stands for Attributive Language with Complement [116]. This is a base language which allows:
  - Atomic negation (negation of concept names that do not appear on the left-hand side of axioms)
  - Concept intersection
  - Universal restrictions

  - Limited existential quantification

- $\mathcal{FL}$ stands for Frame-based description Language [72]; it allows:
  - Concept intersection
  - Universal restrictions
  - Limited existential quantification
  - Role restriction
- $\mathcal{EL}$ stands for Existential Language; it allows:
  - Concept intersection
  - Existential restrictions (of full existential quantification)

The naming scheme for DL languages often follows the order below:
$((\mathcal{ALC}|\mathcal{S}|\mathcal{FL}|\mathcal{EL})[\mathcal{H}]|\mathcal{SR})[\mathcal{O}][\mathcal{I}][\mathcal{F}|\mathcal{N}|\mathcal{Q}]$
The meaning of the DL naming scheme letters are as follows:

- $\mathcal{S}$ denotes $\mathcal{ALC}$ with transitivity statements.
- $\mathcal{H}$ denotes role hierarchies which allow for simple inclusions.
- $\mathcal{R}$ denotes complex role inclusions. In above naming scheme, $\mathcal{SR}$ means that it is an $\mathcal{ALC}$ which has been extended with all kinds of RBox axioms, as well as self-concepts[2]. It subsumes all of $\mathcal{ALC}, \mathcal{ALCH}, \mathcal{S}$, and $\mathcal{SH}$.
- $\mathcal{O}$ denotes nominal concepts.
- $\mathcal{I}$ denotes inverse roles.
- $\mathcal{F}$ denotes functional roles which can be expressed as $\top \sqsubseteq 1.\top$. It becomes obsolete once $\mathcal{N}$ is present and both are superseded by $\mathcal{Q}$.
- $\mathcal{N}$ denotes number restrictions, e.g. to express a concept of 'a mother of at least 3 children' (Woman $\sqcap \geq 3$ hasChild).
- $\mathcal{Q}$ denotes qualified number restrictions, e.g. to express the concept of 'a mother of at least 3 male children' (Woman $\sqcap \geq 3$ hasChild.Male).

[2]The self-concept enables inclusion of "role loops", i.e. situations where an individual is simultaneously source and target of the same relation

As one family of DL languages, $\mathcal{ALC}$ [116] has the least expressivity in basic DL language. It is described here as an example of how different features can be added to a family of DLs and affect the complexity. We refer the reader to a description logic navigator[3] for more detailed information about the complexity of a particular DL language. $\mathcal{ALC}$ allows us to construct complex concepts from simpler ones using various language constructs. The capabilities include direct or indirect expression, e.g. concept disjointness, domain and range of roles and the empty role.

$\mathcal{ALC}$ supports all Boolean operators on concepts $(\sqcap, \sqcup, \neg)$ as well as universal and existential role restrictions. Top concept $\top$ and bottom concept $\perp$ can be expressed indirectly but are typically included explicitly. In this DL, RBox axioms are not allowed, neither are role inverses, cardinality constraints, nominal concepts and self-concepts. $\mathcal{ALC}$ is a proper fragment of OWL [54] and is generally considered to be a prototypical description logic for research investigations.

Some $\mathcal{ALC}$ extensions support the transitive roles, i.e. $\mathcal{CIQ}, \mathcal{TSL}, \mathcal{ALC}_+, \mathcal{ALC}_{R^+}$ and $\mathcal{ALC}_\oplus$. Of these, $\mathcal{CIQ}, \mathcal{TSL} and \mathcal{ALC}_+$ all support role expressions with transitive or transitive reflexive operators. Extensions of $\mathcal{ALC}$ described in [56] are shown in Figure 2.1. The extensions of $\mathcal{ALC}$ with role transitivity are:

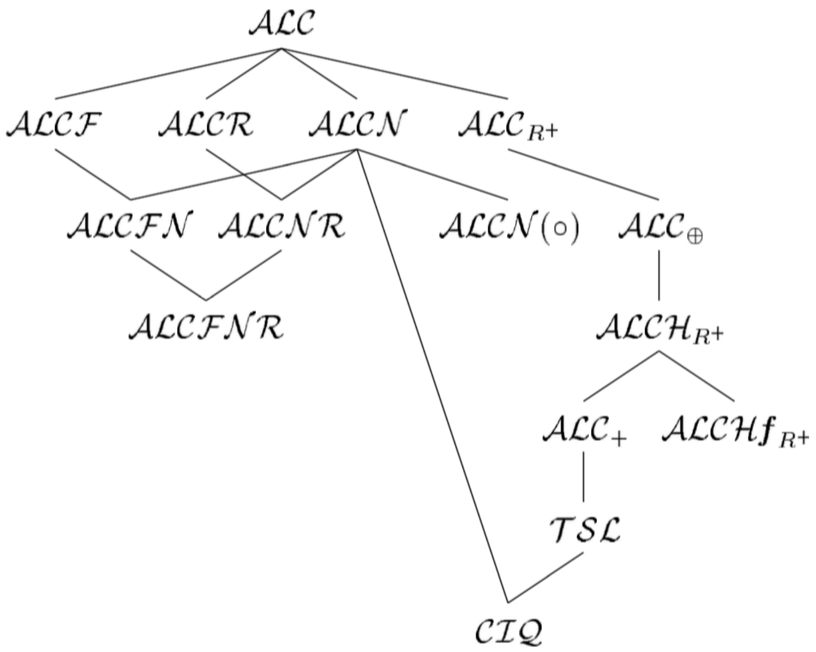


Figure 2.1: $\mathcal{ALC}$ extensions

[3]http://www.cs.man.ac.uk/ ezolin/dl/

- $\mathcal{ALC}_{\mathcal{R}^+}$ [113] (in DL naming schemes often abbreviated to $\mathcal{S}$) — $\mathcal{ALC}$ augmented with only transitively closed primitive roles (axioms of the form $R \in \mathbf{R}_+$) and no *primitive role introduction* ($R \sqsubseteq S$) is allowed. The complexity of deciding the satisfiability of $\mathcal{ALC}_{\mathcal{R}^+}$ concept expression is PSPACE-complete, the same as for $\mathcal{ALC}$;

- $\mathcal{ALC}_{\oplus}$ [113] —$\mathcal{ALC}_{\mathcal{R}^+}$ augmented with a restricted form of primitive role introduction axioms by associating each non-transitive role R with its transitive orbit $R^{\oplus}$. For example, it can assert an inclusion relation: son $\sqsubseteq$ son$^{\oplus}$, daughter $\sqsubseteq$ daughter$^{\oplus}$ and descendant $\sqsubseteq$ descendant$^{\oplus}$, where $\{\text{son}^{\oplus}, \text{daughter}^{\oplus}, \text{descendant}^{\oplus}\} \subseteq \mathbf{R}_+$. The complexity of the concept satisfiability problem is EXPTIME-complete, the same as for $\mathcal{ALC}_+$;

- $\mathcal{ALC}_+$ [10] — $\mathcal{ALC}$ augmented with union ($\sqcup$), composition ($\circ$) and transitive closure role (R^+) expressions. Its concept satisfiability problem is known to be EXPTIME-complete.

As described in [57], the transitive orbit of a role R, denoted $R^{\oplus}$, is a transitive role which subsumes R and can be defined by the axioms $R^{\oplus} \in \mathbf{R}_+$ and $R \sqsubseteq R^{\oplus}$. The interpretation of $R^{\oplus}$ is therefore a superset of the interpretation of the transitive closure of R, but not necessarily the smallest one: i.e. only $(R^{\oplus})^{\mathcal{I}} \supseteq (R^+)^{\mathcal{I}}$ is granted. In other words, it is simpler than transitive closure [56, p.59].

The relation between $\mathcal{ALC}$, $\mathcal{ALC}_{R^+}$, $\mathcal{ALC}_{\oplus}$ and $\mathcal{ALC}_+$ is shown in Figure 2.2.

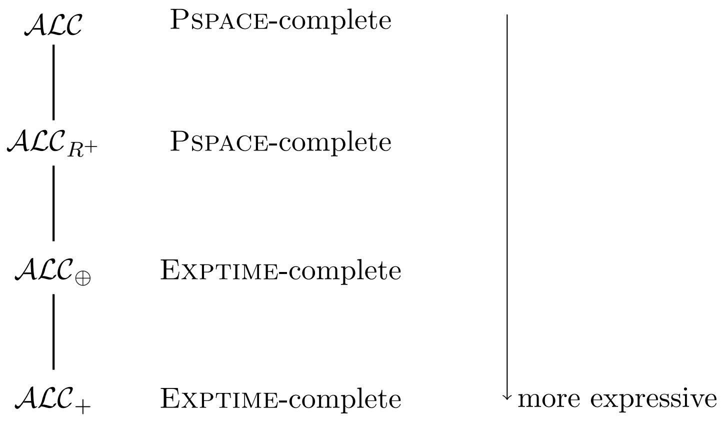


Figure 2.2: Some extensions of $\mathcal{ALC}$ with transitive relation

2.4 Ontologies

The term *ontology* has existed from the beginning of philosophy. Since the Aristotelian era, a study of classifying things in the world, included describing existence, has been a major concern. Artificial Intelligence (AI) has adopted the term ontology to describe a real world problem in a machine-readable specification. Studer et al. [126] suggest the best definition of ontology in AI as *"a formal, explicit specification of a shared conceptualisation"*. Ontologies originate from the branch of philosophy meaning "a systematic form of existence." Gruber [45] states the definition of ontology as "specification of conceptualisation in a formal and explicit way that can be shared with others." In a knowledge-based system, the term "exists" refers to any concept that can be represented.

In computer science, the concept of ontologies is used to organise information and deal with the representation of entities, ideas, and events along with their properties and relations based on their categories [124]. Since the early 1990s ontologies have become widely researched. The most interesting part of ontologies is the understanding of knowledge that can be shared and communicated to humans, agents or application systems. Ontologies have become an important asset in describing the structure and semantics of information exchange [37].

The concept of ontologies is similar to database schema, although ontologies have different characteristics:

- Ontologies use a language with richer syntax and semantics.
- They use a semi-structural natural language to describe information rather than a tabular model.
- Terminologies are shared and consensual.
- They provide a domain theory rather that the structure of data containers.

2.4.1 Ontology representations

Ontologies require a formal logical language to express the terminology in a certain domain. DL as a language with well-defined semantics and powerful reasoning tools has been chosen as the best representation of ontologies. Recent web standards such as XML and RDF can be used as a standard syntax to express ontologies. XML (eXtendible Markup Language) is a tag-based language for building tree structures using linear syntax. Each leaf of the tree has a well-defined tag and context so that the information can be understood. XML uses seven key terms for presenting information, elements, attributes, references, comments, processing instructions, CDATA and Prolog.

RDF (Resource Description Framework). RDF is an application of XML with additional meta-information to the semantic web. RDF is expressed using three data models, known as triples, called subjects, predicates and objects. The *subject* of an RDF statement is either a Uniform Resource Identifier (URI) or a blank node, which denotes resources. Blank nodes are indicated anonymous resources. The *predicate* is a relationship between resources and/or any atomic values provided by data type in XML. The *object* is a property value that is mapped by the predicate. An object can be a URI, blank node or a Unicode string literal. Sets of triples are called RDF graphs. RDF Schema (RDFS) extends RDF with vocabulary for schema modelling.

OWL (Web Ontology Language). More specifically, ontologies can be built using ontology languages. Some well-known languages used in the community are CyCL, KIF, Ontolingua, Frame Logic, CLASSIC, XOL, OIL, DAML+OIL and OWL. OWL extends RDFS into a very expressive ontology language. DL syntax can be used in representing OWL ontologies, as DL is the basis for widely used ontology languages. A *concept* in DL is referred as a *class* in OWL and a *role* in DL is referred as a *property* in OWL. Horrocks et al. [55] describe the basic differences between OWL and RDF; OWL uses the ability of RDF to express basic facts, and uses the capability of RDF schema to create statements about the class-and property-structures and extends them in some ways, as explained below:

- Classes

  In OWL, classes can be declared and organised into a subsumption ("subclass") hierarchy. RDF Schema can declare the classes in the same way as OWL. Additionally, as an extension of RDFS, OWL classes can be specified in some logical operators (intersections, unions or complements) with other classes, or as enumerations of specified objects. OWL can also define whether a class is disjoint with other classes and whether an individual is distinct to other individuals.

- Property

  OWL and RDFS share the capability of declaring properties, organising these properties into a "subproperty" hierarchy and providing domains and ranges for these properties. Here OWL is extending the RDFS with the capability of specifying that the domains of OWL properties are OWL classes, and ranges can be either OWL classes or externally-defined datatypes, such as string or integer. OWL can state that a property is transitive, symmetric, functional or is the inverse of another property.

- Restrictions

  The major extension of OWL over RDFS is the ability to provide restrictions on how properties behave (locally) on a class. OWL can define classes where a particular property is restricted so that all the values of that property must belong to a certain class (or datatype); at least one value must come from a certain class (or datatype); there must be at least certain specific values; and, there must be at least or at most a certain number of distinct values.

2.4.2 Ontologies for e-commerce

Standard for product ontologies definition. E-commerce needs a standard so that users and businesses can communicate with each other using the same understanding. The emerging technology of ontologies has facilitated this need. There are efforts from a number of researchers to build standards to fulfil the need of a single understanding of ontology classes. Initiatives such as eOWL, GoodRelation and schema.org are described in this section.

In addition to creating standards, to build good communication between participants in e-commerce there are also efforts to interlink data available on the web called Linked Open Data (LOD). From the interlinking lines, it can be seen that DBpedia is one of the most complete sources for ontology definitions. DBpedia extracts structured information from Wikipedia and makes this information available on the Web. Some open web ontologies can also be used to find the classification of commercial products:

- **Product Ontology**[4]

  This service provides approximately 300,000 precise definitions of products or services that extend schema.org and GoodRelation for e-commerce markup. Product Ontology builds the class definition based on English Wikipedia entries. Any new

[4]www.productontology.org

entry in the English Wikipedia will shortly be available in Product Ontology. A class definition can be retrieved using this URI:

```
http://www.productontology.org/id/Racing_bicycle
```

Another major innovation in utilising ontologies to represent commercial products has been developed by Hepp, head of the E-Business and Web Science Research Group at the Universität der Bundeswehr Munich. They developed GoodRelations[5] [50], a lightweight ontology for annotating products or services, offerings and other aspects of e-commerce, so that users can search suitable suppliers using ontologies. GoodRelations has also collaborated with Google and Yahoo. The details of e-commerce scenarios such as eligible countries, payment and delivery options, quantity discounts, opening hours, terms and conditions can be expressed easily. Prior to the initiatives to develop GoodRelations, Hepp built eClassOWL[6] [49] in 2003 to describe the types and properties of products and services on the semantic web.

Another initiative which comes from community collaboration between Bing, Google, Yahoo and Yandex is called Schema.org [46], which was launched on 2 June 2011. It provides a shared vocabulary and focuses on defining the item types and properties that are most valuable to search engines. It builds a schema for many categories including films, music, organisations, TV shows, products, places and many more. Since 2012, e-commerce schema from schema.org has been officially integrated with GoodRelations. An example to retrieve the information can be seen using this link:

```
http://schema.org/Person
```

- **Used Car Ontology**[7]

  The Used Cars Ontology (UCO) was created by Hepp Research GmbH and Makolab S.A. This ontology describes the details of used cars and their history, such as ownership records, any damage, modifications, features, MOT testing, accident information, replacement of core parts, parking type, whether the owner smoked, etc. UCO

[5]www.purl.org/goodrelations/
[6]www.heppnetz.de/projects/eclassowl/
[7]http://ontologies.makolab.com/uco/ns.html

is an extension of Good Relations. Two other related ontologies for the automotive industry are designed to be used in combination with UCO: Vehicle Sales Ontology[8] and Volkswagen Vehicle Ontology[9]. VSO describes many automotive types, such as cars, boats, bikes and other vehicles. While VVO describes Volkswagen-specific features of vehicles such as paint, parking, roofs, seats, services, steering wheels and traffic patterns.

Interlinked data. The promise of shared knowledge in ontologies has now become a reality. A large number of datasets have been published and are freely accessible to anyone. The World Wide Web has shifted from a web with hyperlinked documents into a web of shared and linked data. The new initiative to interlink datasets was initialised by the Linking Open Data community in January 2007 and supported by W3C (World Wide Web Consortium) Semantic Web Education and Outreach Group. About 50 billion facts from subjects like biology, chemistry, economics, energy, geography, media and others are available free (e.g. under Creative Commons license) on the Linked Open Data (LOD) cloud. This data is published in RDF format and most are allowed to be reused for commercial purposes. Linked Open Data has the main focus on high-quality metadata management [15]. The interconnected datasets in LOD [3] are presented in Figure 2.3.

LOD visualisation contains links to the available datasets. Richard Cyganiak, from DERI, and Anja Jentzsch, from Freie Universität Berlin, work to compile the data regularly and provide it with a linked diagram that can be accessed freely [3]. At the last update in February 2017, the diagram contained 1,163 datasets. It has grown very quickly since the beginning of May 2007 when it contained only 12 datasets. The rapid growth of the LOD in the last 10 years shows a great opportunity for the further work of this thesis in terms of the ontology integration.

2.5 Machine Learning

Machine learning deals with the task of learning from data. It plays a central role in many areas of computer technology, due to the increasing need to build intelligent computer systems. In today's busy world, a system that can help solve problems automatically is important, e.g. people need a vehicle which can learn to drive on the road, people need to

[8]http://purl.org/vso/ns
[9]http://purl.org/vvo/ns

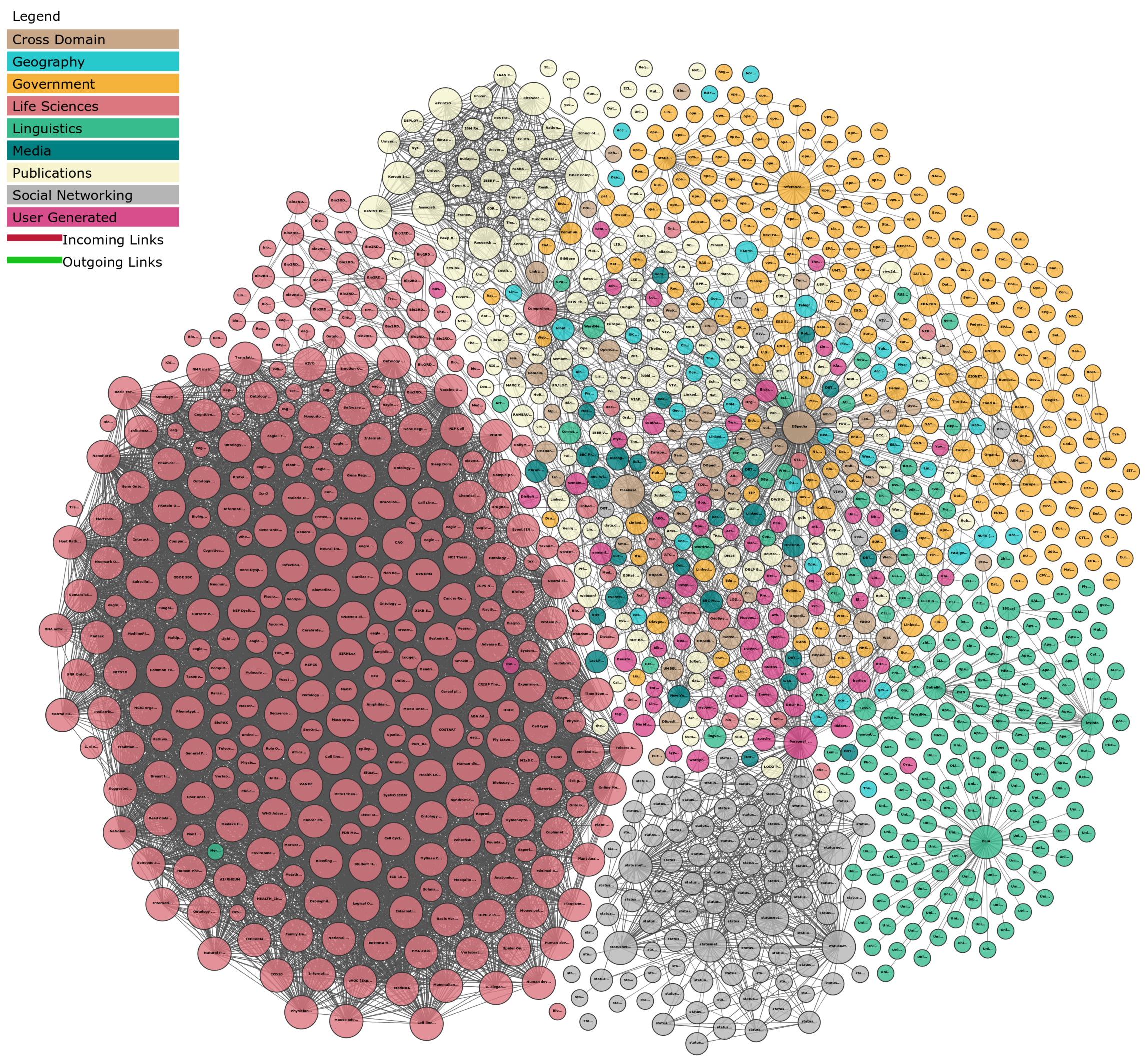


Figure 2.3: Linking Open Data cloud diagram in 2017

protect themselves from fraudulent credit card transactions and so on. Briefly, this field is concerned with constructing a computer program that can automatically improve with experience [87].

In line with the definition given by Mitchell [87], there is a common understanding of machine learning, according to Flach [38]:

> Machine learning is all about using the right features to build the right models that achieve the right tasks.

It can be said that machine learning has three basic ingredients:

1. Well defined tasks

   Task is a description of the problem that has to be solved in a certain domain. A common way to simplify tasks is by classifying problems into classes, called a *classification* problem. Another common way is labelling the training data with urgency scores (e.g. 0 to 10), called a *regression* problem. Alternative ways to learn the training data without any prior information about it, are called *clustering* problems. According to Mitchell three features are needed to define the learning problem: the class of the tasks, the measure of performance to be improved and the source of experience.

2. Right models

   A model is learned from the given training data in order to solve a certain task. This is the central concept of machine learning. There are three common groups of models, *geometric models, probabilistic models* and *logical models.* A geometric model is constructed using geometric concepts, such as lines, planes, and distances. It is relatively easy to visualise the data using a geometric classifier keeping two or three dimensions. A probabilistic model learns from the training data by looking at the probability of how the next incoming data will be grouped. This is a process of reducing uncertainty. The logical model is constructed by translating the problem into rules (if-then-else function) that are understandable by humans. Some rules are easily organised into a tree structure.

3. Best features

   Feature is a measurement that can be applied to any instance in the domain problem. Determining the features to be used depends on the models that have been chosen, because the models are defined in terms of features. Some features used in machine learning are of type integer (e.g. number of occurrences), Boolean (e.g. identifying whether a sentence contains a certain word) and finite sets (e.g. set of colours and shapes). Deciding any pair of features and models is often carried out as an iterative process in Machine Learning. Sometimes the right features are captured after constructing the models. If the model does not perform satisfactorily then it is necessary to analyse the performances and understand which part of the features needs to be improved [38].

The stages of applying machine learning for solving real world problems are described in [83]:

1. Formulating the problem
2. Determining the representation
3. Collecting the training data
4. Evaluating the learned knowledge
5. Fielding the knowledge base

The first step in applying machine learning is formulating the problem. In other words, defining the task, as explained above. Several techniques can be used to make a real-world problem much simpler. At this stage, it is important to define the goal clearly. Following that, the next stage in applying machine learning techniques is determining the representation of the training data and knowledge to be learned. This stage refers to choosing features to describe examples and characterise learning result. The third stage is collecting the training data for the induction process. In most application domains, the training data can be collected with help from experts to classify the data or to generate them. After collecting the training data, the learner in machine learning will induce the rules from them. This process is called knowledge acquisition. A standard approach to evaluate the learned knowledge is dividing the data into two sets, training and test. The process can be repeated with different splits until the desired rules performance is reached. The most important part of the evaluation process is that there is an involvement of experts to examine the learned knowledge. Fielding the knowledge base in the broadest possible sense to solve the problem is the final stage. The learned knowledge can be used even without a computer system to help in the decision making.

In inductive learning, machine learning techniques are often divided into *supervised, unsupervised* and *reinforcement learning.* These groupings of Machine Learning techniques are based on how the system learns from the examples that are provided. In supervised learning, the computer as a learner is led by a controller, which provides guidance on actions. Examples of inputs and the desired outputs are presented. The system will then learn from the given data to find a general rule that maps the inputs onto the outputs. In unsupervised learning, there is no concept of the target data. The learning process is performed by using only the set of input data. In reinforcement learning, the learner searches an optimal model through interaction with a dynamic environment.

The algorithms that are commonly used in each category are:

1. Supervised learning: Backpropagation, Bayesian statistics, Case-based reasoning, Decision trees, Nearest Neighbour Algorithm, Support vector machines, Random Forests, Information fuzzy networks (IFN)

2. Unsupervised learning: Artificial neural network, Data clustering, Expectation-maximisation algorithm, Self-organising map, Radial basis function network, Generative topographic map, Information bottleneck method

3. Reinforcement learning: Temporal difference learning, Q-learning, QV-learning, Sarsa or Expected-Sarsa, Actor-Critic, Acla, Cacla

2.6 Inductive Logic Programming

The term Inductive Logic Programming (ILP) was introduced by Muggleton [90] as an intersection of machine learning, especially inductive learning, and logic programming [93]. Golem [95] was the first ILP system to be applied to a wide variety of real-world applications. ILP has its theoretical roots in computational logic [79] and it has now been well studied in both machine learning and logic programming for two decades [94]. Earlier work on model inference by Shapiro [121] and inductive generalisation by Plotkin [101] contributed to the introduction of ILP as a research area. Nowadays, ILP research not only works in fundamental theories, but has also solved many problems in real-world applications [47].

Inductive concept learning. *Induction* as opposed to deduction, means infering from specific facts or instances to general principles. In machine learning, learning general rules from a set of given examples is called *inductive learning* (also called supervised learning), where the general principles are expressed in some logical language, e.g. FOL, logic programs, or DLs. *Inductive concept learning*, which is a more specific problem setting than inductive learning, is a task in which the main goal is to find a logical description of a concept from instances (and non-instances) of that concept [69].

ILP representation. In logic programming, a clause refers to a disjunction of literals, which may be either positive or negative. They can be used in two ways:

- as disjunctions: e.g. $p \vee q$
- as implications: e.g. $\neg p \rightarrow q$

A Horn clause is a clause with at most one positive literal, i.e.:

- only one positive literal e.g. $[\neg p_1, \neg p_2, \ldots, \neg p_n, q]$
- no positive literal e.g. $[\neg p_1, \neg p_2, \ldots, \neg p_n]$ and $[\,]$

Note that $[\neg p_1, \neg p_2, \ldots, \neg p_n, q]$ is a representation for $(\neg p_1 \vee \neg p_2 \vee \cdots \vee \neg p_n \vee q)$ which is equivalent to $p_1 \wedge p_2 \wedge \cdots \wedge p_n \rightarrow q$. In Prolog, it is written as: $q : -p_1, p_2, \ldots, p_n$. A Horn clause with exactly one positive literal is a definite clause; a Horn clause with no positive literals is sometimes called a goal clause, especially in logic programming.

ILP systems use logic programming, which is a subset of FOL. Logic programming is used to represent examples, background knowledge and hypotheses. As an overview, logic programming [78] is a computer programming paradigm which uses formal logic to express statement. Major languages in logic programming include Datalog, Answer Set Programming (ASP) and Prolog. The languages of logic programs provide sufficient expressiveness for solving problems in relation learning [69]. These logic programming systems use *Horn* clauses to represent problems. ILP can generate hypothesis from given positive and negative examples, background knowledge and clauses of hypothesis.

Positive examples are ground literals that are labelled as true by the user, while negative examples are ground literals that are stated to be false by the user. Suppose we have a set of positive examples E^+ and a set of negative examples E^-, we can define the logic program as consistent and complete as stated in [17] using the definitions below:

> A *logic program* P is complete (with respect to E^+) if and only if (*iff*), for all examples $e \in E^+$, $P \vdash e$.

A *logic program* P is consistent (with respect to E^+) *iff*, for no example $e \in E^-$, $P \vdash e$.

When using ILP, we need a *hypothesis language* to represent the hypothesis space we expect the system to learn and *background knowledge* which refers to declarative prior knowledge. The hypothesis language, and indirectly the background knowledge, determine the search space of possible concept descriptions. A *hypothesis language* in general machine learning, mentioned in [21], is defined as:

The language in which the hypotheses (also referred to as patterns or models) it outputs are described.

Formally, Muggleton [91] describes the ILP task using the notation below:

Given: a hypothesis language L_H,
Find: a logic program (hypothesis) $H \in L_H$ which shall follow the conditions:

- Necessary: $B \not\models E^+$
- Sufficient: $B \wedge H \models E^+$
- Consistent (either weak or strong)
  - Weak consistent: $B \wedge H \not\models \Box$
  - Strong consistent: $B \wedge H \wedge E^- \not\models \Box$

Where B is background knowledge, E^+ are positive examples and E^- are negative examples. The above symbols are read as: $\wedge$ (logical and), $\models$ (entails/logically proves), $\Box$ (falsity). The necessary condition is to ensure that no set of positive examples appears in the background knowledge. The sufficient condition is a requirement of the system to produce the hypothesis together with the background knowledge that satisfies positive examples. The weak and strong consistencies are the expression of how strict the system is in accepting noise in the hypothesis coverage. The unique feature of ILP that allows us to define background knowledge separately from the target predicate, and use them in the learning process, makes it more beneficial for the problem of learning relations.

2.7 Active Learning

With supervised learning, all data points are labelled, so that the system can learn and find a general model in the training data. In some cases labelling data manually takes a long time and it is very expensive. It may be necessary to hire a human expert to label each of them, especially when the data is abundant and in various formats. For example, to build document classification from a website, it will take some time to classify which document should be labelled as 'sport', 'news', 'gossip', etc. In such cases, the learner algorithm may actively choose which data points to label and ask the *oracle* to label them. This type of iterative supervised learning is called Active Learning (AL). The chosen data points for labelling are called *queries*. AL continually develops and tests new hypotheses as part of the interactive learning process.

As explained in [117], AL strategies can be categorised into the two major types: *stream-based* active learning and *pool-based* active learning. In *stream-based* active learning, one unlabelled example is considered at a time and then the learner decides whether to query or ignore it. While in the *pool-based* active learning, a large pool of unlabelled examples is gathered at once. It is then ranked by the informativeness. The data point which has the highest rank will be queried first. The computing resources and the type of the data are the main considerations when deciding which type of AL strategy to be used. However, the pool-based type is much more common to be applied. But there are some conditions that the stream-based approach is more appropriate e.g. when the memory or processing resource is limited or when the data set is too large and has to be scanned sequentially from disk.

According to [96], there are three different categories to select the next queries (called *selective sampling*), namely:

1. Uncertainty reduction
   data points are queried based on the least confidence prediction produced by the learner (the most uncertain).

2. Expected-error minimisation
   the learner selects the data points which will minimise future errors.

3. Version space reduction
   queries are made by selecting the one that can reduce the version space as much as possible.

Uncertainty sampling. Uncertainty sampling is one of the well-known methods in uncertainty reduction. This technique was introduced by Lewis and Cattlet [73]. Uncertainty sampling focuses on selecting the data points which are most uncertain. Therefore, a measurement of uncertainty to define uncertainty regions is needed for selecting candidate queries in the pool. Setting the threshold for defining uncertainty region is also important. There are many ways to measure uncertainty, such as distance from the hyperplane and label probability, denote by $P_\theta(y\,|x)$, read as the probability under θ model that the data point x will be given the label y. The following methods can be used for an algorithm which uses label probabilities:

- Least Confident:

  $$x^*_{\text{LC}} = \text{ argmax}_x\ 1 - P_\theta(\hat{y}\,|x)$$

  where $\hat{y} = \text{ argmax}_y\ P_\theta(y\,|x)$ or the class label with the highest posterior probability under the model θ. This method is querying the data points whose predicted output is the least confident (the most uncertain).

- Smallest Margin:

  $$x^*_{\text{SM}}\ =\ \text{argmin}_x\ P_\theta(\hat{y_1}|x)\ -\ P_\theta(\hat{y_2}|x)$$

  In this formula, $\hat{y_1}$ and $\hat{y_2}$ are the first and the second most probable labels for data point x under the model θ. This strategy uses the ambiguity of the possible label as the uncertainty measure. The data point with the small margin is more ambiguous. Therefore, it can provide the most information for the learner.

- Label Entropy:

  $$x^*_{\text{LE}}\ =\ \text{argmax}\ _x\ \sum_i P_\theta\ \ (y_i|x)\log P_\theta(y_i|x)$$

  In this formula, y_i ranges over all possible labels. Entropy [120] is a measure of a variable's average information content. This uncertainty sampling method chooses the data point with maximum label entropy. It is often considered as a measure of impurity in machine learning.

Error/variance reduction. In this method, the data points to be labelled are selected based on how well they can predict future labelling. In other words, the learner decides to pick out the questions that once they know the answer, can minimise future errors. The expected error reduction strategy was proposed by Roy and McCallum [110]. In error reduction, the learner identifies all possible outcomes and computes a weighted sum to give an expected value for each option. They then choose the best-expected value which means the lowest expected future error.

Despite its ability to produce a more accurate classifier with less labelled data, the expected error reduction, in most cases, is very costly. The cost for estimating the expected future error for each query in the pool is very high. In addition, as any new model must be re-trained for every possible labelling of every possible query in the pool, the computational cost gets higher and higher.

In some cases, the expected error still can be reduced by considering the regression problems. Geman et al. [40] describe the three basic components of the learner's expected error that correlated with each other, *noise* (unreliability of the true label), *bias* (error due to the model class) and *variance* (squared-loss with respect to the target function). The first two components, *noise* and *bias* are out of learner's control. The only component which can be controlled by the learner is *variance*. The learner should minimise the variance to reduce the expected error value. For classification problems, the Fisher Information Ratio can be used for variance reduction.

Query by committee and disagreement-based method. *Hypothesis* in machine learning is defined as a rule or model that is built to explain the training data and make future predictions on the new data. Version space reduction works by generating a committee of several hypotheses and makes queries on which the committee most disagrees. In a binary classification problem, this strategy means making queries that remove approximately half of the version space. Query by Committee (QBC) is a method in version space reduction. Instead of using one hypothesis to query the most informative data points to be labelled, as in uncertainty sampling, QBC considers more than one hypothesis that is available in the *version space*. Version space is defined by Mitchell [86] as the subset of hypotheses which are consistent with the training data.

QBC was introduced by Seung et al. [118]. It was developed based on the disagreement heuristics model [28]. The *committee* that is referred to by this strategy consists of some hypotheses members in the version space that is chosen using ensemble methods such as *Random forests*, *Bagged classifiers*, etc. The main idea of QBC is querying the data points which can reduce the number of hypotheses in the version space. The favoured data points are queried based on the degree of disagreement in the committee, which is measured by a measurement such as *Entropy of predicted responses* or *KL-divergence* of

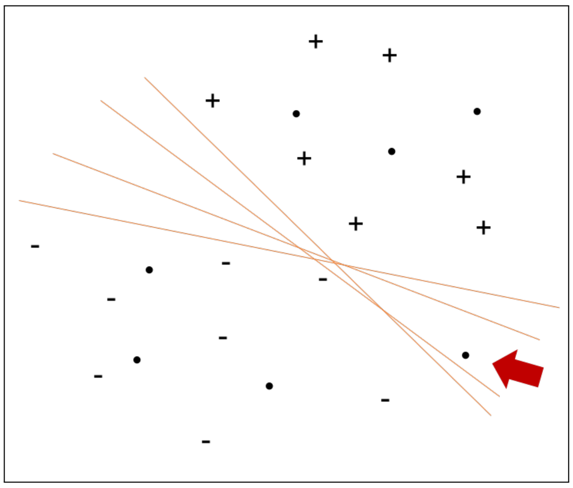


Figure 2.4: Query by Committee

predictive distributions. The QBC selection method is illustrated in Figure 2.4. In the figure, the data point to be selected is the one that the classifier line mostly disagrees with (shown by the red arrow).

2.8 Summary

In this chapter, the foundational theory of machine learning and RSs is presented, so that the scope and field of the research in this thesis can be clearly understood.

The basic knowledge described in this chapter includes an introduction to the machine learning method, representation and the problem. To understand the method used in this thesis, an explanations of machine learning tasks, ILP and AL are presented. A formal knowledge representation is also introduced which includes the theory of DLs and also the related research area of ontologies and their implementation in e-commerce, which will be useful to support our motivation for using DL representation in an RS. Finally, to understand the problem, research and applications of RSs and studies of PCs are provided. Related work and the state of the art in the two research fields of machine learning and RSs, will be described in the next chapter.

Chapter 3

Related Work

In this chapter, recent studies related to recommender systems and machine learning, particularly in preference learning and ILP in DL, are discussed to analyse the current state of the art. Preference learning as a research field in machine learning is described in this chapter. The use of pairwise comparisons in preference learning has been explored in the decision-making area as well as in the recommender systems research. Related studies in the recommender systems area, including the use of ontologies, active learning, the aspect of explainable information and pairwise comparisons, are presented. This chapter also discusses related work in ILP using DL representation, namely DL-Learner and its refinement operator, to identify the capability of existing systems to address problems in pairwise preference learning.

3.1 Preference Learning

Preference Learning (PL) [39] is a field in machine learning where the main task involves inducing predictive preference models from empirical data. In general, a PL task involves predicting preferences for a new set of items from a known value of preferences in existing items. Most commonly, the predicted preference relation is in the form of a total order ranking problem. One of the most important areas in this field, called "label ranking" [61], proposes a method to predict the order of preference from a given set of label preferences. Similar to label ranking, in research by Kamishima [65], the user is asked to provide a general order of preferences. A method called Nantonac Collaborative Filtering is proposed to produce recommendations based on the general ordering preferences given by the user.

According to [32], there are two main ways of modelling preferences, quantitative and qualitative preferences. The first model is associated with a number (or a quantity) representing its worth (e.g., "my preference for car type is a sedan"), while the second type of modelling relates to each other via pairwise comparisons (e.g., "I prefer car 1 over car 2"). The first model is not easy for everyone though, since humans are not always comfortable in expressing their preferences directly in terms of a value. It is normally much easier and arguably more natural to provide information about preferences in separate pieces, preferably in a qualitative way. In practice, this is achieved through queries consisting of pairs of items along with their descriptions, where the user only needs to select the better of the two items. The use of pairwise comparisons is still limited, not only because the approach has not yet been adopted by the major e-commerce companies, but also because choosing the most useful pairs and building a hypothesis about user preferences still needs attention.

The use of Pairwise Comparison (PC) can minimise the inconsistency of users providing numerical ratings. In the PL field, even though this method seems more comfortable for some users to express their preferences, the approach to how the learning algorithm will assess the quantitative information still remains a challenge. If the pairwise preferences are satisfied by the transitivity property of an order relation, then the ranking of preferences can be produced easily. For example, considering preference relation "$\succ$", if $A \succ B$, $B \succ C$ and $A \succ C$ then $A \succ B \succ C$. But there are some intransitive relations that occur in pairwise preferences which are possible to make a cycle. For example, if $A \succ B$, $B \succ C$ and $C \succ A$ then the ranking of preferences between A, B and C cannot be determined. Researchers agree that approaches to this kind of problem are different from ranking individual objects which are evaluated using their individual utility values [39].

Pairwise comparisons can also be used to learn about community preferences. Abbasnejad et al. [1] decompose user populations into communities of shared preferences and build user preference models as an infinite Dirichlet Process (DP) mixture of communities. The proposed algorithm is evaluated against the previous full Gaussian Process by [48] and the results show that it scales better, with an accuracy as good as the previous approach.

PC can also be used in Multi-Criteria Decision Making (MCDM). It can be used to estimate the decision maker's preferences and solve problems in prioritising alternatives. The PC method is used as an intermediate step for measuring intangible criteria in MCDM. Siraj [122] explores the problem of prioritisation as an optimisation problem and proposes a new method to prioritise alternatives using a graph-theoretic approach. The relative importance of two elements is assessed by the decision maker using a *ratio judgment*. The ratio explains how much an element is preferred to the other element. Ratio judgements can be represented using either a matrix or a graph. When represented using matrix notation, the decision maker should provide a complete set of judgements. If an incomplete judgement exists, a method to fill in the gaps should be performed. By using graph representation, the matrix judgement can be drawn as a fully connected digraph. Not only addressing the prioritisation task, but also the problem of selecting the best item from multi-attributes choices has also been commonly solved using a decision support system, one of which is implemented by Bohanec and Rajkovic [22]. This uses a dataset of cars to evaluate user preferences. A well-known MCDM method, PROMETHEE (Preference Ranking Organisation Method for Enrichment Evaluations) also uses PC to achieve the goal i.e. rank the alternatives. Please refer to [16] for more details on the literature survey of this method.

3.2 Related Work in Recommender Systems

3.2.1 Pairwise preference in recommender systems

The idea of using comparisons in a Recommender System (RS) is still growing. However, some users may find it difficult to articulate their preferences. Some users are also uncomfortable in giving comments and numerical ratings to an item. There are also tendencies that users are not consistent in rating items. The most recent work on pairwise recommender system was published in 2018 by Pan et al. [99] which proposes an algorithm called CoFiSet (Collaborative Filtering via Learning Pairwise Preferences over Item-Sets). Instead of using a single item, they use pairwise preference on a set of items (item-set). Qian et al. [103] also published a study on pairwise comparisons as a preference elicitation method. They performed an experiment using one thousand items from Yahoo Used Car and propose an approach to learn user preferences using orthogonal queries to select pairs. Linear SVM is then used to approximate the preference. Jensen et al. [63] use pairwise comparisons in a music recommender system. They apply a Gaussian Process regression model for comparisons between music tracks. They propose a new method to improve classic collaborative filtering which is based on single ratings. Similar work was performed by Rokach and Kisilevich [109]. They use a Lazy Decision Tree with pairwise comparisons at each node. Their results show that pairwise comparison is better than a single-item-based approach. Pairwise comparison is also used to simplify the process of building a user preferences profile, such as research by Jiang [64]. In that work, a pair of items is shown to users to judge which one they like most. This method can minimise users' confusion when choosing from a number of items.

In RS research, pairwise preference has not been broadly implemented in a real system, as rating based systems are more popular. Many well-known websites like Amazon, eBay and YouTube use ratings as an input to produce recommendations. However, using pairwise preferences as a method to rank has been analysed in other domains, such as ranking team sports [68]. The main goal of this method is to produce a global ranking. This method can also be used in RS, but instead of getting a personalised recommendation, all users will get the same (general) order in the recommendation list.

Fang and Si [34] successfully used pairwise comparisons to solve the implicit feedback problem. An experiment was conducted on the online scientific community dataset, nanoHUB. It was not necessary for the user to complete many tasks before getting a recommendation. The implicit feedback used here is *click-through* data. The pairwise approach is utilised to compare the probability of the relevance of two resources. When a recommended item is accessed by the user, it can be assumed that the item is more relevant than the others, despite its ranking position. The pairwise preferences were modelled using a *logistic function* of features. A set of items was used rather than the individual rating score of each item. Fang and Si claim that it is the first learning to rank method for real world recommender systems with implicit feedback.

Pairwise comparison is also used to mine user preferences by utilising the Bradley Terry model as a measure of the competitive ability in pairwise comparisons [64]. This model uses the content features of each item. They perceive the process of mining a user's preferences as the competition problem in game theory. If item A has a stronger competitiveness than item B, the user tends to give a higher rating to A. The competition occurs between the different content features of two items. After a user gives a rating several times, the next user preferences value is estimated by summing the value of each feature of the new item using k-nearest neighbour classification.

With personalisation as the core of RS research, Blėdaitė [20] combines the use of a rating-based RS with pairwise preferences, so that each user receives a different recommendation based on their preference profile. A method called Personalised Differential Matrix with Ratings and Pairwise Preferences (PDMRPP) is proposed. It was evaluated in the film domain, combining ratings with pairwise preferences and incorporating those to the collaborative filtering method resulting in better accuracy. A formula that gives a higher score to a more popular film with very diverse ratings is proposed. Those movies with the highest score will be shown, to be rated by the user.

3.2.2 Ontologies in recommender systems

Research into improving recommender systems using ontologies includes Middleton et al. [84], who explain the basic understanding of how ontologies have been used in recommender systems in recent years. As a research interest, the idea for incorporating ontologies into RSs is relatively new.

The literature also has research on the use of interlinking between concepts to build a cross-domain RS. Tobìas [131] developed a recommender system that suggests music artists and compositions for places of interest using structured information from Linked Data repositories, DBpedia. The semantic graph/network is used for knowledge representation. To aim the objectives, metrics are developed to measure how far the semantic relations between items are meaningful. Similar to Tobias's work, Moe and Aung [88] also develop a cross-domain recommender system by utilising ontologies to link facial skin problem (as the problem) and related cosmetics (as the target). The Ford-Fulkerson algorithm is used to bridge between two domain ontologies.

In the film domain category, Ostuni et al. [98] present Cinemappy, a location-based system that computes contextual movie recommendations. The content-based engine employs graph information in DBpedia. Geographic criteria are exploited, instead of a simple geographic distance, to enrich the location-based information. Cheng et al. [26] have also developed an ontology-based RS for film recommendation, called AOPRS (Adaptive Ontology-Based Personalised Recommender System). They built their own film domain ontology, MyMovieOntology (MMO) in OWL, based on IMDb (Internet Movie Database) and used it as input data and search space. Related work is found in [97], which proposes a hybrid recommendation algorithm for films and music to compute top-N recommendations from implicit feedback. The algorithm is called SPrank. Similar to Cinemappy, SPRank also uses DBpedia to extract semantic path-based features. A learning-to-rank algorithm is then used to compute the recommendation.

An ontology-based recommender system is also implemented in the library domain. PORE (Personal Ontology-Based Recommender) which is developed by Liao et al. [74]

has the main goal of giving personalised recommendation in the Chinese library. It builds a personalised ontology for each user based on their preferences. In 2015, Chen published a paper on optimising PORE using the MapReduce algorithm [25].

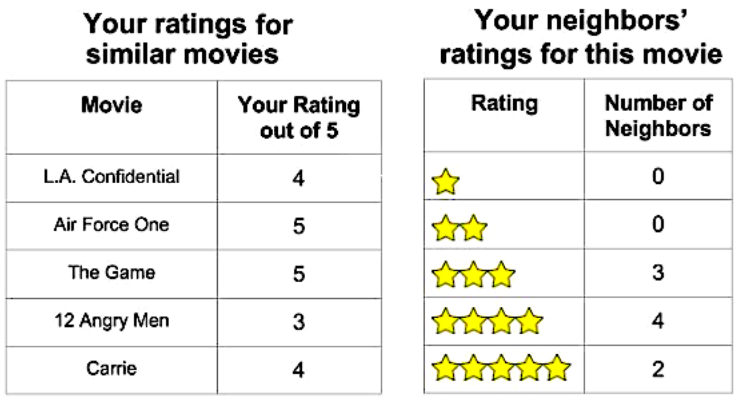
Ontologies are also used to overcome the problem with missing user preferences, called *Ontology Filtering* [115]. The information is captured by the topology of the ontology and then used to estimate the missing preferences. When the missing user preference has been completed, a recommendation will be made.

Further related research on ontology RSs identified by Lops et al. [80] are:

- SiteIF - a personal agent for multilingual news website
- ITR (Item Recommender) – a system that recommends items in several domains
- SEWep (Semantic Enhancement for Web Personalisation) – a web personalisation system that uses usage log and semantics of web contents
- Quickstep – a recommender system for online academic research papers
- Foxtrot – extends Quickstep by improving the interface and implementing an email notification
- Informed Recommender – a recommender system that transforms consumer's product review into a structured form
- News@hand – a system to recommend news
- A recommender system for interactive Digital Television
- JUMP System – intelligent delivery of information to knowledge workers

3.2.3 Explainable recommender systems

Many recommender systems provide recommendation as black boxes without explaining how the learning algorithm found the solutions. As explained in [130], nowadays there is a need to develop explainable recommender systems which can provide transparency where the reasoning and data behind a recommendation algorithm are presented to users. In Amazon, the explanation is provided as: "Customers Who Bought This Item Also Bought ...". In addition to transparency, user trust and loyalty can also be increased with explanations, as well as increasing user-satisfaction, it makes it quicker and easier for



Your ratings for similar movies

Movie	Your Rating out of 5
L.A. Confidential	4
Air Force One	5
The Game	5
12 Angry Men	3
Carrie	4

Your neighbors' ratings for this movie

Rating	Number of Neighbors
★	0
★★	0
★★★	3
★★★★	4
★★★★★	2

Figure 3.1: Explanation interface for ACF

users to find what they want and persuade them to try or purchase a recommended item. Current research on explanation interfaces for Automated Collaborative Filtering (ACF), such as shown in Figure 3.1, is proposed in [2, 18, 51]

Criteria for explanations in recommender systems are discussed in [130]:

1. transparency: explain how the system works
2. scrutability: allow users to tell the system it is wrong
3. trust: increase users' confidence in the system
4. effectiveness: help users make good decisions
5. persuasiveness: convince users to try or buy
6. efficiency: help users make decisions faster
7. satisfaction: increase the ease of use or enjoyment

3.2.4 Active learning in recommender systems

In recommender systems, the problem can be considered as a classification problem [33]. It can also be treated as a regression problem, since the rating uses discrete numerical values [111]. The recommendation results are obtained by using several machine learning techniques depending on the rating information [64], for example neural networks, association rules, nearest neighbour, Rocchio algorithm, Decision Tree, linear classifier and Naïve Bayes.

In the recent literature, recommender systems consider the use of active learning as a part of machine learning that allows the system to interact with users in the preference acquisition process [111]. A very common approach to elicit user preferences is by asking users to give ratings or feedback. Unfortunately, this approach cannot be used to span the items presented to the users. Instead of considering the interactive process for eliciting user preferences as an intrusive process, there are alternative ways to perform this using exploratory processes. Some systems provide a "surprise me!" button to motivate the user.

Based on the different motivations for achieving the goal, Rubens et al. [111] divide the use of active learning in recommender systems into three types, uncertainty-based, error-based and ensemble-based. The motivation to reduce uncertainty may not always align with accuracy improvement. The uncertainty-model is used to find out which thing is the wrong one. This approach is used to reduce uncertainty in rating estimates, decision boundaries and model parameters. The second approach is to reduce the predictive error by using the relation between one of these instruments, error and the change in the output estimates, test set error, change in parameter estimates and variance of parameter estimates. While the third approach attempts to identify useful training points based on consensus between models in the ensemble or multiple candidate models. In addition, they discuss conversation-based active learning that can be used to sharpen the user preferences until the most desired item is found.

3.3 DL-Learner

Description Logics (DLs) are language that can be used to represent knowledge in a structured, formal, yet understandable way. Because of their well-defined semantics and powerful reasoning tools, DL has been chosen as the best representation of ontologies. The name description logics is taken from the fact that they provide a formal way to represent the important ideas of the application domain as concept descriptions. The term *logic* also means that they are equipped with a formal, logic-based semantics as explained in [11]. Many DLs can be considered as fragments of First Order Logic.

The implementation of ILP in DL has successfully shown some good results on learning about concepts. One such result is DL-Learner [70] which aims to find a correct class description in a specific ontology by given a set of positive and negative examples of the individuals. It builds a hypothesis in the form of class descriptions (axioms) which can contain conjunctions, disjunctions and/or existential quantifications. DL-Learner is another improvement on previous ILP implementations in DL, such as YinYang [62] and DL-FOIL [35]. Kietz [66] and Konstantopoulos [67] also work in ILP in DL.

DL-Learner is proposed by Lehmann [70] to learn concepts in description logics and OWL. It is published as a part of his thesis [71]. The tool implementation of this framework is publicly available[1] and can be accessed free under General Public License (GPL). DL-Learner works by taking input from a knowledge base and a set of examples in the knowledge base and then proceeding into the learning step using the ILP approach. The output produced by the DL-Learner is a set of valid descriptions from a given positive and (if any exist) negative examples. When learning about a concept description of class A, it defines the negative examples in two ways:

- under the Open World Assumption (OWA), the negative examples consist of all individuals which can be proved to be an instance of the negation of class A

- under the Closed World Assumption (CWA), the negative examples are a collection of individuals, which cannot be proved to be an instance of class A. Applying CWA is usually preferred in those systems to learn a good description of a set of instances (examples).

Lehmann [71] introduces the two types of refinement operators used in DL-Learner, they are, the complete OWL refinement operator and the ideal $\mathcal{EL}$ refinement operator. The first type is the most expressive operator, designed for most OWL structures, while the second type is designed to support the lightweight DL language, $\mathcal{EL}$, aiming at the completeness, properness and finiteness of an ideal refinement operator. The second type is the first published ideal (i.e. finite, complete and proper) refinement operator in description logics. Further information about properties of DL refinement operators can be found in [71].

To explain how the DL-Learner complete OWL operator works, the following symbols are used:

- ρ is the operator which will be applied recursively to traverse the structure of the concepts

[1] http://dl-learner.org/

- $sh_{\downarrow}(A)$ is a downward refinement operator in the subsumption hierarchy of the concept A
- $sh_{\uparrow}(A)$ is an upward refinement operator in the subsumption hierarchy of the concept A
- $ar(r)$ is an atomic range of a role r
- $ad(r)$ is an atomic domain of a role r
- N_C is a set of concept names
- A, A', B, C and D is a concept
- M is a set of concepts
- mgr_B is a set of most general applicable roles with respect to a concept B
- r is a role
- $\top$ is `OWL:Thing`
- $\bot$ is `OWL:Nothing`

The DL-Learner OWL complete operator ρ is applied to 8 different conditions:

1. if the current concept is $\bot$, then $\rho = \emptyset$
2. if the current concept is $\top$, then $\rho = \{C_1 \sqcup \cdots \sqcup C_n | C_i \in M_B (1 \leq i \leq n)\}$ in which M_B is the set of available concepts, defined as the union of the following sets:
   - $\{A \mid A \in N_C, A \sqcap B \not\equiv \bot, \text{there is no } A' \in N_C \text{ with } A \sqsubset A'\}$
   - $\{A \mid A \in N_C, \neg A \sqcap B \not\equiv \bot, \neg A \sqcap B \not\equiv B, \text{ there is no } A' \in N_C \text{ with } A' \sqsubset A\}$
   - $\{\exists r.\top \mid r \in mgr_B\}$
   - $\{\forall r.\top \mid r \in mgr_B\}$
3. if the current concept is a concept A in which $A \in N_C$, then $\rho = \{A' \mid A' \in sh_{\downarrow}(A)\} \cup \{A \sqcap D \mid D \in \rho(\top)\}$
4. if the current concept is a negated concept $\neg A$ in which $A \in N_C$, then $\rho = \{A' \mid A' \in sh_{\uparrow}(A)\} \cup \{\neg A \sqcap D \mid D \in \rho(\top)\}$

5. if the current concept is $\exists r.D$, then $\rho = \{\exists r.E \mid A = ar(r), E \in \rho_A(D)\} \cup \{\exists r.D \sqcap E \mid E \in \rho(\top)\} \cup \{\exists s.D \mid s \in sh_{\downarrow}(r)\}$

6. if the current concept is $\forall r.D$, then $\rho = \{\forall r.E \mid A = ar(r), E \in \rho_A(D)\} \cup \{\forall r.D \sqcap E \mid E \in \rho(\top)\} \cup \{\forall r.D \mid s \in sh_{\downarrow}(r)\} \cup \{\forall r.\bot \mid D = A \in N_C, sh_{\downarrow}(A) = \emptyset\}\}$

7. if the current concept is $C_1 \sqcap \cdots \sqcap C_n$, $(n \geq 2)$, then ρ=$\{C_1 \sqcap \cdots \sqcap C_{i-1} \sqcap D \sqcap C_{i+1} \sqcap \cdots \sqcap C_n \mid D \in \rho(C_i), 1 \leq i \leq n\}\}$

8. if the current concept is $C_1 \sqcup \cdots \sqcup C_n$, $(n \geq 2)$, then ρ=$\{C_1 \sqcup \cdots \sqcup C_{i-1} \sqcup D \sqcup C_{i+1} \sqcup \cdots \sqcup C_n \mid D \in \rho(C_i), 1 \leq i \leq n\} \cup \{(C_1 \sqcup \cdots \sqcup C_n) \sqcap D \mid D \in \rho(\top)\}$

In the DL-Learner ideal refinement operator for $\mathcal{EL}$ language, concepts are viewed as tree structures to achieve the ideality. The refinement operator ψ is defined as a function that maps a tree $t \in T_{min}$ to a subset of T_{min} which can be divided into three base operations, namely:

1. label extension, which means extend the destination vertex in the minimal tree

2. label refinement, which means refining the starting vertex in the minimal tree

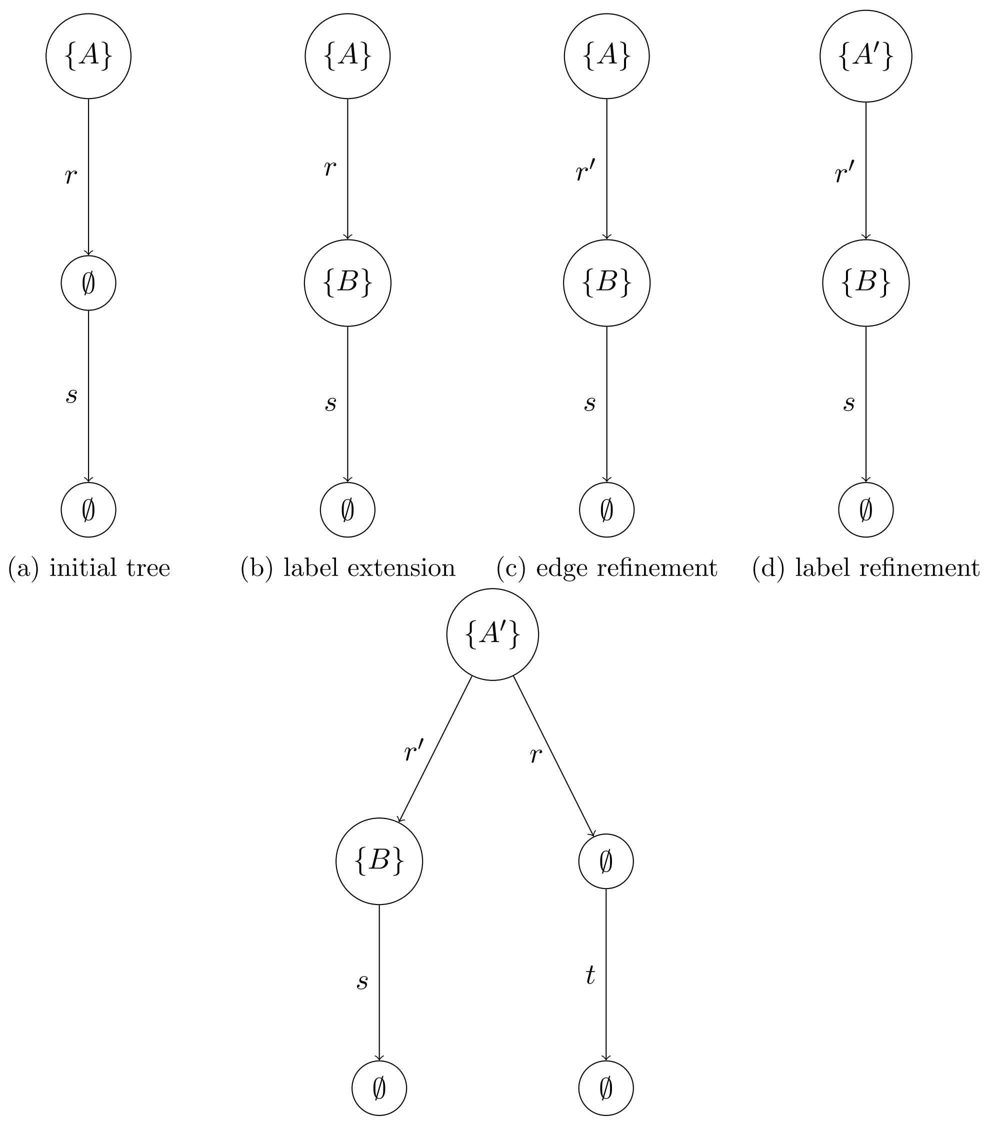
3. edge refinement, which means refining one of the outgoing edges

Assuming $A' \sqsubseteq A$ and $r' \sqsubseteq r$, the ideal refinement operator ψ is illustrated in Figure 3.2.

3.4 Summary

This chapter gives an overview of related studies in both the recommender system and machine learning areas. Pairwise comparisons, which can also be considered as binary relations, are commonly studied in some areas, such as MCDM, preference learning, recommender systems and ILP. In the preference learning field, the pairwise comparisons method is more commonly used for predicting the ranking. Several approaches exist to solve the ranking problem, but the use of pairwise comparisons to learn the order itself has not been widely studied.

From the literature in the recommender systems field, especially those papers which use a pairwise comparison technique, we observe that most studies use statistical machine learning approaches. However, there is a large potential to benefit from applying a logic-based approach, with the advantage of a more expressive representation. Therefore, we identify there is a research gap which can be filled here.



(a) initial tree (b) label extension (c) edge refinement (d) label refinement

(e) attaching subtree

Figure 3.2: DL-Learner $\mathcal{EL}$ refinement operator

While the implementation of pairwise comparisons in the recommender system area can be found in several studies, a commercial real-world application does not yet exist. Some of the general research challenges in the recommender systems field are also discussed in this chapter. One of these is to produce an explainable recommendation, which has the potential to be resolved using the advantage of logic representation. Several studies on the use of ontologies in recommender systems are also provided.

In the machine learning area, there exists an implementation of ILP in DL for learning about concepts, but the use of ILP in DL for learning about relations is still not popular. In this chapter, the DL-Learner framework and its refinement operators are reviewed. DL-Learner is somewhat related to the work in this thesis; while it learns about concepts, one contribution of this thesis is learning the Domain and Range axioms of a specific object property.

In the next chapter, the study on tasks of learning from binary relations using existing systems is described in detail.

Chapter 4

Learning Binary Preference Relations

The use of data that genuinely reflects user preferences is essential to the success of any recommender system. We used the user's answers to classify new unlabelled data and make a prediction about classes. The general annotation process is illustrated in Figure 4.1. It shows how we derived conclusions about preferences regarding individual attributes from data pairs of the form "Car 1 is-better-than Car 2". The bold arrow represents the annotation from the user and the dotted arrows show possible implications about individual attributes that the learning algorithm will consider.

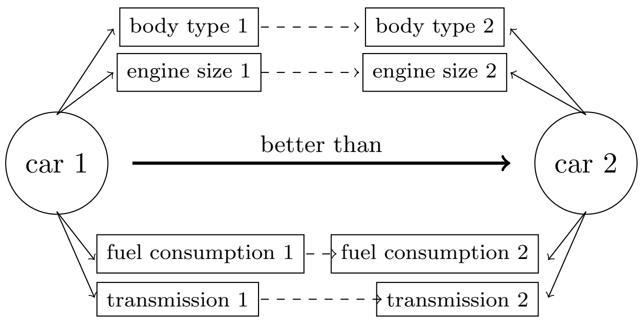


Figure 4.1: User annotation

In this chapter, user preferences are learned from a set of pairwise questions using a number of existing classification systems which fall into two categories of machine learning approaches, statistical and logic-based systems. The aim of this chapter is to explore the opportunities for using a logic-based approach in the recommender system area, which has not been widely studied, and also to compare it with a common statistical approach.

The two publicly available datasets [1,65] used for all offline experiments in this thesis are explained in this chapter. Finally, a comprehensive analysis of the benefits and drawbacks of systems that fall into those two approaches, statistical and logic-based, is also presented.

4.1 Problem Statement

Supervised learning is a type of machine learning algorithm in which the learner receives a set of labelled examples as training data and makes predictions for all unseen points. This matches the description of a Preference Learning (PL) problem in making a prediction about user preferences. One common type of supervised learning problem is binary classification, which is learned from two classes. In PL from pairwise comparisons, there are two classes to be learned, i.e. the “good” class and the “not good” class, but the nature of the pairwise dataset lends itself to two other learning tasks:

1. Learning the top preference class (the best of all)
   In this task, the characteristics of the group of items that do not have any better comparisons in the group was learned.
2. Learning to order pairs of items (the relative order of preferences)
   In this task, how items relate to other items through the “betterthan” relationship was learned.

In order to achieve the learning goal, the experiments in this chapter are divided into the two learning tasks described above.

4.2 Modelling Paradigms

AI research has tended to fall into two largely separate approaches, logical and statistical. The former tends to emphasis handling complexity, while the latter focuses on uncertainty [31]. The first approach represents knowledge symbolically and the system attempts to reason using the symbolic knowledge. Systems that fall into this category include, logic programming, description logics, classical planning, symbolic parsing, rule induction, etc. The second approach uses a mathematical function to build the model. Systems that fall into this category include: Naive Bayes, SVM, k-nearest neighbour, neural networks, etc. The mapping of how the different machine learning algorithms used in the experiment address the problem is shown in Table 4.1.

Table 4.1: Mapping of solution

Approach	Algorithm	Learning Task	
		top preference class	order of preferences
Propositional Logic	Decision Tree		✓
First-Order Logics	Aleph		✓
Description Logics	DL-Learner	✓	
Statistical	Bradley-Terry Model	✓	✓
Statistical	SVM		✓

4.2.1 Logic-based approaches

The experiments in this chapter were performed using existing systems from three families of logic:

1. Propositional logic

   Propositional logic is concerned with propositions and their interrelationships (logical connectives). The notion of a proposition here cannot be defined precisely. Roughly speaking, a proposition is a possible condition of the world that is either true or false, e.g., the possibility that it is raining, the possibility that it is cloudy, and so forth [41]. Learning in this logic also follows restrictions, i.e. both concepts and facts are expressed using a set of propositions and logical connectives. The examples and the hypothesis produced are enumerated in all possible values. One example of a learning algorithm which uses this logic is a Decision Tree (DT). The Decision Tree algorithm is included in the experiment because it is the most simple logic based algorithm that works with a white box system, which means that the hypothesis produced can be read. The number of observations in the dataset is not too large to be enumerated and so the possible rules (hypotheses) produced makes it suitable for learning with a propositional logic based system.

2. First order logic

   First Order Logic (FOL) is more expressive than propositional logic. It allows the use of variables and quantifiers to explain a concept. Inductive Logic Programming (ILP) is a learning algorithm which uses FOL, more specifically Horn clause expression, a special form of FOL representation. ILP uses this representation in both concept and object language. This makes ILP different from the propositional learning algorithm. Examples of the learning algorithm in these logics include FOIL [104], Aleph [125], Progol [92] and Golem [95]. Aleph was used here to learn binary classification.

3. Description logics

   Description Logics (DL) [11] define the world using *concepts* to represent a set of *individuals* and *roles* to represent the binary relations between individuals. A number of classification learning algorithms in DL have been introduced, including work by Fanizzi et al. [35], Kietz [66], Cohen and Hirsh [27], Badea and Nienhuys-Cheng [13], Iannone et al. [62] and Lehmann [70]. Research on specific pairwise preferences in DL is still very limited, such as the one introduced by Di Noia et al. [30], called Ontological CP-Nets. A DL-Learner [70] was used in this experiment to find a valid description of the top preferences class, the best group of cars according to the user (see Section 4.1). The DL-Learner uses a similar idea to ILP in the refinement operator to learn target classes.

4.2.2 Statistical approaches

In addition to logic-based learning, an experiment using statistical machine learning methods was also performed. The number of observations for each case (user) in the dataset (see Section 4.3) is sufficient to be solved using statistical machine learning approaches. For the statistical learner two existing systems are used, the Bradley-Terry model [23] to produce global ranking from pairwise data and SVM as a binary classification algorithm to predict the two classes of "better" and "worse".

The experimental results of the logic-based learning are then compared to the statistical machine learning approach. In a binary classification problem, the SVM searches for the optimal linear separator of all data points in an n-dimensional space then uses it to make predictions about new data. The method has previously been used by Qian et al. [103] in a similar setup in PL with good results.

4.3 Dataset

4.3.1 Car preferences dataset

A dataset of car preferences[1] provided by Abbasnejad et al. [1] in 2013 was used in this experiment. Ten items with 4 features[2] were used in their experiment. Each user provided answers for all 45 possible pairs of items, giving 90 observations for each user. This

[1]http://users.cecs.anu.edu.au/ u4940058/CarPreferences.html
[2]The engine capacity feature was discretised to simplify the learning process

comprises 45 positive examples from the user answers (e.g. car 1 is better than car 2), 45 negative examples from the opposite order of positive examples (e.g. car 2 is better than car 1). Users were presented with a choice to select one car over another based on their attributes. The data was collected from 60 different users from the United States using Amazon's Mechanical Turk. The 10 cars below were presented to the participants:

Table 4.2: Cars description used in Abbasnejad et al. [1] experiments

Item ID	Body Type	Transmission	Fuel Consumed	Engine Capacity	
1	suv	manual	non-hybrid	small	2.5L
2	sedan	automatic	hybrid	large	5.5L
3	sedan	manual	non-hybrid	medium	4.5L
4	sedan	manual	non-hybrid	large	6.2L
5	suv	manual	non-hybrid	medium	3.5L
6	suv	automatic	hybrid	medium	3.5L
7	sedan	automatic	hybrid	medium	3.5L
8	suv	automatic	hybrid	small	2.5L
9	sedan	automatic	non-hybrid	medium	3.5L
10	suv	automatic	non-hybrid	medium	4.5L

4.3.2 Sushi preferences dataset

The second dataset used in the experiments was about sushi preferences,[3] published by Kamishima [65]. The dataset contains individual user preferences for 10 different sushi types. Users were asked to sort the sushi according to their preference in ascending order. For the experiment, the pairs order was generated from each individual preference. Similar to the car dataset explained above, 45 pairs of sushi preferences are built from 10 types of sushi. The specification of sushi is presented in Table 4.3. In this dataset, 5,000 users were involved in the survey. However, in the experiment, only data from the first 60 users was used to make it comparable with the car experiemnt dataset.

4.4 Learning the Top Preference Class

Experiments in this section only use the car dataset because the original format of the dataset is in the form of partial orders. There is no full order in the dataset. This condition is different from the sushi dataset, where each user originally provided the full order of their choices.

[3]http://www.kamishima.net/sushi/

Table 4.3: Sushi description used in Kamishima [65] experiments

ID	Sushi name	Style	Major Group	Minor Group	Heaviness/ Oiliness	Frequency of Consumption	Normalised Price	Frequency of Sushi Sold
0	ebi (shrimp)	maki	seafood	shrimp/crab	2.7289	2.1384	1.8384	0.84
1	anago (sea eel)	maki	seafood	tare	0.9264	1.9902	1.9925	0.88
2	maguro (tuna)	maki	seafood	akami	1.7696	2.3485	1.8747	0.88
3	ika (squid)	maki	seafood	squid/octopus	2.6884	2.0432	1.5152	0.92
4	uni (sea urchin)	maki	seafood	other seafood	0.8130	1.6434	3.2873	0.88
5	ikura (salmon roe)	maki	seafood	roe	1.2649	1.9795	2.6957	0.88
6	tamago (egg)	maki	not seafood	egg	2.3681	1.8662	1.0325	0.84
7	toro (fatty tuna)	maki	seafood	akami	0.5519	2.0575	4.4855	0.8
8	tekka maki (tuna roll)	other	seafood	akami	2.2471	1.8790	1.5798	0.44
9	kappa maki (cucumber roll)	other	not seafood	vegetables	3.7305	1.4568	1.02	0.4

4.4.1 Bradley-Terry experiments

One of the most popular methods to predict global ranking from pairwise comparisons is a method introduced by Bradley and Terry [23]. An experiment was performed using the Bradley-Terry package in R [132]. The full order of preferences according to the BTM coefficient for all users in the car dataset [1] is reported in Table 4.4. To make it easier for the reader, the table only shows the full preferences order, while the BTM coefficients are attached as Appendix A. The $\succ$ notation is used to represent the order of strict preference towards the first mentioned choice. This comes from the calculation where the first choice has a higher BTM coefficient than the second choice. For example $i \succ j$ means i is more preferred than j. When there is no evidence that the preference is strong (no difference in the BTM coefficients), then $\succcurlyeq$ is used to show that the first choice has a weak preference or is equivalent to the second choice.

4.4.2 DL-Learner experiments

An experiment to learn the top preference class was also carried out with the DL-Learner by specifying a set of cars that have first place in the rank (see Table 4.4) as positive examples and the rest of the cars as negative examples. The best car characteristics for each user can be learned by using the DL-Learner. Please note that some users have more than one best car in their preferences. A simple ontology consisting of the item description and its membership was created as an input to be learned by the DL-Learner. More about

Table 4.4: Bradley-Terry experiment result

userID	full order of item (carID) preferences
user1	$7 \succ 8 \succ 2 \succ 9 \succ 6 \succ 3 \succ 5 \succ 1 \succ 10 \succ 4$
user2	$6 \succcurlyeq 10 \succ 5 \succ 2 \succcurlyeq 4 \succ 8 \succ 1 \succ 3 \succcurlyeq 7 \succ 9$
user3	$1 \succ 2 \succ 3 \succ 4 \succ 5 \succ 6 \succ 7 \succ 8 \succ 9 \succ 10$
user4	$8 \succ 6 \succ 7 \succ 10 \succ 2 \succ 9 \succ 5 \succ 1 \succ 3 \succ 4$
user5	$7 \succ 10 \succ 4 \succ 2 \succ 3 \succcurlyeq 6 \succ 9 \succ 5 \succ 8 \succ 1$
user6	$7 \succ 2 \succ 8 \succ 6 \succ 9 \succ 5 \succ 10 \succ 1 \succ 3 \succ 4$
user7	$8 \succ 6 \succ 10 \succ 7 \succ 2 \succ 9 \succ 1 \succ 5 \succ 3 \succ 4$
user8	$9 \succ 2 \succ 7 \succ 3 \succ 4 \succ 6 \succ 10 \succcurlyeq 8 \succ 5 \succ 1$
user9	$8 \succ 7 \succ 2 \succ 5 \succ 1 \succ 6 \succ 3 \succ 4 \succ 10 \succ 9$
user10	$4 \succ 3 \succ 1 \succ 5 \succ 9 \succ 10 \succ 7 \succ 2 \succcurlyeq 6 \succ 8$
user11	$9 \succ 10 \succ 4 \succ 3 \succ 5 \succ 2 \succ 1 \succ 7 \succ 6 \succ 8$
user12	$5 \succ 4 \succ 1 \succ 3 \succcurlyeq 6 \succ 9 \succcurlyeq 2 \succcurlyeq 8 \succcurlyeq 10 \succ 7$
user13	$8 \succ 6 \succ 10 \succ 7 \succ 2 \succ 9 \succ 1 \succ 5 \succ 3 \succ 4$
user14	$6 \succ 8 \succ 10 \succ 5 \succ 1 \succ 2 \succcurlyeq 7 \succ 4 \succcurlyeq 3 \succ 9$
user15	$2 \succ 7 \succ 9 \succ 6 \succ 8 \succ 10 \succ 3 \succ 4 \succ 5 \succ 1$
user16	$9 \succ 10 \succ 2 \succcurlyeq 7 \succ 6 \succ 8 \succ 3 \succ 4 \succ 5 \succ 1$
user17	$8 \succ 6 \succ 7 \succ 2 \succ 9 \succ 10 \succ 1 \succ 5 \succ 3 \succ 4$
user18	$8 \succ 6 \succ 1 \succ 7 \succ 2 \succ 5 \succcurlyeq 10 \succ 9 \succ 3 \succ 4$
user19	$6 \succ 8 \succ 10 \succ 2 \succ 7 \succ 9 \succ 1 \succ 5 \succ 3 \succ 4$
user20	$9 \succ 2 \succ 4 \succcurlyeq 10 \succ 7 \succ 6 \succ 8 \succcurlyeq 3 \succ 5 \succ 1$
user21	$10 \succcurlyeq 9 \succ 3 \succ 4 \succ 5 \succ 1 \succ 2 \succ 7 \succ 6 \succ 8$
user22	$8 \succ 6 \succcurlyeq 7 \succ 5 \succ 1 \succ 10 \succcurlyeq 2 \succ 9 \succ 3 \succ 4$
user23	$9 \succ 7 \succ 2 \succ 4 \succcurlyeq 3 \succ 10 \succ 6 \succ 8 \succ 5 \succ 1$
user24	$8 \succ 6 \succcurlyeq 10 \succ 2 \succcurlyeq 7 \succ 9 \succ 1 \succ 5 \succ 3 \succ 4$
user25	$6 \succ 7 \succcurlyeq 8 \succ 2 \succcurlyeq 10 \succ 9 \succ 5 \succ 3 \succ 1 \succ 4$
user26	$10 \succcurlyeq 6 \succ 8 \succ 2 \succ 7 \succ 9 \succ 5 \succ 1 \succ 4 \succ 3$
user27	$2 \succcurlyeq 4 \succ 3 \succcurlyeq 7 \succ 9 \succ 5 \succ 6 \succ 10 \succ 8 \succ 1$
user28	$5 \succ 1 \succ 6 \succ 8 \succ 10 \succ 3 \succcurlyeq 4 \succ 2 \succcurlyeq 7 \succ 9$
user29	$8 \succ 1 \succ 5 \succcurlyeq 6 \succcurlyeq 7 \succ 3 \succcurlyeq 9 \succ 10 \succ 2 \succcurlyeq 4$
user30	$2 \succ 10 \succcurlyeq 4 \succ 6 \succ 1 \succ 7 \succcurlyeq 3 \succ 9 \succ 8 \succcurlyeq 5$

Table 4.4: Bradley-Terry experiment result (cont.)

userID	full order of item (carID) preferences
user31	$2 \succ 6 \succcurlyeq 7 \succ 4 \succcurlyeq 10 \succ 8 \succcurlyeq 9 \succ 3 \succcurlyeq 5 \succ 1$
user32	$8 \succ 6 \succ 7 \succ 9 \succ 2 \succ 10 \succ 1 \succ 5 \succ 3 \succ 4$
user33	$10 \succ 5 \succ 6 \succ 1 \succ 8 \succ 4 \succ 2 \succcurlyeq 3 \succcurlyeq 9 \succ 7$
user34	$10 \succ 8 \succcurlyeq 6 \succ 9 \succ 2 \succ 1 \succ 4 \succ 7 \succ 5 \succ 3$
user35	$9 \succ 2 \succ 10 \succ 7 \succ 6 \succ 4 \succcurlyeq 8 \succ 3 \succ 5 \succ 1$
user36	$6 \succ 8 \succ 7 \succ 10 \succ 2 \succ 9 \succ 3 \succ 5 \succ 1 \succ 4$
user37	$9 \succ 3 \succcurlyeq 4 \succ 10 \succ 5 \succ 1 \succ 6 \succ 2 \succ 8 \succ 7$
user38	$6 \succ 8 \succ 7 \succ 2 \succ 10 \succ 9 \succ 5 \succ 1 \succ 3 \succ 4$
user39	$6 \succ 8 \succ 2 \succ 7 \succ 10 \succ 9 \succ 5 \succ 1 \succ 4 \succ 3$
user40	$2 \succcurlyeq 6 \succ 7 \succcurlyeq 8 \succ 10 \succ 9 \succ 4 \succcurlyeq 5 \succ 1 \succ 3$
user41	$7 \succcurlyeq 8 \succ 6 \succcurlyeq 9 \succ 5 \succcurlyeq 10 \succ 2 \succ 1 \succ 3 \succ 4$
user42	$6 \succ 8 \succ 10 \succ 5 \succ 1 \succ 7 \succ 9 \succ 2 \succ 3 \succ 4$
user43	$2 \succ 7 \succ 6 \succ 8 \succ 9 \succ 10 \succ 4 \succ 3 \succ 5 \succ 1$
user44	$3 \succcurlyeq 4 \succ 10 \succ 9 \succ 5 \succ 2 \succ 1 \succ 7 \succ 6 \succ 8$
user45	$10 \succ 6 \succ 5 \succ 1 \succ 8 \succ 4 \succ 3 \succ 2 \succ 9 \succ 7$
user46	$2 \succ 7 \succ 6 \succcurlyeq 4 \succ 3 \succ 10 \succ 8 \succ 5 \succ 9 \succ 1$
user47	$2 \succcurlyeq 7 \succ 9 \succ 6 \succ 3 \succ 4 \succ 8 \succcurlyeq 10 \succ 5 \succ 1$
user48	$6 \succ 8 \succ 2 \succ 5 \succcurlyeq 7 \succcurlyeq 10 \succ 1 \succ 9 \succ 3 \succcurlyeq 4$
user49	$3 \succcurlyeq 4 \succ 5 \succ 1 \succ 2 \succ 10 \succ 6 \succ 7 \succcurlyeq 9 \succ 8$
user50	$10 \succ 6 \succcurlyeq 8 \succ 7 \succ 2 \succ 5 \succcurlyeq 9 \succ 1 \succ 3 \succ 4$
user51	$2 \succ 4 \succ 10 \succ 6 \succ 7 \succ 5 \succ 9 \succ 8 \succ 3 \succ 1$
user52	$9 \succ 2 \succ 7 \succ 10 \succ 6 \succ 8 \succ 4 \succ 3 \succ 5 \succ 1$
user53	$2 \succ 6 \succ 7 \succ 8 \succ 5 \succ 10 \succ 1 \succ 3 \succ 9 \succ 4$
user54	$7 \succ 6 \succcurlyeq 8 \succ 2 \succ 1 \succ 5 \succcurlyeq 9 \succ 3 \succcurlyeq 4 \succcurlyeq 10$
user55	$7 \succ 6 \succcurlyeq 8 \succ 2 \succ 1 \succ 5 \succcurlyeq 9 \succ 3 \succcurlyeq 4 \succcurlyeq 10$
user56	$8 \succ 6 \succcurlyeq 7 \succ 2 \succ 1 \succ 5 \succ 10 \succ 9 \succ 3 \succ 4$
user57	$5 \succ 10 \succ 1 \succ 3 \succcurlyeq 4 \succcurlyeq 9 \succ 6 \succ 8 \succ 2 \succ 7$
user58	$10 \succ 7 \succ 9 \succ 3 \succ 8 \succ 6 \succ 2 \succ 1 \succ 4 \succ 5$
user59	$4 \succ 2 \succcurlyeq 10 \succ 3 \succ 6 \succcurlyeq 7 \succ 5 \succ 1 \succ 9 \succ 8$
user60	$5 \succ 6 \succ 1 \succ 10 \succ 8 \succ 4 \succ 2 \succ 3 \succcurlyeq 7 \succ 9$

the ontology representation of the pairwise preference problem is explained in Chapter 5 and the class hierarchy used in the DL-Learner is the same as shown in Figure 5.1a.

The CELOE (Class Expression Learner for Ontology Engineering) learning algorithm [71], which is the implementation of the DL-Learner OWL complete refinement operator, was used in this experiment to maximise the possibility of always finding a solution. The configuration file for `user1` used to run the DL-Learner is shown in Figure 4.2. The description of the top class for `user1` is shown in Figure 4.3. The DL-Learner experiment results from the first five users are shown in Table 4.5, while the complete results are shown in Appendix B. The DL-Learner produced several solutions with 100% accuracy, only the first suggested description is shown. It can be concluded that the best car for `user1` is the car which has the property values: `Automatic and Hybrid and MediumCar and Sedan`. The same method of putting the best cars as positive examples and the rest as negative examples in the configuration file was applied to all 60 users in the car preferences dataset.

```
prefixes = [ ("ex","http://www.mycars.org/ontology#") ]

// knowledge source definition
ks.type = "OWL File"
ks.fileName = "carpreferences.owl"

// reasoner
reasoner.type = "OWL API Reasoner"
reasoner.sources = { ks }

// learning problem
lp.type = "posNegStandard"
lp.positiveExamples = {  "ex:car7"   }
lp.negativeExamples = { "ex:car1", "ex:car2","ex:car3","ex:car4","ex:car5",
                        "ex:car8", "ex:car9", "ex:car6","ex:car10"    }
alg.type = "celoe"
alg.maxExecutionTimeInSeconds = 2
alg.maxNrOfResults = 10
```

Figure 4.2: DL-Learner configuration file example

```
Automatic and Hybrid and MediumCar and Sedan
```

Figure 4.3: DL-Learner learning results example

Table 4.5: DL-Learner experiment results

user ID	the best carID	DL-Learner result with 100% accuracy
user1	car7	Automatic and Hybrid and MediumCar and Sedan
user2	car6,car10	Automatic and MediumCar and Suv
user3	car1	Manual and SmallCar
user4	car8	Automatic and SmallCar
user5	car7	Automatic and Hybrid and MediumCar and Sedan

The aim of the experiments, to learn the top class using a BTM and a DL-Learner, is to show that existing systems in pairwise comparisons and DL can be used to address the learning problem in pairwise preferences. Cars preferred by the users can be seen from the BTM results only by considering the winner of each pair competition. This can be performed without examining the car features. By using this method, it is still possible to learn what is best for the users. In the same task, the DL-Learner results give more information on the user's most preferred car features. In recommender systems, it is important to understand what type of items are suitable for users, so that the correct recommendation can be made.

4.5 Learning to Order Pairs of Items

The second task was divided into two types of data, categorical only and mixed-type. The nature of pairwise preference data is usually mixed between numerical and categorical, but unfortunately not all learning algorithms can handle this type of data. Both representations have their own advantages and disadvantages. An experiment on the two different data representations (i.e. categorical and mixed-type) of both sushi and car preference datasets is described here.

4.5.1 Learning from categorical data

This experiment was carried out by using the SVM and CART DT [24] algorithms on Matlab R2016a running on Mac OSX version 10.11.2. For the ILP algorithm, Aleph 5.0 was run on a Prolog compiler: yap 6.3.2. The mapping from representations to the choice of machine learning algorithms and their implementation is shown in Table 4.6.

Table 4.6: Representation $\rightarrow$ algorithm $\rightarrow$ implementation

Representation	Algorithm	Implementation
Propositional Logic	Decision Tree	`fitctree` on Matlab
First-Order Logics	ILP	Aleph on yap
Statistical	SVM	`fitcsvm` on Matlab

4.5.1.1 Experiment setting with car dataset

Learning with only categorical data requires the dataset to be discretised and encoded to suit the learner input style. For the car dataset, the encoding was applied to the engine capacity attributes as shown in Table 5.1. The rest of the attributes still needed to be encoded to be fed into the SVM and DT-learner. The attributes encoding for the car dataset is shown in Table 4.7.

SVM and DT. The DT-learner can learn categorical data in any format (text or number) but in this experiment, the use of the numeric format was designed to make the encoding process easier, so that the same data format could be used with the other learning algorithm (i.e. SVM). The input for the DT is in pairs format with two classes, i.e.

Table 4.7: Values attributes encoding

body type (x1,x2)	transmission (x3,x4)	fuel consumption (x5,x6)	engine capacity (x7,x8)
1= suv	1= manual	1= hybrid	1= small
2= sedan	2= automatic	2= non-hybrid	2= medium
			3= large

"1" for positive examples and "-1" for negative examples. The data input as a numeric format is shown as: `2,1,1,1,1,2,2,2,1`, where the first 8 digits represent the attribute values (first two numbers are bodytype, second two numbers are transmission, third two numbers are fuel consumption and the fourth two numbers are engine capacity) and the last digit represents the class (positive or negative). A sample of the tree for `user1` was generated using the Matlab CART DT algorithm shown in Figure 4.4.

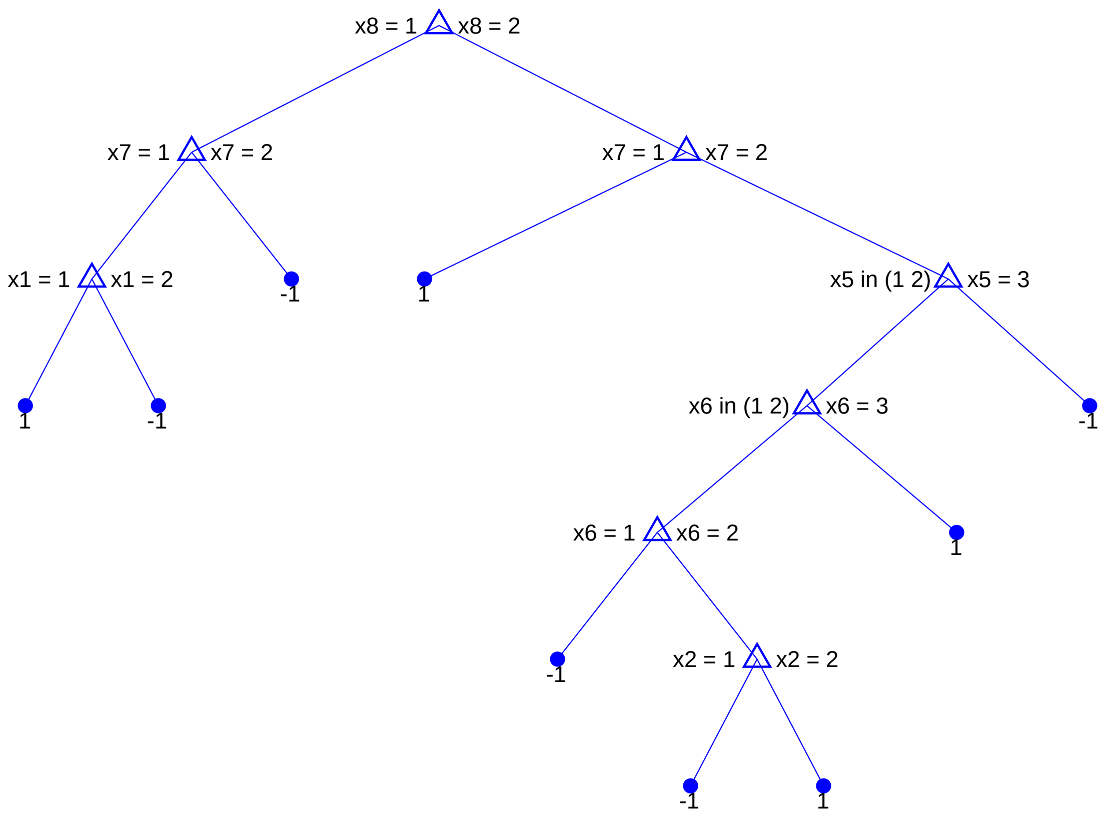
Using the same data conversion as shown in Table 4.7 the dataset was processed using Matlab SVM with the standard setting.

Aleph. Aleph uses separate files to differentiate between positive and negative examples. The positive examples are stored as an '.f' file extension, while the negative examples are stored as an '.n' file extension. The negative examples are the opposite relation of positive examples. All examples follow the logic programming format. The input for Aleph is shown in Figure 4.5. In the first line of the positive examples, it is shown that `bt(car1,car2)`, which means `car1` is better than `car2`. Likewise, in the first line of the negative examples, it is stated that `bt(car2,car1)` which means `car2` is **not** better than `car1`.

Three experiments were performed with Aleph. The flexibility to specify background knowledge and hypothesis language offered by Aleph, as one special feature of the ILP implementation, can be used to learn the pairwise PL in different ways. The three different settings in Aleph are:

- setting 1: learn from single attribute of each item
- setting 2: learn by comparing the same attributes from each item in pairs
- setting 3: learn from a mixed-type data format

The first two settings are explained here while the third setting will be explained in Section 4.5.2.



(a) DT rule in graph mode

```
1   if x8=1 then node 2 elseif x8=2 then node 3 else -1
2   if x7=1 then node 4 elseif x7=2 then node 5 else -1
3   if x7=1 then node 6 elseif x7=2 then node 7 else 1
4   if x1=1 then node 8 elseif x1=2 then node 9 else -1
5   class = -1
6   class = 1
7   if x5 in {1 2} then node 10 elseif x5=3 then node 11 else 1
8   class = 1
9   class = -1
10  if x6 in {1 2} then node 12 elseif x6=3 then node 13 else 1
11  class = -1
12  if x6=1 then node 14 elseif x6=2 then node 15 else 1
13  class = 1
14  class = -1
15  if x2=1 then node 16 elseif x2=2 then node 17 else 1
16  class = -1
17  class = 1
```

(b) DT rule in text mode

Figure 4.4: DT rule sample for car dataset

```
bt(car1,car2).
bt(car7,car8).
bt(car3,car1).
bt(car2,car3).
```

(a) File in '.f' extension containing positive examples

```
bt(car2,car1).
bt(car8,car7).
bt(car1,car3).
bt(car3,car2).
```

(b) File in '.n' extension containing negative examples

Figure 4.5: Aleph's positive and negative examples for car dataset

In the first setting, the learner is aiming to learn the `betterthan` relationship by using each single attribute value in every pair. For example, if the positive example says "`car1` is better than `car2`", then every attribute on `car1` and `car2` will be considered as a candidate hypothesis. An example of a possible candidate hypothesis will be: "`car A` is better than `car B` if `car A` is `sedan` and `car B` is `manual`". The values specified in the example are not necessarily values of the same attributes (`manual` is a transmission type and `sedan` is a body type). To make it clearer for the reader, an example of a supplied hypothesis language for Aleph with this setting is presented in Figure 4.7. A sample of the Aleph valid hypotheses with this setting is shown in Figure 4.8. Aleph stores the hypothesis language together with background knowledge in a file with a '.b' extension.

In the second setting, the learner considered corresponding attribute values only. For example, if the positive example was "`car1` is better than `car2`", then the learner only built the candidate hypothesis based on the same attribute comparisons from every pair. An example of a possible candidate hypothesis is: "`car A` is better than `car B` if `car A` is `sedan` and `car B` is `suv`". In the example, "`sedan`" and "suv" are the values of car body types. A hypothesis language specified for Aleph with this setting is provided in Figure 4.9 and a sample of the valid hypotheses is shown in Figure 4.10.

From the rules produced by Aleph, we can observe that the ILP model provides a more compact representation than the rules produced by the DT. In the above example, only the attribute value which has an association with the positive examples is defined (by the engineer) in Aleph's rules. This is different from the DT, where it uses Information Gain to select the important attributes from the data. Howell [58] states that the ILP rules can be viewed as shortcuts into the conclusions drawn by the decision tree. Additionally, they do not impose a hierarchy on rule importance as the decision tree would. This would be important in the case of missing data. If "node-caps" were a missing attribute in a query, the usefulness of a decision tree with that attribute at its root would be compromised. It

```
% type
fuelconsumption(hybrid).
fuelconsumption(nonhybrid).
transmission(automatic).
transmission(manual).
bodytype(sedan).
bodytype(suv).
enginesize(small).
enginesize(medium).
enginesize(big).

% type
car(car1).
car(car2).

% background knowledge
hasfuelcons(car1,nonhybrid).
hasfuelcons(car2,hybrid).
hasbodytype(car1,suv).
hasbodytype(car2,sedan).
hastransmission(car1,manual).
hastransmission(car2,automatic).
hasenginesize(car1, small).
hasenginesize(car2, large).
```

Figure 4.6: Aleph background knowledge in all settings

```
:- modeh(1,bt(+car,+car)).
:- modeb(1,hasfuelcons(+car,#fuelconsumption)).
:- modeb(1,hasbodytype(+car,#bodytype)).
:- modeb(1,hastransmission(+car,#transmission)).
:- modeb(1,hasenginesize(+car,#enginesize)).
```

Figure 4.7: Aleph hypothesis language in setting 1

```
bt(A,B) :-
   hasfuelcons(B,nonhybrid), hasbodytype(A,sedan),
   hastransmission(A,automatic).
bt(A,B) :-
   hasfuelcons(A,hybrid), hastransmission(B,manual).
```

Figure 4.8: A sample of Aleph valid hypotheses with setting 1

```
:- modeh(1,bt(+car,+car)).
:- modeb(1,carfuel(+car,#fuelconsumption,+car,#fuelconsumption)).
:- modeb(1,carbodytype(+car,#bodytype,+car,#bodytype)).
:- modeb(1,cartransmission(+car,#transmission,+car,#transmission)).
:- modeb(1,carenginesize(+car,#enginesize,+car,#enginesize)).

carfuel(A,X,B,Y):- hasfuelcons(A,X), car(A), car(B),
   hasfuelcons(B,Y), X\=Y .
carbodytype(A,X,B,Y):- hasbodytype(A,X), car(A), car(B),
   hasbodytype(B,Y), X\=Y .
cartransmission(A,X,B,Y):- hastransmission(A,X), car(A), car(B),
   hastransmission(B,Y), X\=Y .
carenginesize(A,X,B,Y):- hasenginesize(A,X), car(A), car(B),
   hasenginesize(B,Y), X\=Y .
```

Figure 4.9: Aleph hypothesis language in setting 2

```
bt(A,B) :-
   carfuel(B,nonhybrid,A,hybrid), cartransmission(B,manual,A,automatic).
bt(A,B) :-
   carfuel(B,nonhybrid,A,hybrid), carenginesize(B,medium,A,large).
```

Figure 4.10: A sample of Aleph valid hypotheses with setting 2

can also be made more concise and meaningful by adding specific background knowledge. For example, in an experiment with Aleph setting 2, we specified the specific hypothesis language in the background knowledge, as shown in Figure 4.9, to limit the rules to only consider comparisons between corresponding attributes. It also means that it was possible to skip unnecessary comparisons and save time.

4.5.1.2 Experiment settings using the sushi dataset

The experiment with the sushi dataset was performed on the same hardware as the car dataset. The settings and results of an experiment using the sushi preferences dataset are explained in this section.

SVM and DT. Similar to the settings explained for the car dataset, the learners were fed using 2 sets containing positive and negative examples. The positive examples were a set of correctly ordered pairs for each user. Then a set of negative examples was built from the opposite order of the user preferences. For each user, there was a complete set of 90 observations, consisting of 45 positive examples and 45 negative examples. The original dataset was already in numeric attribute format, so that in the sushi dataset discretisation was not necessary. A sample of numeric input for the SVM and DT containing positive and negative examples is shown in Table 4.8.

Table 4.8: Formatting sushi dataset for SVM and DT

1	1	0	0	7	6	1.26	2.73	1.98	2.14	2.70	1.84	0.88	0.84	1
1	1	0	0	7	5	1.26	2.69	1.98	2.04	2.70	1.52	0.88	0.92	1
1	1	0	0	7	8	1.26	0.81	1.98	1.64	2.70	3.29	0.88	0.88	1
1	1	0	0	6	7	2.73	1.26	2.14	1.98	1.84	2.70	0.84	0.88	-1
1	1	0	0	5	7	2.69	1.26	2.04	1.98	1.52	2.70	0.92	0.88	-1
1	1	0	0	8	7	0.81	1.26	1.64	1.98	3.29	2.70	0.88	0.88	-1

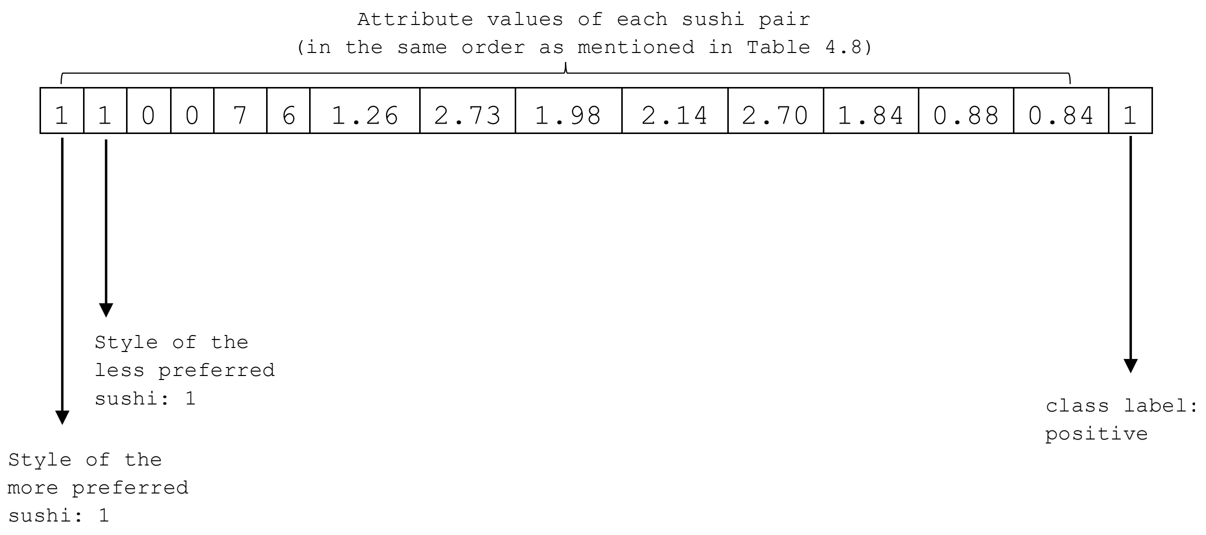
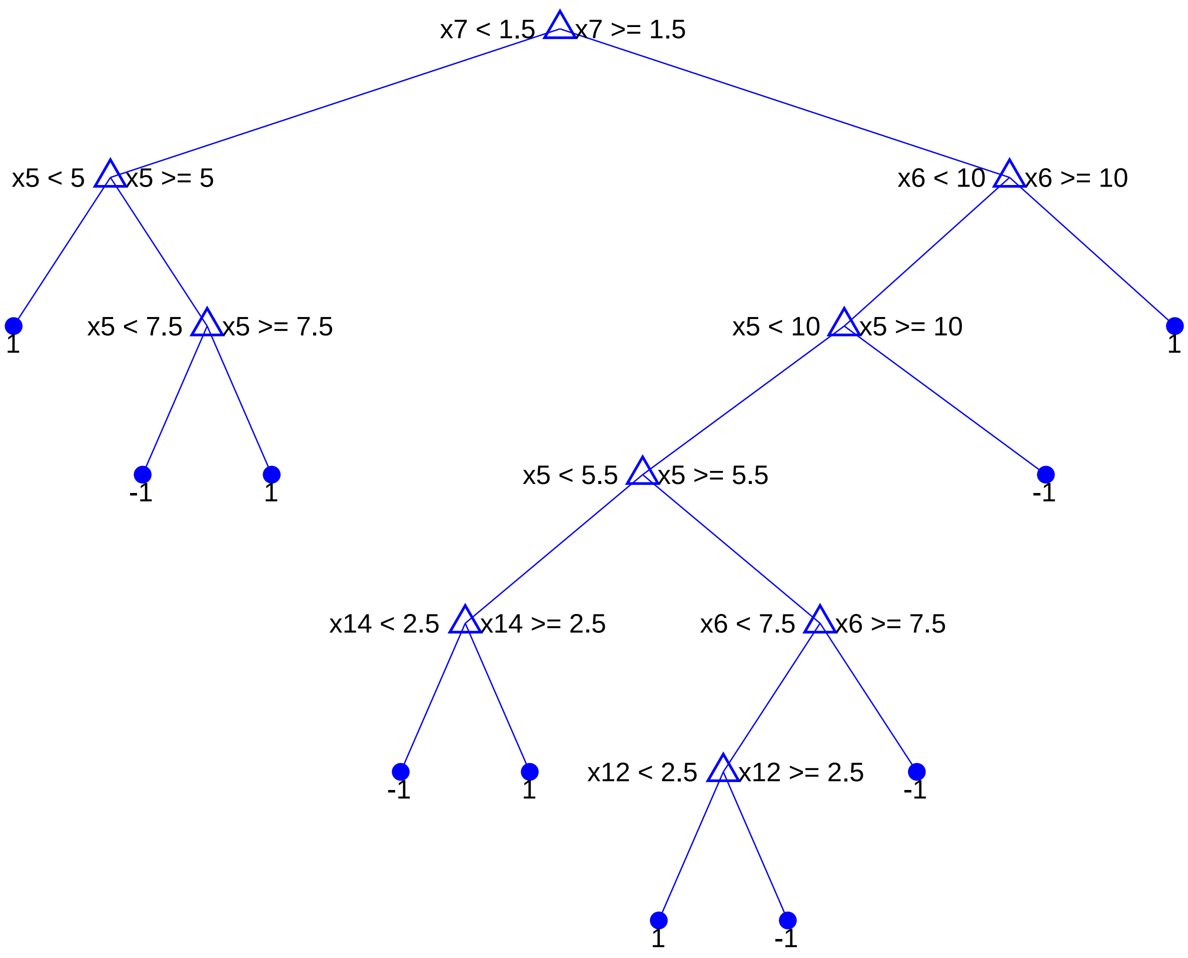

Attribute values of each sushi pair
(in the same order as mentioned in Table 4.8)
1 1 0 0 7 6 1.26 2.73 1.98 2.14 2.70 1.84 0.88 0.84 1
Style of the less preferred sushi: 1
Style of the more preferred sushi: 1
class label: positive

Figure 4.11: Sushi dataset node explanations

Each row represents a user preference on a pair of sushi types. The first 14 columns represent the numeric attributes and the last column is the class label (1=positive; -1=negative). The node explanation is described in Figure 4.11. The user ID is not required in the learning process as the individual unique preferences were learned each time. The same setting is repeated for each user in the dataset. The standard setting of Matlab CART DT implementation was followed and all attributes were treated as numeric. For the SVM implementation, Matlab used linear kernel function as the default setting. The DT used the rules to predict the new unseen data. In this experiment, some of the rules produced by the DT are shown in Figure 4.12. The node interpretation for the DT also follows Figure 4.11. For example, `x1` is the style of the more preferred sushi, `x2` is the style of the less preferred sushi, `x3` is the major group of the more preferred sushi and `x4` is the major group of the less preferred sushi.



(a) DT rule in graph mode

```
 1  if x7<1.5 then node 2 elseif x7>=1.5 then node 3 else -1
 2  if x6<4 then node 4 elseif x6>=4 then node 5 else 1
 3  if x6<10 then node 6 elseif x6>=10 then node 7 else -1
 4  if x6<2 then node 8 elseif x6>=2 then node 9 else -1
 5  class = 1
 6  if x14<2.5 then node 10 elseif x14>=2.5 then node 11 else -1
 7  class = 1
 8  if x5<5 then node 12 elseif x5>=5 then node 13 else 1
 9  class = -1
10  if x6<7.5 then node 14 elseif x6>=7.5 then node 15 else -1
11  class = 1
12  class = 1
13  class = -1
14  if x6<6.5 then node 16 elseif x6>=6.5 then node 17 else -1
15  class = -1
16  class = -1
17  class = 1
```

(b) DT rule in text mode

Figure 4.12: DT rule sample for sushi dataset

Aleph. All the settings in Aleph were the same as in the previous experiment (explained in Section 4.5.1.1). The sushi dataset formatting for Aleph positive and negative examples is shown in Figure 4.13. Three different Aleph settings for the hypothesis pattern were also used. Some of the background knowledge (incomplete) in setting 1 is shown in Figure 4.14a and in setting 2 in Figure 4.14b.

```
bt(sushi1,sushi0).
bt(sushi7,sushi2).
bt(sushi2,sushi5).
bt(sushi2,sushi9).
bt(sushi5,sushi3).
```

(a) File in '.f' extension containing positive examples

```
bt(sushi0,sushi1).
bt(sushi2,sushi7).
bt(sushi5,sushi2).
bt(sushi9,sushi2).
bt(sushi3,sushi5).
```

(b) File in '.n' extension containing negative examples

Figure 4.13: Aleph positive and negative examples for sushi dataset

Figure 4.14a means Aleph should consider hypotheses which include the attributes of each sushi in the pair. An example of a consistent hypothesis produced by this pattern is: 'in a pair, any `sushi A` is better than any `sushi B`, if `sushi A` has style of maki and `sushi B` has major group of `seafood`' (please see Figure 4.15). This is different from Figure 4.14b which means that Aleph is only allowed to consider hypotheses which compare the same attributes: "in a pair, any `sushi A` is better than any `sushi B`, if `sushi A` has style of (`maki` or `not maki`) and `sushi B` has style of (`maki` or `not maki`), in which the two sushis do not have the same style".

From the head mode (`modeh`), Aleph builds an hypothesis by looking for any examples that match this pattern `bt(+sushi,+sushi)` (see Figure 4.13); `bt()` means a predicate 'better than', while `+sushi` means it is an input of type '`sushi`'. The body modes (`modeb`) is a function of `sushistyle(+sushi,#style, +sushi, #style)` which has two types of input: '`sushi`' and '`style`'. How this function works is specified in the following line: `sushistyle(A,X,B,Y) :- hasstyle(A,X), sushi(A), sushi(B), hasstyle(B,Y), X\=Y`, which means that the search will consider building a candidate hypothesis if both items have different sushi styles.

A sample of the consistent hypotheses produced by Aleph is shown in Figure 4.15. The Aleph hypotheses above can be read as:

- any sushi A is better than any sushi B if sushi B has minor group: akami and sushi A has minor group: akami and sushi B has heaviness: 2 (i.e. between 2.2 and 3.7).

```
% specify the hypothesis pattern
:- modeh(1,bt(+sushi,+sushi)).
:- modeb(1,hasstyle(+sushi,#style)).
:- determination(bt/2,hasstyle/2).

% specify the different type of sushi
style(maki).
style(notmaki).

% specify the sushi name
sushi(sushi0).
sushi(sushi1).

% explain the relationship between sushi and attribute ‘style’
hasstyle(sushi0,notmaki).
hasstyle(sushi1,notmaki).
```

(a) Setting 1: single attributes

```
% specify the hypothesis pattern
:- modeh(1,bt(+sushi,+sushi)).
:- modeb(1,sushistyle(+sushi,#styleconsumption,+sushi,#styleconsumption)).
:- determination(bt/2,sushistyle/4).

% specify the different type of sushi
style(maki).
style(notmaki).

% specify the sushi name
sushi(sushi0).
sushi(sushi1).

% explain the relationship between sushi and attribute ‘style’
hasstyle(sushi0,notmaki).
hasstyle(sushi1,notmaki).

% do not include the attribute that has the same value
sushistyle(A,X,B,Y):- hasstyle(A,X),sushi(A),sushi(B),hasstyle(B,Y),X\=Y .
```

(b) Setting 2: comparable attributes

Figure 4.14: Aleph background knowledge for sushi dataset

- any sushi A is better than any sushi B if sushi B has minor group: roe and sushi A has minor group: akami.
- any sushi A is better than any sushi B if sushi A has major group: seafood and sushi B has minor group: squid.

```
    bt(A,B) :-
       hasminor(B,akami), hasminor(A,akami), hasheaviness(B,2).
    bt(A,B) :-
       hasminor(B,roe), hasminor(A,akami).
    bt(A,B) :-
       hasmajor(A,seafood), hasminor(B,squid).
```

Figure 4.15: A sample of Aleph consistent hypotheses for sushi dataset

4.5.1.3 Experiment result

The accuracy of the three existing algorithms is shown in Table 4.9. The experiment shows that in terms of the algorithm accuracy, the SVM outperformed Aleph and the DT in the car dataset but the DT shows the highest accuracy amongst the other algorithms on the sushi dataset. There is no single winner in this experiment. There is also no significant difference between the two different hypothesis language settings of Aleph in both datasets. The result of the ANOVA test shows that there is a significant difference in the accuracy of the algorithms, with under 5% significance level assumptions for both datasets, as shown in Table 4.10. An ANOVA is conceptually similar to a multiple two-sample t-test, but is more conservative (resulting in less type I errors).

Table 4.9: Mean and standard deviation of 10-fold cross validation test

	SVM	DT	Aleph setting 1	Aleph setting 2
car dataset	0.8317±0.12	0.7470±0.10	0.7292±0.08	0.7033±0.10
sushi dataset	0.7604±0.09	0.8094±0.06	0.7789±0.06	0.75±0.09

Table 4.10: ANOVA ($\alpha = 0.05$) statistical results on SVM, DT and Aleph

	F	p-value	F-crit
car dataset	16.8230	6.1592×10^{-10}	2.6429
sushi dataset	2.7211	0.0451	2.6429

4.5.2 Learning from mixed-type data

One of the advantages of using ILP systems is their ability to handle both numeric and categorical data without needing conversion. In this section, an experiment with Aleph is described in detail, including the hypothesis language, the background knowledge and the experiment results. Aleph with mixed type data settings was used in both datasets, car and sushi preferences.

4.5.2.1 Experiment settings with car dataset

In the car dataset, each item had four attributes, 3 of them were categorical attributes (body type, fuel and transmission) and 1 was numeric (engine size). Each user answered on the same set of items. In this experiment, the same background knowledge file was used for every user, but the set of positive and negative examples, which represented the user preference, was different. Figure 4.16 shows that the categorical and numerical attributes were treated differently. A complete background knowledge file, stored in a file with '.b' extension, is shown in Appendix C.1. The hypothesis in this setting considered comparisons of values from the same attributes. As an example: "car A is better than car B if car A is a hybrid car and car B is a non-hybrid car, and car A has a greater engine size than car B." This example is shown in Figure 4.18a.

```
% specify the hypothesis language
:- modeh(1,bt(+car,+car)).
:- modeb(1,carfuel(+car,#fuelconsumption,+car,#fuelconsumption)).
:- modeb(1,carbodytype(+car,#bodytype,+car,#bodytype)).
:- modeb(1,cartransmission(+car,#transmission,+car,#transmission)).
:- modeb(1,carenginegreaterthan(+car,+car,-float,-float)).
:- modeb(1,carenginelessthan(+car,+car,-float,-float)).

% specify different rules for categorical attributes
carfuel(A,X,B,Y):- hasfuelcons(A,X), car(A), car(B), hasfuelcons(B,Y), X\=Y.


% specify rules for numerical attribute
carenginegreaterthan(A,B,X,Y):- car(A), car(B),
   hasenginesize(A,X), hasenginesize(B,Y), X>Y.
carenginelessthan(A,B,X,Y):- car(A), car(B), hasenginesize(A,X),
   hasenginesize(B,Y), X<Y.
```

Figure 4.16: Aleph's background knowledge for car dataset in setting 3

4.5.2.2 Experiment settings with sushi dataset

In the original sushi dataset there were 7 attributes, three of them (i.e. style, major and minor) were categorical and the other four (i.e. heaviness/oiliness, frequency of consumption, normalised price and frequency of sushi sold) were numeric. In the sushi dataset, every user was asked to give a full ranking of the same set of sushis; therefore, the background knowledge used in the Aleph experiments is the same for all users. The hypothesis in this setting considers comparisons of values from the same attributes. Figure 4.17 shows that in the body mode declaration, the rules are set differently between the categorical and numerical attributes. The complete background knowledge for the sushi dataset is shown in Appendix C.2. An example of a valid hypothesis is: "sushi A is better than sushi B if sushi A is a seafood and sushi B is not a seafood, and sushi A has a greater level of heaviness/oiliness than sushi B." This sample is shown in Figure 4.18b.

```
% hypothesis language
:- modeh(1,bt(+sushi,+sushi)).
:- modeb(1,sushistyle(+sushi,#style,+sushi,#style)).
:- modeb(1,sushiheavinesslessthan(+sushi,+sushi,-float,-float)).
:- modeb(1,sushiheavinessgreaterthan(+sushi,+sushi,-float,-float)).


% specify different rules for categorical attributes
sushistyle(A,X,B,Y):- hasstyle(A,X),sushi(A),sushi(B),hasstyle(B,Y),X\=Y .


% specify rules for numerical attribute
sushiheavinesslessthan(A,B,X,Y):- sushi(A), sushi(B), hasheaviness(A,X),
  hasheaviness(B,Y), X<Y .
sushiheavinessgreaterthan(A,B,X,Y):- sushi(A), sushi(B), hasheaviness(A,X),
  hasheaviness(B,Y), X>Y .
```

Figure 4.17: Aleph's background knowledge for sushi dataset in setting 3

```
bt(A,B) :-
   carfuel(B,nonhybrid,A,hybrid), carenginegreaterthan(A,B,C,D).
```

(a) Aleph's hypothesis for car dataset

```
bt(A,B) :-
   sushimajor(B,seafood,A,notseafood), sushiheavinesslessthan(A,B,C,D).
```

(b) Aleph's hypothesis for sushi dataset

Figure 4.18: Aleph's sample solutions with setting 3

4.5.2.3 Experiment result

The results of the Aleph experiments with the mixed type setting are shown in Table 4.11. An ANOVA test was performed to evaluate whether there was a significant difference between the three different Aleph settings. The results of the ANOVA test, with a confidence level of 0.05 show that there is a significant difference on the sushi dataset, with p-value $= 1.81 \times 10^{-7}$, but not the car dataset.

Table 4.11: Aleph accuracy with mixed-type data setting

	Aleph setting 3
car dataset	0.7208±0.11
sushi dataset	0.7015±0.07

4.6 Summary

There are two main tasks in learning pairwise comparison preferences, learning the top class and learning to order the pairs. An experiment to learn the top class was carried out using two existing learning algorithms, BTM and DL-Learner. The second task was performed using three classification algorithms, SVM, DT and Aleph. A comparison between statistical and logic-based approaches on PL with pairwise comparisons was also highlighted, although some researchers argue that separating those two ML approaches is not always necessary.

In learning to order pairs of items, the results of the three different approaches are quite interesting. The statistical approach, SVM, works very well when all the data is in a numeric format and it can be very practical. On the other hand, Aleph and DT also showed good results, with some advantages of a more readable model (a set of rules) and they work well for both numerical and categorical data. In contrast to the DT, Aleph as the FOL learner algorithm, has a special feature of being more flexible in defining the rules.

It is shown that the full benefits of logic-based methods are to be expected for richer representations, where a range of background concepts (either provided by the software designers or gradually inferred from the data) can be used to model users with complex preferences. While in general, the statistical approach has proven to have practical advantages, the experiments show that logic-based approaches offer a number of benefits over those based on statistics:

- rule flexibility: the flexibility of adding background knowledge to the logical learner to limit the rules and build more meaningful result (e.g. using Aleph implementation) can be used.
- readability: the results of the logical approach (both the DT and Aleph) are more readable and easier to understand for further use (i.e. providing a recommendation).

The benefits of using the logic-based approaches motivated this thesis to use an ILP technique, which is based in FOL, to address the pairwise preferences problem. However, as observed, an FOL-based system is not easy to implement in a recommender system due to the complexity of the language representation. An effort to balance the trade-off between the benefits and the shortcoming of using a logic-based approach in recommender systems has been made. We propose an approach which applies a FOL-based system in a different logic representation, i.e. Description Logics. In the next chapter, the proposed approach is described in detail.

Chapter 5

Inductive Learning of Ordinal Data in Description Logics

Inductive Logic Programming (ILP) is very promising due to the advantages of using background knowledge to generalise consistent rules from given examples. Here we propose an algorithm named APARELL (Active PAirwise RELation Learner), which was built on the basis of ILP in Description Logics (DL). This chapter describes APARELL, a basic pairwise relation learning algorithm to solve strict order relations. Active Learning (AL) will be discussed further in Chapter 6. The algorithm is applied in the preference learning domain, as the automation of preference learning has become essential in many e-commerce applications, following the trend of customer personalisation. The use of DL representation in the e-commerce area, as the basic language of the semantic web, is also rising in popularity. We aim to learn preferences from a set of pairwise comparisons, where the user is given a pair of items and asked to explicitly choose the preferred one of each pair. The use of DL representation in this research is also motivated by a suggestion in [12] which states one benefit of using DLs over other logic representations is that they produce a more readable and human-friendly result for novice users. Furthermore, it gives an opportunity to integrate the proposed algorithm with e-commerce systems in the future. We evaluate the algorithm using two real world datasets in the sushi and car preference domains and show that the accuracy of APARELL outperforms the other 3 baseline algorithms, SVM, Decision Tree and Aleph.

This research builds on previous work combining ILP and DL, called the DL-Learner [70]. This aims to find the correct class description in a specific ontology from a set of positive and negative examples of individuals. While the DL-Learner can only learn class definitions, we aim to learn definitions of relations. In the previous chapter, we show how the DL-Learner learns the top class of preferences. However, it cannot be used to learn the order of preferences. We address this problem by using APARELL, which is described in the rest of this chapter.

5.1 Problem Representation

The preference learning problem is defined as follows:

> **Given:** a set of individuals, a class hierarchy, a mapping from individuals to classes, and a set of preferences represented as pairs of individuals, where the first individual is preferred over (strictly better than) the second,
>
> **Find:** a disjunction of axioms defining the domain and range of the relation `betterthan` (where each axiom is expressed as a conjunction of classes) that is complete and consistent with the given preferences.

This problem is categorised as a supervised learning problem, where the user assigns one of two possible labels for each pair of items after considering their attributes. We then use these labels to search for a definition of the Domain and Range class memberships that render the relation true.

We propose an approach to learn the relations and apply the resulting system to learn the strict order (i.e. better than). The properties of strict order relations are *anti-symmetric*, which means that if item A is better than item B, then in any case item B cannot be better than item A, unless A=B. That special case is excluded by the assumption of `betterthan` being anti-reflexive (i.e. X cannot be seen as better than itself). It is also *transitive*, which means whenever item A is better than item B, and item B is better than item C, then item A is better than item C.

In the preprocessing step, the data was completed using transitive closure. The ontology reasoner can always be used to extract all pairs of items satisfying the preference relation as a result of applying the transitivity property. We added these inferred examples to the set of positive examples, so that the complete closure is used by the learning algorithm. We also use the anti-symmetry property to produce all negative examples as a 'mirror image' of the positive ones (after completing their transitive closure), i.e., all pairs derived from a positive example through the swap of the first and second element in the pair. This is also consistent with how the ontology reasoner works. Providing anti-symmetry and anti-reflexive properties, it will infer that any pair of individuals with the reverse order of the relation is being inconsistent with the ontology. We also remove any cycle appearing in the relations.

The representation of a sample preference learning problem in DL is shown in Figure 5.1. The class hierarchy is given to the system as an input. We evaluated our algorithm using a simple class hierarchy, as shown in Figure 5.1a, in order to make the solutions comparable to the Aleph representation. In this section, we use Turtle[1] syntax to describe RDF data[2].

Turtle syntax provides a way to create an RDF triple in the form of groups (as subjects, predicates or objects) for ease of reading. For example, given the original RDF data with full URIs (Uniform Resource Identifiers) as below:

```
<http://www.mycars.org/ontology#car1>
  <http://www.mycars.org/ontology#betterthan>
  <http://www.mycars.org/ontology#car3>.

<http://www.mycars.org/ontology#car1>
  <http://www.mycars.org/ontology#betterthan>
  <http://www.mycars.org/ontology#car4>.
```

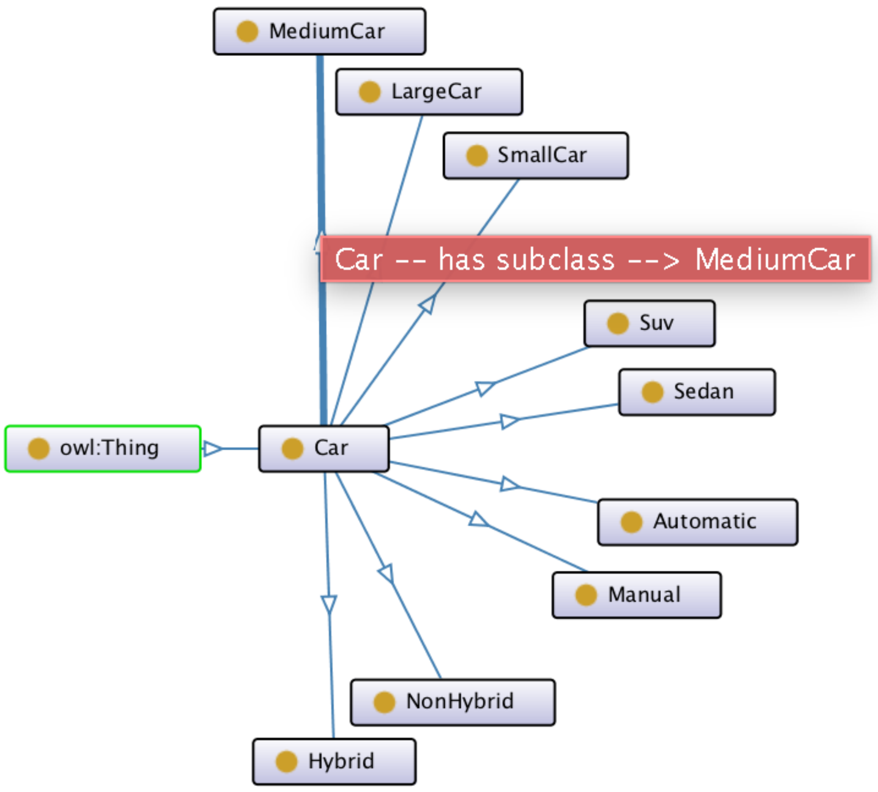
By factoring out common portions of URIs, Turtle syntax provides a shorter form as shown below:

```
@prefix myontology: <http://www.mycars.org/ontology#>.
myontology:car1 myontology:betterthan myontology:car3,
                                      myontology:car4.
```

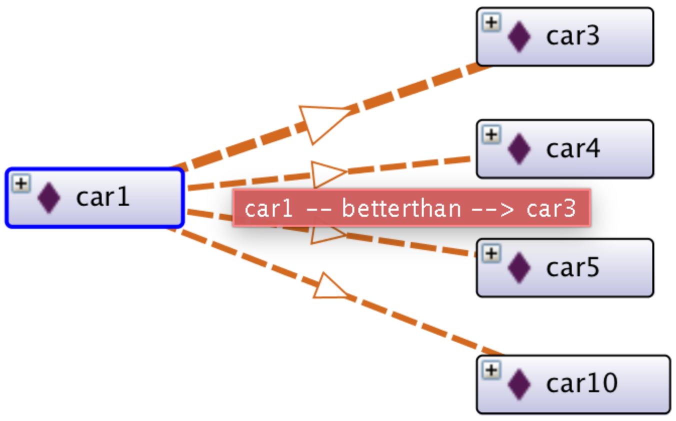
5.1.1 Hypothesis language

The aim of ILP is to find a theory that is complete (it covers all the given positive examples) and consistent (it covers no negative examples). In our algorithm, a hypothesis

[1]https://www.w3.org/TR/turtle/
[2]Turtle stands for Terse RDF Triple Language



(a) Simple car class hierarchy



(b) The user annotation is translated into object properties “betterthan”

Figure 5.1: Problem representation in the Protègè visualisation

is built for a specific object property that we want to learn, as in the case of the property `betterthan`. We describe `betterthan` as an object property in Turtle syntax as shown below:

```
myontology:betterthan rdf:type owl:ObjectProperty.
```

Each of the possible hypotheses about this relation is then described as a pair of class definitions, which specify membership of the domain $\mathcal{D}$ and the range $\mathcal{R}$ of the relation. The same hypothesis language can be described in Aleph notation as the following mode declarations:

```
:- modeh(1,betterthan(+car,+car)).
:- modeb(1,hasbodytype(+car,#bodytype)).
:- modeb(1,hasfuelcons(+car,#fuelconsumption)).
:- modeb(1,hasbodytype(+car,#bodytype)).
:- modeb(1,hastransmission(+car,#transmission)).
:- modeb(1,hasenginesize(+car,#enginesize)).
```

5.1.2 Background knowledge

Our algorithm represents background knowledge through classes and their membership, as shown in the example below (which uses Turtle syntax):

```
myontology:Sedan rdf:type owl:Class ;
               rdfs:subClassOf myontology:Car ;
               owl:disjointWith myontology:Suv .

myontology:car1 rdf:type owl:NamedIndividual ,
                        myontology:Car ,
                        myontology:Manual ,
                        myontology:NonHybrid ,
                        myontology:SmallCar ,
                        myontology:Suv .
```

The same background knowledge can be spelt out in Aleph as follows:

```
car(car1).         %type predicate
bodytype(sedan).   %type predicate
hasbodytype(car1,suv).         %bk
hasfuelcons(car1,nonhybrid).   %bk
hastransmission(car1,manual).  %bk
hasenginesize(car1, small).    %bk
```

5.1.3 Examples

In our algorithm, the set of positive examples is defined by user preferences and the preprocessing step which computes the transitive closure of these preferences, which are then negated to produce all negative examples. The following Turtle code expresses that car1 is better than car3, car4, car5, and car10:

```
myontology:car1 myontology:betterthan myontology:car3 ,
                                      myontology:car4 ,
                                      myontology:car5 ,
                                      myontology:car10 .
```

In traditional ILP syntax, the same examples are represented as ground facts of the predicate `betterthan/2`, where the arguments are of type `car`. So, the positive examples in Aleph are written as: `betterthan(car1,car3)`, and the negative, obtained by their reversal, as `:-betterthan(car3,car1)`.

5.2 Proposed Algorithm

The algorithm was implemented in Java using two Java-based API libraries: OWL API[3] [53] and RDF4J API[4] to handle DL. The user manual for an implementation of the proposed algorithm with a command line interface is provided in Appendix D. We follow the four basic procedures used in the Progol/Aleph greedy learning approach:

[3]http://owlapi.sourceforge.net/
[4]http://rdf4j.org/

1. **Select a positive example**. Each instance of the relation can be seen as a pair of object IDs. Similar to Aleph, we proceed through the example sequentially.

2. **Build the bottom clause**. The bottom clause is constructed from the conjunction of all non-disjoint classes in which each individual in the pairs is the member.

3. **Search**. This step is to find all clauses consistent with the data.

4. **Remove covered positive examples**. Our algorithm is greedy in its treatment of positive training examples. We removed all covered positive examples once consistent clauses were added to the current theory.

The pseudocode for APARELL is shown in Algorithm 1. It takes five arguments as an input: (i) background knowledge, (ii) a set of positive examples, (iii) a set of negative examples, (iv) a setting of literal (depth) limit, and (v) a relation name to be learned (in this case, it is *betterthan*). The algorithm will then process the input by following the four basic steps listed above.

After some initialisation, the loop is performed if positive examples are still available to be processed (some positive examples are removed during the search). The generalisation of each example starts from the size of 2 literals (*clause_size* is set to 2 in step 6), e.g. `Manual` *betterthan* `Sedan` (`Manual` is counted as 1 literal and `Sedan` is also counted as 1 literal). This generalisation is built on a combinations of each literal in the bottom clause. For example, in Figure 5.2, the bottom clause is given in the bottom box with the $\perp$ symbol. It contains a set of classes on the left, x_1={`Manual, NonHybrid, Smallcar, Suv`} and a set of classes on the right, x_2={`MediumCar, Manual, NonHybrid, Sedan`}. From the example, the generalisation will start from: `Manual` *betterthan* `MediumCar`. The algorithm will skip the evaluation if the literals on the left are the same as the literals on the right, e.g. `Car` betterthan `Car`. Any hypothesis/clause that is consistent with positive examples and which does not cover any negative examples will be stored in a theory T. An increment in the clause size (step 18) is needed whenever the algorithm cannot find a consistent clause in the current clause size (level). Otherwise, if one or more consistent clauses have been found in the current level, the algorithm will stop the search and remove the covered positive examples (step 25). If the algorithm cannot find any consistent clauses before it reaches the specified depth limit, the current example will be added to the theory T and it will be removed from the set of positive examples.

Algorithm 1 APARELL algorithm

```
    Input – background knowledge $B$,
            a set of positive examples $E^+ = \{\langle e_1, e_2\rangle, \ldots, \langle e_n, e_m\rangle\}$,
            a set of negative examples $E^- = \{\langle e_2, e_1\rangle, \ldots, \langle e_m, e_n\rangle\}$,
            literal (depth) limit $l$,
            a relation name $r = betterthan$
    Output – A theory $T$ represented as a set of clauses, each defining a pair of concepts
    specifying the relation's domain and range
 1: set $T = \emptyset$ /* initialisation*/
 2: set $x_1 = \emptyset$ /* the list of constraints on Domain */
 3: set $x_2 = \emptyset$ /* the list of constraints on Range */
 4: set $flag$ = false /* whether a consistent generalisastion of the bottom clause has been
    found */
 5: while $E^+$ is not empty do
 6:   set $clause\_size = 2$ /* initial value for $|x_1| + |x_2|$ */
 7:   set $x_1 = \emptyset$, $x_2 = \emptyset$, $flag = false$ /* reset in every loop*/
 8:   select $e^+ \in E^+$ to be generalised, and define $\langle e_1, e_2\rangle = e^+$
      /* build the bottom clause $\langle x_1, x_2\rangle$ for $e^+$ as follows: */
 9:   • set $x_1 = \{C_1, \ldots, C_m\}\forall C_i$ such that $e_1$ is a member of class $C_i$
10:   • set $x_2 = \{D_1, \ldots, D_n\}\forall D_i$ such that $e_2$ is a member of class $D_i$
11:   while $flag$==false and $clause\_size < l$ do
        /* search (top-down) through all generalisations of the bottom clause */
12:     for all $x'_1 \subseteq x_1, x'_2 \subseteq x_2, |x'_1| + |x'_2| == clause\_size$ do
13:       if $x'_1$ betterthan $x'_2$ is consistent and more general than $e^+$ then
14:         add the clause to $T$
15:         set $flag$ = true
16:       end if
17:     end for
18:     if $flag$==false and $clause\_size < l$ then
19:       set $clause\_size = clause\_size + 1$ /* increment clause size */
20:     else if $flag$==false and $clause\_size == l$ then
21:       add $e^+$ to $T$ and remove $e^+$ from $E^+$ /* exception */
22:     end if
23:   end while
24:   if flag==true then
25:     remove covered positive examples from $E^+$
26:   end if
27: end while
28: return $T$
```

5.2.1 Search and refinement operator

A top down approach similar to the one in Progol/Aleph was used, starting with the smallest clause length. The algorithm proceeds by considering an increasing number of properties (literals) constraining each of the two objects in the relation. The refinement operator starts from the top as written in DL below:

$$\top \sqcap \exists betterthan.\top$$

The bottom clause contains the conjunction of n constraints (of type class membership) on the Domain side and same number of constraints again on the Range side of the relation. This will produce $n \times n$ possible pairs on the first level of generalisation (clause length=2). All combinations of constraints are evaluated, apart from those that imply the same class membership of both arguments (i.e. *X is better than Y because they both share the same property/class membership)* and those that have already been considered. An example of the refinement operator from a positive example `car1` is better than `car3` is illustrated in Figure 5.2 (please see Section 5.5.1 for the car preferences original description).

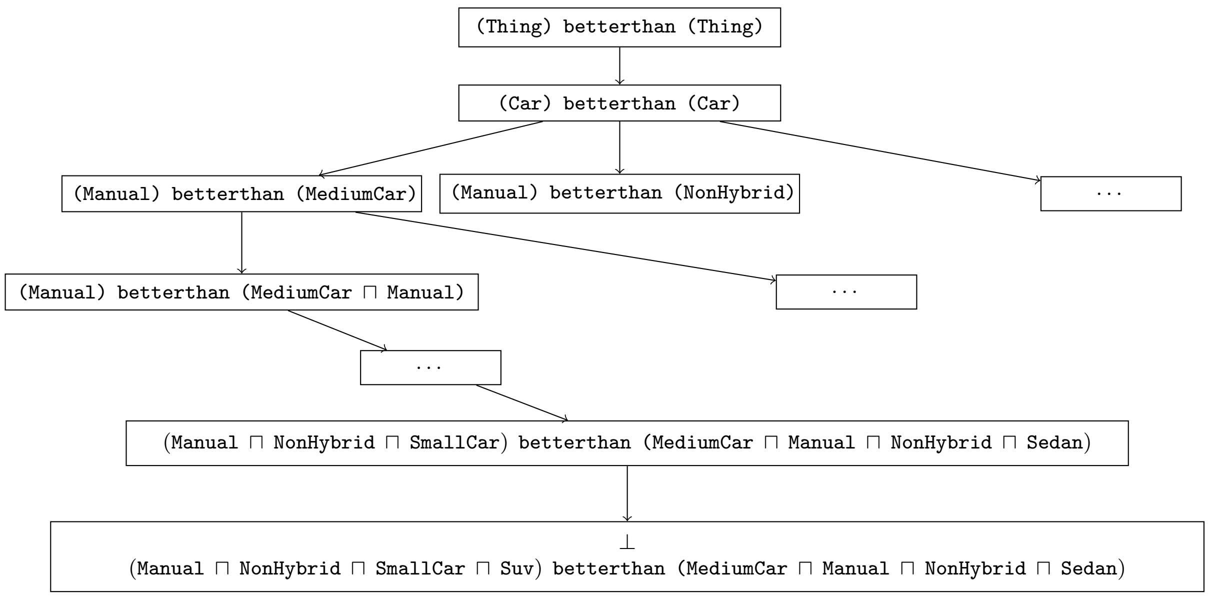

(Thing) betterthan (Thing)
(Car) betterthan (Car)
(Manual) betterthan (MediumCar)
(Manual) betterthan (NonHybrid)
...
(Manual) betterthan (MediumCar ⊓ Manual)
...
...
(Manual ⊓ NonHybrid ⊓ SmallCar) betterthan (MediumCar ⊓ Manual ⊓ NonHybrid ⊓ Sedan)
⊥
(Manual ⊓ NonHybrid ⊓ SmallCar ⊓ Suv) betterthan (MediumCar ⊓ Manual ⊓ NonHybrid ⊓ Sedan)

Figure 5.2: APARELL refinement operator

Comparisons with ALEPH. The coverage of a clause is expressed through P and N, where P is the number of positive examples covered and N is the number of negative examples covered. In cases where one solution has the same score as an alternative solution, Aleph will only return the first solution found. In our algorithm, we consider all new clauses that cover at least two positive examples and none of the negatives. (This is done to ensure consistency and that at least a minimum level of generalisation takes place.) The search will not stop until all possible combinations at each level have been considered. The resulting theory is a disjunction of clauses. Therefore, any test examples covered by one of them is classified as positive.

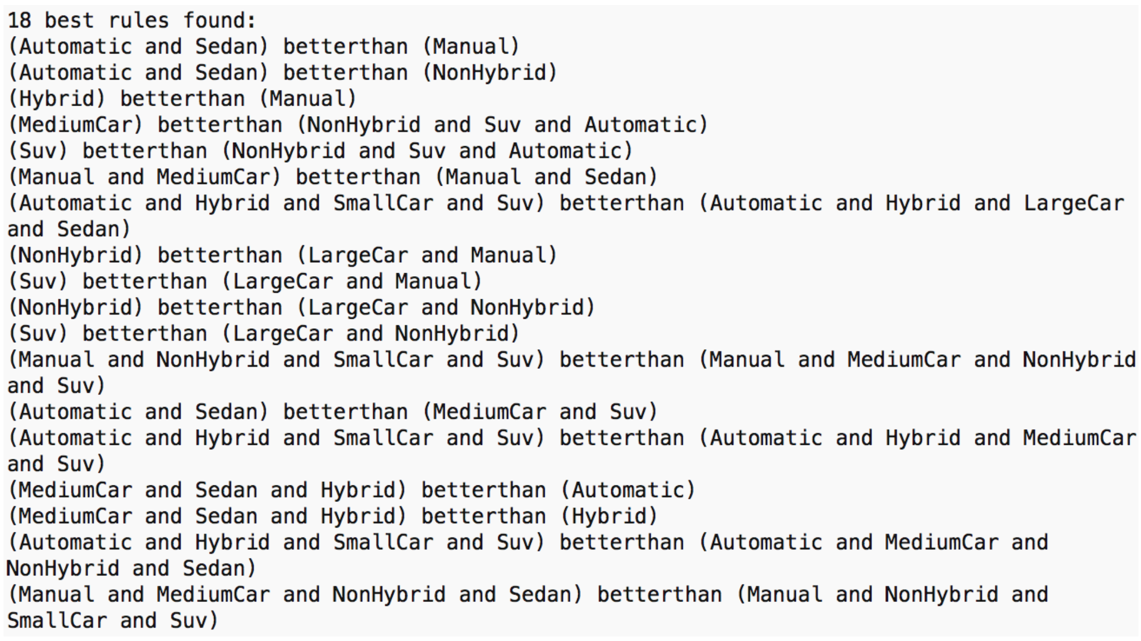
Our algorithm retains *all* consistent clauses generalised from a given positive example, rather than just one of them, as Aleph would have done. (At the same time, both algorithms discard the positive example from the training set after this generalisation step in a greedy learning manner.) This is the most important difference between the two algorithms and a possible explanation for any observed difference in their performance.

If a consistent clause has not been found yet, the algorithm continues to refine the current candidate by adding one literal to constrain either objects in the relation. Similarly to Aleph, when APARELL cannot find a consistent generalisation, it adds the bottom clause itself to the theory.

CWA and OWA. We implemented our algorithm using the Closed World Assumption (CWA). For the problem of learning strict order, it makes virtually no difference whether our system operates under the CWA or Open World Assumption (OWA). Under the OWA, we learn two hypotheses: one as mentioned before, the other with positive and negative examples swapped. For the given domain (of learning strict order), the second hypothesis (i.e. the one that is swapped between the negative and positive examples) is almost the exact negation of the first (modulo the choice of a training sample). The resulting coverage is therefore almost identical to that of the CWA hypothesis. The resulting hypothesis of learning the given positive and negative example from userID=1 in the car dataset is shown in Figure 5.3a, while the hypothesis with the swapped positive and negative example is shown in Figure 5.3b.

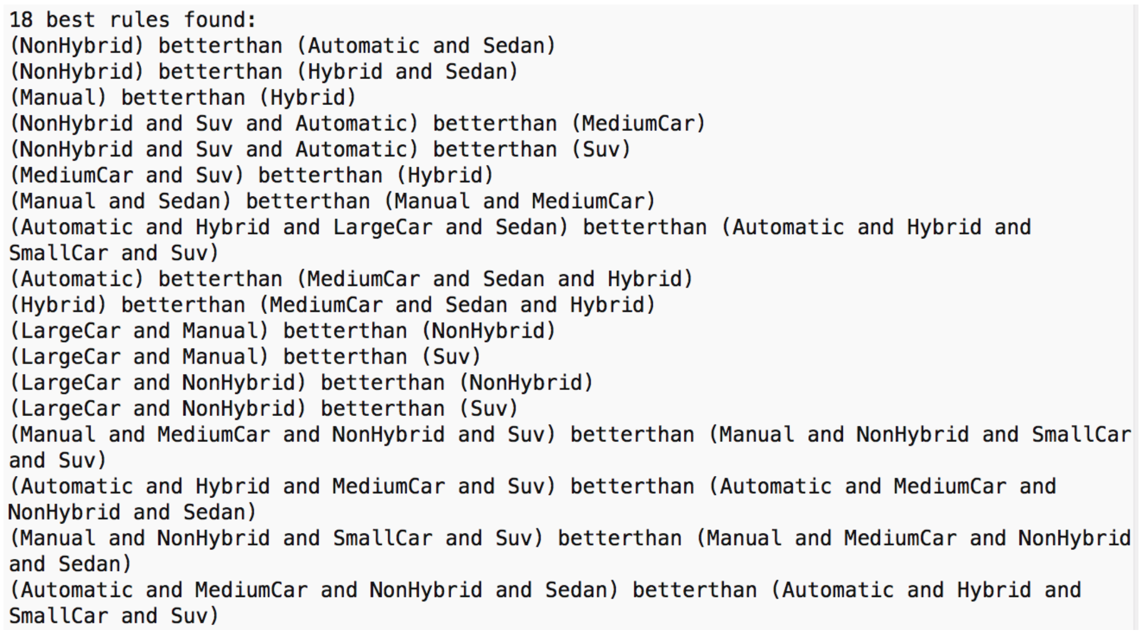
5.2.2 Complexity of APARELL algorithm

The algorithm uses breadth-first search with limited depth. For each positive example, the generalisation search tree will contain $2^n \times 2^n$ nodes in the worst case (if the depth is not limited), with n representing the number of attributes. This number comes from the combinations of clauses built from the attribute values in the bottom clause. If there are n attributes in the data set, then on the left 2^n possible attribute value combinations will



```
18 best rules found:
(Automatic and Sedan) betterthan (Manual)
(Automatic and Sedan) betterthan (NonHybrid)
(Hybrid) betterthan (Manual)
(MediumCar) betterthan (NonHybrid and Suv and Automatic)
(Suv) betterthan (NonHybrid and Suv and Automatic)
(Manual and MediumCar) betterthan (Manual and Sedan)
(Automatic and Hybrid and SmallCar and Suv) betterthan (Automatic and Hybrid and LargeCar
and Sedan)
(NonHybrid) betterthan (LargeCar and Manual)
(Suv) betterthan (LargeCar and Manual)
(NonHybrid) betterthan (LargeCar and NonHybrid)
(Suv) betterthan (LargeCar and NonHybrid)
(Manual and NonHybrid and SmallCar and Suv) betterthan (Manual and MediumCar and NonHybrid
and Suv)
(Automatic and Sedan) betterthan (MediumCar and Suv)
(Automatic and Hybrid and SmallCar and Suv) betterthan (Automatic and Hybrid and MediumCar
and Suv)
(MediumCar and Sedan and Hybrid) betterthan (Automatic)
(MediumCar and Sedan and Hybrid) betterthan (Hybrid)
(Automatic and Hybrid and SmallCar and Suv) betterthan (Automatic and MediumCar and
NonHybrid and Sedan)
(Manual and MediumCar and NonHybrid and Sedan) betterthan (Manual and NonHybrid and
SmallCar and Suv)
```

(a) Learning from positive and negative



```
18 best rules found:
(NonHybrid) betterthan (Automatic and Sedan)
(NonHybrid) betterthan (Hybrid and Sedan)
(Manual) betterthan (Hybrid)
(NonHybrid and Suv and Automatic) betterthan (MediumCar)
(NonHybrid and Suv and Automatic) betterthan (Suv)
(MediumCar and Suv) betterthan (Hybrid)
(Manual and Sedan) betterthan (Manual and MediumCar)
(Automatic and Hybrid and LargeCar and Sedan) betterthan (Automatic and Hybrid and
SmallCar and Suv)
(Automatic) betterthan (MediumCar and Sedan and Hybrid)
(Hybrid) betterthan (MediumCar and Sedan and Hybrid)
(LargeCar and Manual) betterthan (NonHybrid)
(LargeCar and Manual) betterthan (Suv)
(LargeCar and NonHybrid) betterthan (NonHybrid)
(LargeCar and NonHybrid) betterthan (Suv)
(Manual and MediumCar and NonHybrid and Suv) betterthan (Manual and NonHybrid and SmallCar
and Suv)
(Automatic and Hybrid and MediumCar and Suv) betterthan (Automatic and MediumCar and
NonHybrid and Sedan)
(Manual and NonHybrid and SmallCar and Suv) betterthan (Manual and MediumCar and NonHybrid
and Sedan)
(Automatic and MediumCar and NonHybrid and Sedan) betterthan (Automatic and Hybrid and
SmallCar and Suv)
```

(b) Learning from negative and positive being swapped

Figure 5.3: An example of APARELL hypothesis under CWA and OWA

be built, starting from size 1 up to size n. Each of these combinations needs to be paired with the attribute value combinations from the right side which are of the same size, i.e. 2^n, to create clauses/nodes in the search tree. Those two numbers are then multiplied, giving $2^n \times 2^n$.

In the worst case, if the search cannot find a consistent clause for the positive examples, then the search will be $m \times 2^n \times 2^n$, where m represents the number of positive examples. If the algorithm finds a consistent hypothesis at a certain level, it will stop and remove the covered positive examples to reduce the complexity. This greedy method has been proven very effective in the original ILP greedy algorithms such as Aleph and Progol in reducing the complexity while still producing good accuracy in a number of problems. Since the number of attributes is the most significant parameter of the algorithm complexity, it can be simplified to $\mathcal{O}(2^{2n})$ (where n is the number of attributes); therefore the complexity is exponential.

5.2.3 Current limitations

The algorithm can handle a multi-level class hierarchy. We only allow conjunctions of literals in the clauses, effectively limiting the language to $\mathcal{EL}$ description logic. With this limitation, we can still produce a fairly accurate model with the results being easier to interpreted. The most expensive process is membership checking (using the ontology reasoner) for all possible hypotheses. This is used for scoring the hypothesis coverage. One possible way to reduce the complexity is by minimising the search tree and checking the redundancy without reducing the accuracy. The search in our algorithm follows a breadth-first search with limited depth (specified as l parameter) which makes the complexity of the algorithm exponential with respect to the number of possible attribute values of each item, but it is capped by a certain depth. Another limitation of the proposed algorithm is that we do not provide a set of minimal solutions. Unlike Aleph, we produce a full set of solutions to make a better prediction.

While we produce a more accurate result, the process takes about 5 times longer than Aleph. For example, given 45 training examples, in which each item has 4 attributes, our algorithm takes 409 ms to finish while Aleph is faster, performing in just 88 ms. Although it is not a proper head-to-head comparison when the two systems were developed using different platforms.

5.3 Improving the Scalability

Our system can read two different input type options, an OWL file and an RDF database. In the first type, the system reads all inputs (i.e. the ontology file, the training examples and the test examples) from OWL text files and processes the query using the OWL API. Each example is stored as a pair separated by a comma (e.g. “car1, car2”). A set of examples is stored in a CSV file format and the positive and negative examples need to be stored in different text files. The difference between RDF and OWL can be read in Section 2.4.1.

In this section, the procedure for using the second input type, an RDF database is explained. A triplestore was used as the main input to read the ontology model. The GraphDB[5] triplestore was implemented, and all queries processed remotely using the SPARQL query language. Triplestore is used in our system to accommodate a larger dataset, which cannot be handled in terms of the performance speed, by a text file.

The steps used in applying ILP with the RDF database are explained below:

1. Build a connection to a triplestore database server.

2. Build a membership table to speed up the scoring process. In this step, we execute a SPARQL query to retrieve the class memberships. We need to store in memory which individual belongs to a certain class. This makes the learner work faster when it comes to counting the hypothesis coverage scores. By using a membership table, it is not necessary to run the query every time, saving resources.

3. Select a positive example to be generalised.

4. Build the bottom clause. In this process, similar to step 2, we use a SPARQL query to retrieve all classes of the individual on the left and right sides of the relation.

5. Search the solutions in the same way as explained in the previous section, to build the refinement operator.

6. Remove the covered examples and repeat the generalisation procedure until all examples are covered by rules.

[5]http://graphdb.ontotext.com/

Step numbers 3 - 6 are the main steps in the proposed algorithm (see Section 5.2). This procedure was evaluated using the existing datasets [1,65] and the results show that using triplestore can make the process 4 times faster than using textfiles.

5.4 Search in a Complex Class Hierarchy

In the case of a more complex class hierarchy that consists of more than one level, we consider the hypothesis search over all inferred class memberships of each individual. For example, if the class hierarchy and its membership in Figure 5.4 is given to the system, the algorithm progresses as follows:

1. **Select a positive example.** As an example, car 1 is better than car 3, is generalised. Please see Table 5.2 for the properties of the cars.

2. **Build the bottom clause**. In this step, the reasoner is used to infer all classes which include car 1 and car 3 in their direct and indirect membership. The bottom class of the above example is shown as follows:

   ```
   (Car and EconomyCar and FamilyCar and Manual and
             NonHybrid and SmallCar and Suv) betterthan
             (Car and Manual and MediumCar and NonHybrid and Sedan)
   ```

   The above bottom clause is only used for generalisation guidance, since `SmallCar` $\sqsubseteq$ `EconomyCar` $\sqsubseteq$ `Car`. The examination of a clause like this is skipped because the membership of the intersection between a class and its superclass will be the same as the membership of the class itself. For example, the membership of `SmallCar` $\sqcap$ `EconomyCar` is {`car1,car8`}, which is the same as the membership of `SmallCar` itself.

3. **Search**. The search begins with the shortest clause length considered (2 literals) as shown below:

   ```
   Car betterthan Manual
   Car betterthan MediumCar
   Car betterthan NonHybrid
   Car betterthan Sedan
   ```

```
EconomyCar betterthan Car

EconomyCar betterthan Manual

...
```

If no consistent hypothesis of length 2 is found, the search parameter is increased to 3:

```
EconomyCar and FamilyCar betterthan Car

EconomyCar and FamilyCar betterthan Manual

...
```

Please note that we also skip evaluations of same value comparisons like: `Car betterthan Car` to speed up the search.

4. **Remove covered positive examples**. We count the coverage for each hypothesis built and remove any covered positive examples from the training dataset.

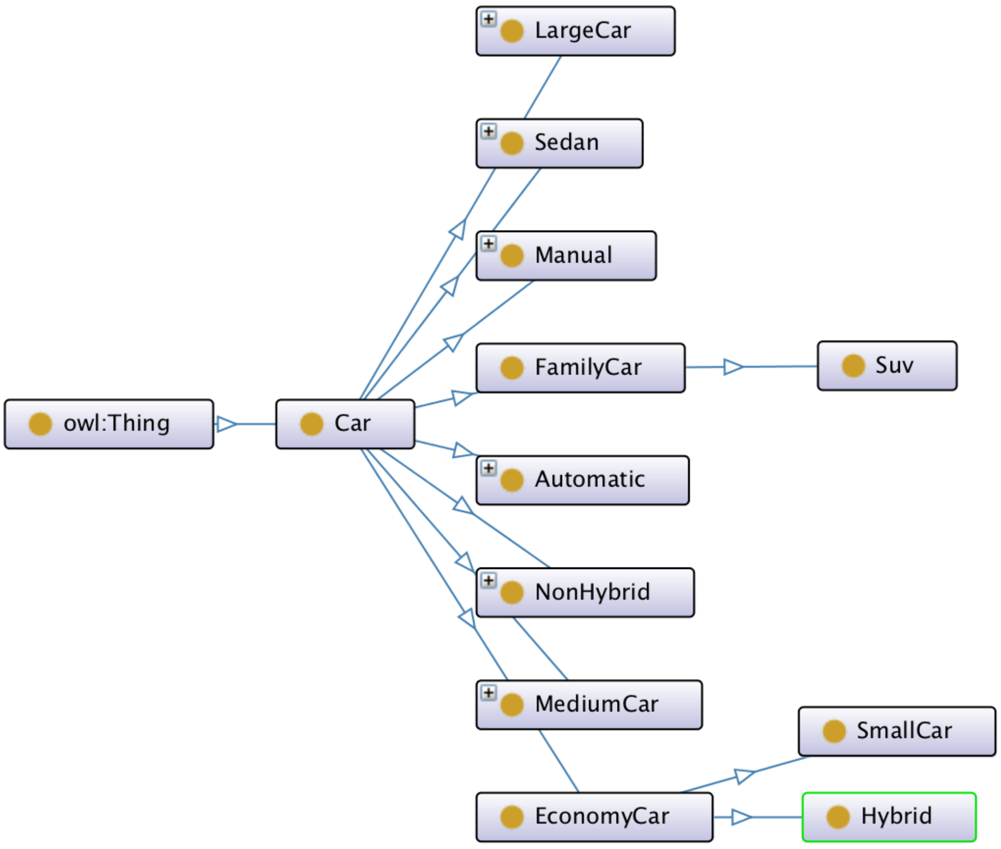


Figure 5.4: An example of more than one level of car class hierarchy

5.5 Evaluation

5.5.1 Dataset

In this experiment, we use the same datasets as in Chapter 4. Both the sushi and the car datasets have 10 items to rank, giving 45 preference pairs per user. We took 60 users from each dataset and performed 10-fold cross validation for each individual user's preferences. The car dataset has 4 attributes, body type, transmission, fuel consumption and engine size. We provide a sample of data from [1] to make this clearer as shown in Table 5.1 and Table 5.2. Despite the difference in the number of attributes in the two datasets, we found that with the clause length was set to 4 (in Aleph and in our algorithm), it was sufficient to produce consistent hypotheses.

The second dataset used in the experiments is about sushi preferences[6], published by Kamishima [65]. The dataset contains individual user preferences for 10 different sushi types. The users were asked to sort the sushis according to their preferences in ascending order. For the experiment, we generated the pairs order from each individual preference. Similar to the car dataset explained above, 45 pairs of sushi preferences were built from 10 types of sushi. The specification of sushi is presented in Table 4.3. In the dataset, 5,000 users were involved in the survey, but in this experiment, only data from the first 60 users were used, to make it comparable with the experiment on the car dataset. The sushi preference dataset is quite large compared to the car preference dataset, as it has 7 attributes giving the specification of each sushi . With the large size of the sushi dataset, we were able to test the performance of the algorithm and evaluate how it grows. With the literal setting=2, 7 attributes on each pair creates 49 (7×7) possible combinations of values to be visited as candidate hypotheses. The combination could be even larger with a higher setting of maximum literals allowed in the algorithm. For example, with depth setting=4, for each user there will be 1,274 nodes × 45 (number of examples) to visit. Please note that comparisons of the same values are skipped and some examples are removed during the search. The sushi class hierarchy used in this experiment is shown in Figure 5.5.

[6]http://www.kamishima.net/sushi/

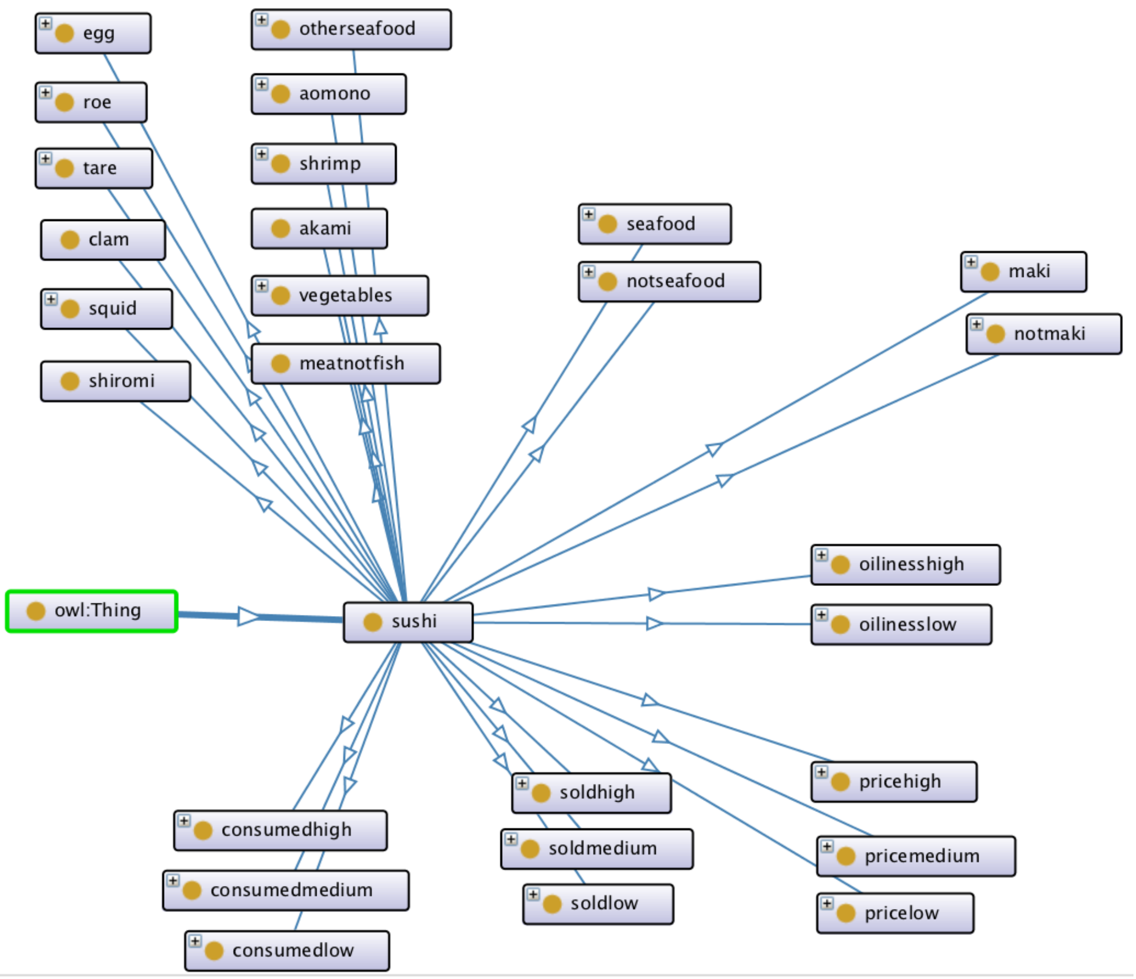


Figure 5.5: Sushi class hierachy

Table 5.1: Item descriptions on car dataset

Car ID	Bodytype	Transmission	Fuel	Engine size
1	suv	manual	non-hybrid	small
2	sedan	automatic	hybrid	large
3	sedan	manual	non-hybrid	medium
4	sedan	manual	non-hybrid	large
5	suv	manual	non-hybrid	medium
6	suv	automatic	hybrid	medium
7	sedan	automatic	hybrid	medium
8	suv	automatic	hybrid	small
9	sedan	automatic	non-hybrid	medium
10	suv	automatic	non-hybrid	medium

Table 5.2: A sample of car preference dataset

User ID	Item1 ID	Item2 ID
1	1	3
1	1	4
1	1	5
1	1	10

5.5.2 Evaluation method

The goal of this evaluation is to assess the accuracy of the predictive power of the algorithm to solve the preference learning problem. To achieve this, we set up four experiments:

1. Asses the accuracy of the algorithm compared to three baseline algorithms, SVM, Aleph and DT, on 60 users in the car dataset and the sushi dataset. An ANOVA and a post-hoc test were done to assess if any of the differences in the algorithms' performance were significant.

2. Asses the capability of the algorithm to learn relations from a more complex class hierarchy by using the car dataset.

3. Asses the accuracy and the performance of the algorithm on a larger dataset by conducting an experiment with the 5,000 users of the sushi dataset.

4. Asses the accuracy of the algorithm on different training example sizes compared to the three baseline algorithms.

Accuracy on 60 users. We compared our algorithm with three other machine learning algorithms: SVM, the Matlab CART Decision Tree (DT) learner and Aleph. SVM is a very common statistical classification algorithm that is used in many domains. Similar work on pairwise preference learning was performed by Qian et al. [103] showing that SVM can also be used to learn in this domain. Both DT and Aleph were included in the evaluation since both are logic based learners, with the first in propositional logic and the second in First Order Logic.

We built a simple class hierarchy as explained in Section 5.1 for each dataset. We learned the individual preferences and evaluated the model using 10-fold cross validation. We repeated the same test for all users then found the average accuracy. The accuracy result is shown in Table 5.3. This experiment stopped at a length of 4 literals (the same as Aleph's default clause length). According to the ANOVA test with $\alpha = 0.05$, the results show that there is a significant difference between the algorithms, with a p-value of 1.14×10^{-21} for the car dataset and 2.97×10^{-3} for the sushi dataset.

An ANOVA is conceptually similar to multiple two-sample t-tests but is more conservative (resulting in less type I errors). After performing an ANOVA test, we needed to find which algorithms were significantly different using Fisher's Least Significant Difference (LSD). The results of this post-hoc test are shown in Table 5.4. Please note that the results of the 10-fold cross validation may have a bias due to the fact that the use of transitivity chains (closures) may create an overlap between training and test data.

Table 5.3: Mean and standard deviation of 10-fold cross validation test

	SVM	DT	Aleph	Our algorithm
car dataset	0.8264±0.12	0.7470±0.10	0.7292±0.08	**0.8456**±0.06
sushi dataset	0.7604±0.09	0.7869±0.07	0.7789±0.06	**0.8138**±0.07

Table 5.4: Post-hoc Fisher's Least Significant Difference (LSD)

Algorithm 1	Algorithm 2	Sushi dataset		Car dataset	
		p-value	means diff.	*p*-value	means diff.
Aleph	DT	0.561	same	0.292	same
Aleph	Our algorithm	0.011	different	0	different
Aleph	SVM	0.177	same	0	different
Our algorithm	DT	0.049	different	0	different
Our algorithm	SVM	0	different	0.257	same
DT	SVM	0.054	same	0	different

Accuracy on more complex class hierarchy. As an additional experiment, we evaluated the car dataset with the complex class hierarchy (see Figure 5.4) using literal depth limit=4. The accuracy of our algorithm was performed using 10-fold cross validation. The results of the experiment with the complex class hierarchy are shown in Table 5.5. The accuracy results show no difference between using simple class hierarchy and complex class hierarchy.

Table 5.5: Car dataset with complex class car hierarchy

mean	0.8409
std.deviation	0.1405

Accuracy on a larger dataset. In this experiment, our algorithm still showed the highest accuracy compared to the three baseline algorithms. The average accuracy on individual preferences from 5,000 sushi dataset users is shown in Table 5.6. In the first experiment, evaluated the mean difference between the algorithms using an ANOVA and a post-hoc test. Here, the p-value of the ANOVA (α=0.05) test was 0. This is common in a large dataset. According to [75], performing a statistical test to analyse the mean difference in a large dataset can be problematical, as p-values tend to drop quickly to zero. From the results shown in Table 5.6, a low standard deviation rate is a good indication that the accuracy of our algorithm is excellent in any case in the dataset.

Table 5.6: Experiment result on 5000 users

	SVM	DT	Aleph	Our algorithm
mean	0.7599	0.8004	0.7867	0.8150
stdev	0.0912	0.0612	0.0697	0.0604

Accuracy on different training example sizes. We performed several experiments with the algorithms by varying the proportion of training examples and testing it on 10% of examples. For a more robust result, we validated each cycle with 10-fold cross validation. The results of these experiments are shown in Figure 5.6, where it can be seen that APARELL still works better, even with the smallest number of training examples.

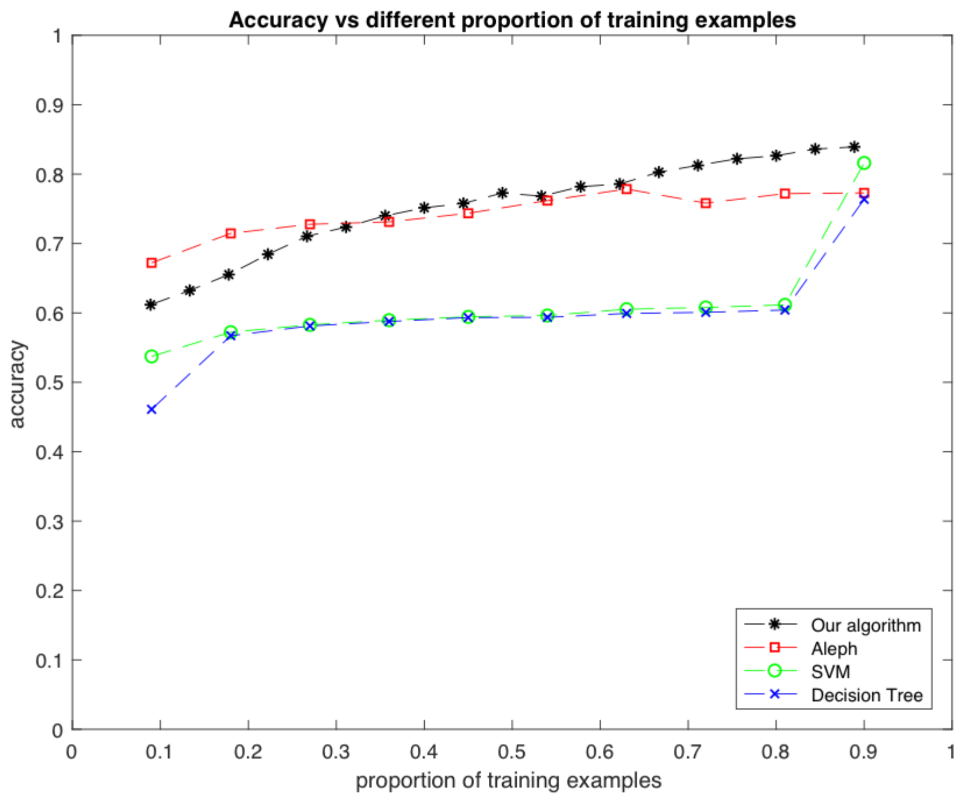


Figure 5.6: Accuracy by varying number of training examples

5.5.3 Algorithm performance

A further test was run to examine algorithm performance by different clause length settings (see Table 5.7). The evaluation was performed on both datasets by training the model on 90% of the examples and testing it on the remaining 10%. We also recorded the algorithm execution times of 60 users × 90% of examples (2,382 total examples in the car dataset and

2,400 total examples in the sushi dataset, please note that some of the positive examples are removed during the search). The algorithm was executed on Java 8 Eclipse IDE with 8 GB Memory 1867 MHz DDR3 and a 2.9 GHz Processor Intel Core i5 machine.

The results show there is no significant accuracy improvement after 4 literals. Surprisingly, the algorithm ran very slowly at 2 literals for the car dataset (where the achieved accuracy is the lowest for all tested clause lengths). The reason is in the removal of the covered positive examples (see Section 5.2). If we cannot find a consistent hypothesis when generalising a positive example, we add an exception and only remove one example. However, when the algorithm finds consistent hypotheses, it removes more than one example, which results in much fewer positive examples, thus speeding up the search. This anomaly does not occur in the sushi dataset due to the larger number of attributes, so that the possibility of finding a consistent hypothesis for clause length 2 is higher.

Table 5.7: Performance on different clause length settings

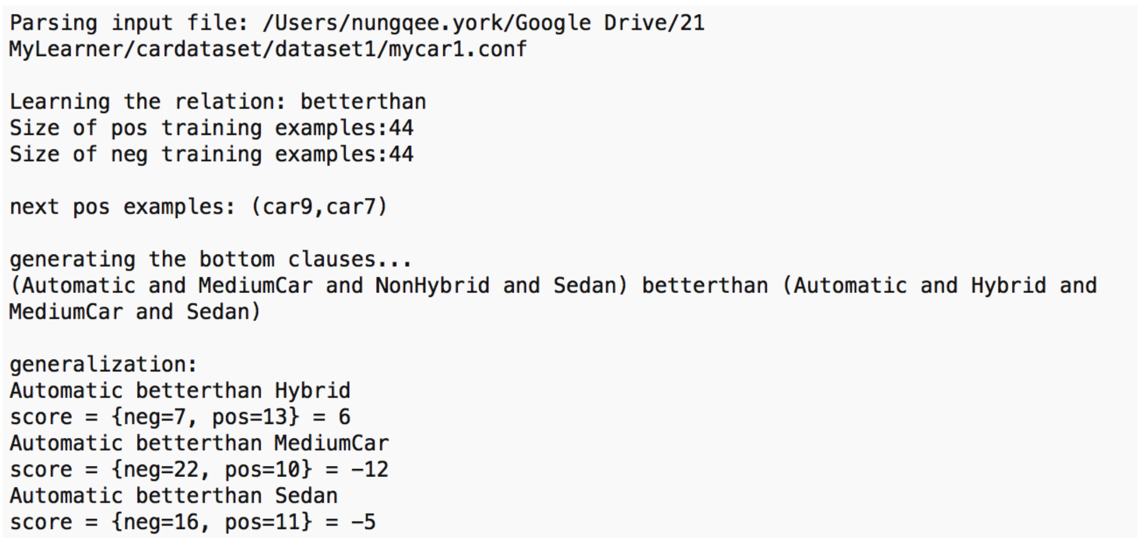
Clause length	Car dataset		Sushi dataset	
	Accuracy	Running time	Accuracy	Running time
2 literals	0.7433 ±0.15	17,824 ms	0.8135±0.13	12,189 ms
3 literals	0.8533 ± 0.14	11,338 ms	0.8117±0.13	19,040 ms
4 literals	0.8434 ±0.12	13,050 ms	0.8266±0.13	36,922 ms
5 literals	0.8217 ±0.15	14,223 ms	0.8150± 0.14	60,943 ms

5.6 Sample Output

An example of consistent hypotheses found by our algorithm is shown in Figure 5.3. The output of the command line software also prints out all possible hypotheses in the search space and their coverage score, as shown in Figure 5.7.

5.7 Summary

In this chapter, the ILP implementation in DL was demonstrated, to study relation in general and in particular to learn user preferences from pairwise comparisons. In terms of accuracy, the experiments show that our algorithm outperformed the other baseline algorithms, but this is a time consuming process. In order to produce a complete and consistent hypothesis, our algorithm takes much longer than the three baseline algorithms. In fact, the proposed algorithm has proven statistically significantly better than all tested alternatives in all but one case. The exception in question is when compared to SVM on



```
Parsing input file: /Users/nungqee.york/Google Drive/21
MyLearner/cardataset/dataset1/mycar1.conf

Learning the relation: betterthan
Size of pos training examples:44
Size of neg training examples:44

next pos examples: (car9,car7)

generating the bottom clauses...
(Automatic and MediumCar and NonHybrid and Sedan) betterthan (Automatic and Hybrid and
MediumCar and Sedan)

generalization:
Automatic betterthan Hybrid
score = {neg=7, pos=13} = 6
Automatic betterthan MediumCar
score = {neg=22, pos=10} = -12
Automatic betterthan Sedan
score = {neg=16, pos=11} = -5
```

Figure 5.7: A sample of APARELL generalisation

the car dataset, where our algorithm achieves a seemingly higher mean accuracy, but the result is not statistically significant (in other words, it is a draw).

As shown in the previous chapter, it could be argued that the original ILP system, which is based on logic programming, can be used to solve the problem in preference learning as well as propositional logic and statistics-based systems. It is also true that as a fragment of FOL, representing the problem in DL make it more restrictive in terms of the expressiveness. However, in this chapter, we have shown that the knowledge base in the preference learning problem can be represented by using DL representation in a more human-friendly and more readable way than other representations, as suggested in [12]. In addition, research in DL has been paying attention to scalable triplestore database management systems, which can accommodate data growth in the future. This might also be advantageous for our research. In the next chapter, the active learning part of APARELL, improving the accuracy with the least training data, is described.

Chapter 6

Active Learning to Support the Inference Process

In the previous chapter, the basic module of a relation learner based on ILP in DL, APARELL, is explained. While the use of ILP in DL can be very useful in learning preferences in pairwise comparisons, another challenge remains on how to select the most informative pairs when asking users. In the real world, the problem of selecting the next pairwise comparisons to be annotated can be more complicated. There will be thousands of possible pair combinations with different attribute descriptions.

Active Learning (AL) aims to reduce the number of examples as early as possible. The quality of an AL strategy really depends on which learning algorithm is used and the problem itself [117]. It is not possible to generalise the quality of an AL strategy to all learning algorithms. There are a number of sampling methods in AL which are commonly used to address certain classification problems, such as uncertainty sampling, error/variance reduction and query by committee. Most of these well-known AL strategies use probabilities to measure the likelihood of each data point being classified as a certain class. However, they cannot be used to support our learning algorithm, because probability has not yet been implemented in APARELL. An explanation of existing AL strategies is provided in Chapter 2.

Here we explain the active learning strategy to enhance the basic module of APARELL. A new active learning strategy to choose the most informative pairs using the distance measurement between items in pairs is proposed. The proposed AL strategy has been evaluated on the car preferences dataset and compared to two other sampling methods: (i) random sampling, and (ii) maximum distance sampling.

6.1 Problem Definition

In the domain of pairwise preference learning, the active learning problem is defined as follows:

Given: a set of unlabelled data in pairs

Find: the most informative data in pairs to be fed into the learning algorithm to produce a more accurate result with fewer examples.

The term *informative* here means the capability to provide a significant contribution to the predictive power of the chosen learning algorithm.

6.2 Proposed AL Strategy

A new approach was developed based on the idea of version space [86] reduction, where the next point to be labelled is the one that reduces the size of the version space faster. It is useful for our learning algorithm to explore every possibility as wide as possible at the beginning. We can then proceed with the search in each local version space in order to give the learner more supporting facts. An illustration of the search process is shown in Figure 6.1. For example, a user is given two pairs of cars to chose from, as below:

- car 1 **OR** car 2
- car 3 **OR** car 4,

where each car has the attributes:

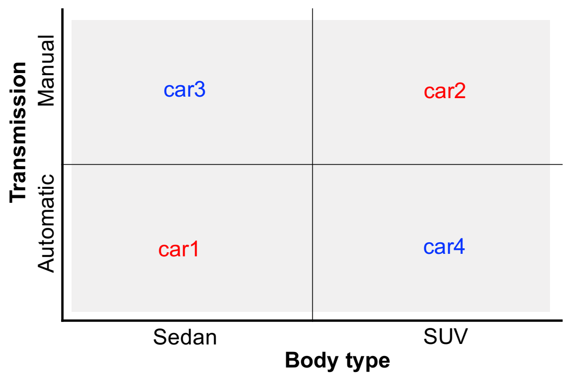
- car 1 is a sedan and automatic car
- car 2 is an SUV and manual car
- car 3 is a sedan and manual car
- car 4 is an SUV and automatic car,

and the user's answers are:

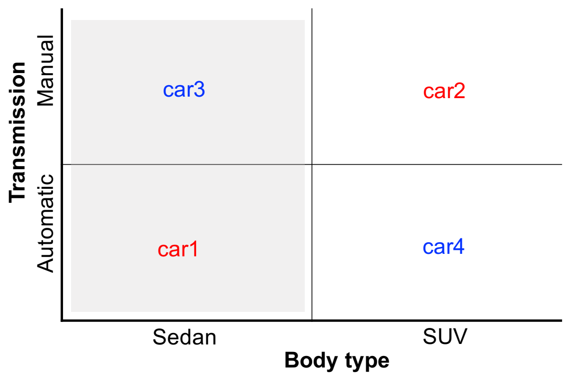
- car 1 is better than car 2
- car 3 is better than car 4.

From the above example, the data is processed in two steps. In the first step, each diagonal space in four quadrants is explored (see the grey shaded area in Figure 6.1a). This is called the ‘exploring step’. The user is given a set of pairs considering the most different attributes (later called *the furthest distance*) of each item in the pairs, e.g. the pair car 1 and car 2 was selected because the cars do not share any common attributes. The aim of this step is to decide which areas need to be explored further. A method of measuring the distance between two items is needed to select which pair of item has the furthest distance, which in this case means the most informative pair. The measurement method used is explained in Section 6.2.1.

In the second step, the search focuses on half of the quadrants (see the grey shaded area in Figure 6.1b). In this step, the search is focused on more specific attributes preferred by the user, which leads the learner to higher accuracy. This is called the ‘refining step’. More details on each step are explained in Section 6.2.2.



(a) The exploring step



(b) The refining step

Figure 6.1: Illustration of the selection of the next pairwise process

6.2.1 Pair distance measurement

The idea of using a distance measure to separate the data as far as possible is similar to performing pre-clustering on the data. However, performing data clustering on a large dataset is very time consuming and cannot be performed simultaneously with the learning process. Pre-clustering can also be problematic when very limited data is available. The intuition behind the use of this method is that performing a distance measurement allows the learner to get better sampling data from different attribute values. The pair distance is measured to obtain the most informative data points from different regions in the attribute dimension space.

The ALGO_DISTANCE algorithm, proposed by Ahmad and Dey [7] was used to measure the distance for *single valued variable* categorical data. This means, if $(X_1, X_2, \ldots, X_m)$ define a data object having m attributes then every attribute value $X_i (i = 1, \ldots, m)$ can take only one value. The ALGO_DISTANCE has been evaluated in supervised and unsupervised learning methods. It is based on the fact that the similarity between two attribute values depends on their relationship with the other attributes. If the distance between two data points is large, they most likely do not share any common attribute values.

ALGO_DISTANCE measures the distance between every pair of attribute values in the same attribute category then repeats this for all attributes in the data set. For example, in the car dataset, it measures the distance between the attribute value `Manual` and the attribute value `Automatic` in the `Transmission` category. The dissimilarity between two attribute values in the same category is computed with respect to the co-occurrence with every other attribute value in each different category of the data set. The average value of the co-occurrence distances will then give the distance between two distinct attribute values in that data set.

Suppose there is a categorical attribute A_i in a data set, with two values x and y. For the given data set, suppose A_j denotes another categorical attribute. Let w denote a subset of values of A_j. Using set-theoretic notation, $(\sim w)$ denotes the complementary set of values occurring for attribute A_j. The co-occurence of each attribute value with other attribute values can be calculated using conditional probabilities. Let $P_i(w|x)$ denote the probability that an element having value x for A_i, has a value belonging to w for A_j. The distance between attribute values x and y of A_i with respect to attribute A_j is denoted by $\delta^{ij}(x, y)$ and is defined as follows:

$$\delta^{ij}(x, y) = P_i(\omega|x) + P_i(\sim \omega|y) - 1, \tag{6.1}$$

where ω is the subset w of attribute A_i values that maximise the sum of $P_i(\omega|x)$ and $P_i(\sim \omega|y)$. Since both $P_i(\omega|x)$ and $P_i(\sim \omega|y)$ lie between 0 and 1.0, to restrict the value of $\delta^{ij}(x, y)$ to between 0 and 1, the sum needs to be subtracted by 1. This can be achieved using a maximising function **find_max** which is shown in Algorithm 2.

Suppose there are m categorical attributes in the data set, A_i denotes the i-th categorical attribute. The distance between two distinct attribute values x and y of any categorical attribute A_i (denoted $\delta(x, y)$) is calculated with respect to each j-th attribute $(j \neq i)$:

Algorithm 2 Function find_max()

```
Input – Two attribute $A_i$ and $A_j$, two attribute values $x$ and $y$ of $A_i$
Output – Distance $\delta^{ij}(x, y)$
Let $v_j$ be the number of categorical values of attribute $A_j$, and $u[t]$ denote a particular value of $A_j$, $1 \leq t \leq v_j$. $P(u[t]|x)$ denotes the probability that an object having value $x$ for $i$-th attribute has value $u[t]$ for $j$-th attribute.
1: $\delta^{ij}(x, y) = 0$ /* distance initialised to 0 */
2: $w' = \emptyset$ /* Support set initialised to NULL */
3: for $t = 1; t < v_j; t{+}{+}$ do
4:   if $P(u[t]|x) \geq P(u[t]|y)$ /* $u[t]$ occurs more frequently with $x$ than with $y$ */ then
5:     add $u[t]$ to $w'$ /* $u[t]$ is added to support set */
6:     $\delta^{ij}(x, y) = \delta^{ij}(x, y) + P(u[t]|x)$
7:   else
8:     add $u[t]$ to $\sim w'$ /* u[t] is added to complement of support set */
9:     $\delta^{ij}(x, y) = \delta^{ij}(x, y) + P(u[t]|y)$;
10:  end if
11: end for
12: $\delta^{ij}(x, y) = \delta^{ij}(x, y) - 1$;
```

$$\delta(x, y) = \frac{1}{m-1} \left(\sum_{j=1}^{m} \delta^{ij}(x, y) \right) \qquad (i \neq j)\,, \tag{6.2}$$

where the following properties hold:

1. $0 \leq \delta(x, y) \leq 1$,

2. $\delta(x, y) = \delta(y, x)$,

3. $\delta(x, x) = 0$.

The pseudocode of ALGO_DISTANCE, which uses the above equation, is given in Algorithm 3.

Following the calculation of the distances of each attribute values, the distance between a pair of data values/points is computed using the Manhattan Distance. Suppose there are two data objects X and Y, where:

- X has a set of m attribute value $\{x_1, x_2, \ldots, x_m\}$,

- Y has a set of attribute values $\{y_1, y_2, \ldots, y_m\}$,

- x_m and y_m are the values of attribute A_m,

Algorithm 3 ALGO_DISTANCE

```
    Input - data set $D$ with $m$ attributes and $n$ data objects, in which the numerical
    attributes have been discretised.
    Output – Distance between every pair of attribute values for all attributes.
 1: for every attribute $A_i$ do
 2:    for every pair of categorical attribute values $(x, y)$ of $A_i$ do
 3:       Sum = 0;
 4:       for every attribute $A_i \neq A_j$ do
 5:          Compute $\delta^{ij}(x, y)$ using find_max();
 6:          Sum = Sum + $\delta^{ij}(x, y)$;
 7:       end for
 8:       $\delta(x, y)$ = Sum/$(m - 1)$
 9:    end for
10: end for
```

the distance between two data objects $\vartheta(X, Y)$ can be computed as follows:

$$\vartheta(X, Y) = \sum_{i=1}^{m} \delta(x_i, y_i) \quad \text{(Manhattan Distance)} \tag{6.3}$$

Ahmad and Dey [6] introduced the use of ALGO_DISTANCE to measure the distance of data with mixed type attributes. This section focuses on a data set in which all attributes are categorical.

6.2.2 Selecting the next pairwise comparisons

The goal of the proposed AL strategy is to make better predictions using as few pairwise comparisons as possible. The pseudocode for the proposed AL strategy is written in Algorithm 4. This strategy aims to reduce the version space by half with the following steps:

1. Exploring step

   In the first step, the exploration of all possibilities is performed as far as possible, as there is no initial information about user preferences. The system starts to learn from a set of pairs which have maximum distances. It is argued that the rules from the first set have a possibility of containing the correct attributes to describe the best car so far. In the system output, these attributes appear as any clause on the left, but not on the right, of the relation `betterthan`.

2. Refining step

   The previous step gives information on which car attributes are preferred by the user, so we can now continue the search by focusing only on these attributes. In this step, more specific pairwise comparisons will be selected, based on the rules produced by the learner in the previous step. As an example, in the exploring step, the learner produces rules as below:

```
(Suv) betterthan (Sedan)
(Manual and Suv) betterthan (Hybrid)
(Suv and Manual) betterthan (Automatic)
(Suv) betterthan (MediumCar and Hybrid)
(Suv and Manual) betterthan (MediumCar)
(Suv and NonHybrid) betterthan (MediumCar)
(Manual) betterthan (Sedan and MediumCar)
(Manual and LargeCar) betterthan (Sedan)
```

Therefore, everything (every attribute) that appears on the left, but not on the right, of the relation `betterthan` is considered to be selected as the most informative pairs and it is concluded that:

```
Attributes of the best car so far = {Suv, Manual, NonHybrid, LargeCar}
```

From the above example, the item set in the refining step is limited only by considering each of the attributes above (i.e. SUV, manual, non-hybrid, and large car). We can use query to retrieve any pair that contains items matched with those characteristics. In a real recommender system, users can perform a filtering search interactively, so that the pairs shown to them are exactly what they want. But in an offline experiment, since we can only use the preferences existing in the data set, all possible best attribute values produced in the first round need to be tested, whether they are good enough to predict the overall user preferences or not.

The refining step is performed for all attribute values found in the first step. For example, the refining step is started from the attribute value `Suv`. Pairs that have the same `Suv` type (a subset of the car dataset that shares the common attribute value `Suv`) were selected. This method made a big contribution to the correctness of the prediction.

Algorithm 4 Active learning module of APARELL

```
1: Measure the distance for every attribute value pair in the data set using
   ALGO_DISTANCE
2: Set parameter $k$ as the sampling size in each batch/step
3: Initialise a set of positive examples $E^+ = \emptyset$
4: Run the initial step:
         • Select $k$ pairs with the furthest distance
         • Add the selected pairs to $E^+$
         • Generate the negative examples $E^-$ where all pairs have the opposite
           order from $E^+$
         • Sort by descending distance
         • Run APARELL
         • Get all attribute values that appear on the left but not on the right
           side (the best item attributes).
5: Run the refining step:
         • Get the best attribute values produced by step 4 (called best_class)
         • Retrieve all pairs which belong to best_class (first item and second
            item in the pairs share a common attribute value of best_class)
         • Select $k$ pairs from best_class with the highest distances
         • Add the selected pairs to $E^+$
         • Generate negative examples $E^-$ where all the pairs have the opposite
           order from $E^+$
6: Sort all pairs in $E^+$ in descending order by distance
7: Run APARELL with a set of examples $E^+$ and $E^-$
```

6.3 Evaluation

6.3.1 Dataset

In this evaluation, the car preferences dataset [1] is used. The description of available cars is shown in Table 6.1.

Table 6.1: Item descriptions

Car ID	Bodytype	Transmission	Fuel	Engine size
1	suv	manual	non-hybrid	small
2	sedan	automatic	hybrid	large
3	sedan	manual	non-hybrid	medium
4	sedan	manual	non-hybrid	large
5	suv	manual	non-hybrid	medium
6	suv	automatic	hybrid	medium
7	sedan	automatic	hybrid	medium
8	suv	automatic	hybrid	small
9	sedan	automatic	non-hybrid	medium
10	suv	automatic	non-hybrid	medium

6.3.2 Experiment setting and result

Before evaluating the proposed strategy, we need to set up the prerequisites, i.e. measure the distance of each attribute value pair in the car dataset. The distance of each attribute value in the dataset using ALGO_DISTANCE [7] is presented in Table 6.2.

Table 6.2: Distance between pairs of attributes based on ALGO_DISTANCE

Value 1	Value 2	Distance
sedan	suv	0.13333
automatic	manual	0.27778
hybrid	non-hybrid	0.27778
large	medium	0.27778
large	small	0.33333
medium	small	0.27778

Three experiments were used to evaluate the proposed AL strategy:

1. Assessing the goodness of different selection strategies by distances on predicting the best car attributes.
2. Comparing two different sorting methods, i.e. maximum and minimum with random sampling.
3. Assessing the proposed AL strategy against random and maximum ordering methods.

Predicting the best car attributes. After measuring the distance for each attribute value in the dataset, it is possible to calculate the distance of all pairs of cars. We need to assess how distance affects the prediction power of the learning algorithm. In this section, we evaluate whether maximum distance is better at predicting than minimum distance or random. They are also compared to the full set of training data.

An experiment to find which distance type contributes most to better prediction was run. This experiment was designed to evaluate how good different selection strategies to predict the best car attributes were. We took an exact number of pairs in each subset of attribute values to limit the data. For each user, we took 9 pairs, with each of them being a member of one of nine different classes (please see Figure 5.1a): `Automatic`, `Manual`, `Suv`, `Sedan`, `Hybrid`, `NonHybrid`, `SmallCar`, `MediumCar`, and `LargeCar`. This can also be explained as the pairs sharing at least one common attribute value between them.

In the maximum distance setting, we selected pairs with the greatest distances, while in the minimum setting, we choose pairs with the closest distances. We ran the learner and analysed how many best attribute values were produced by those pairs. To make it easier for the reader, we borrowed the term *recall* from binary classification, in which the quality of the prediction is measured by the fraction of relevant instances retrieved over the total amount of relevant instances. The same setting were repeated for every user and the average values obtained for all users.

The results of this experiment are shown in Figure 6.2. From the results, it can be argued that when APARELL learns from selected pairs with maximum distances, it can produce a set of rules which contain on average 67.62% correct best car attributes.

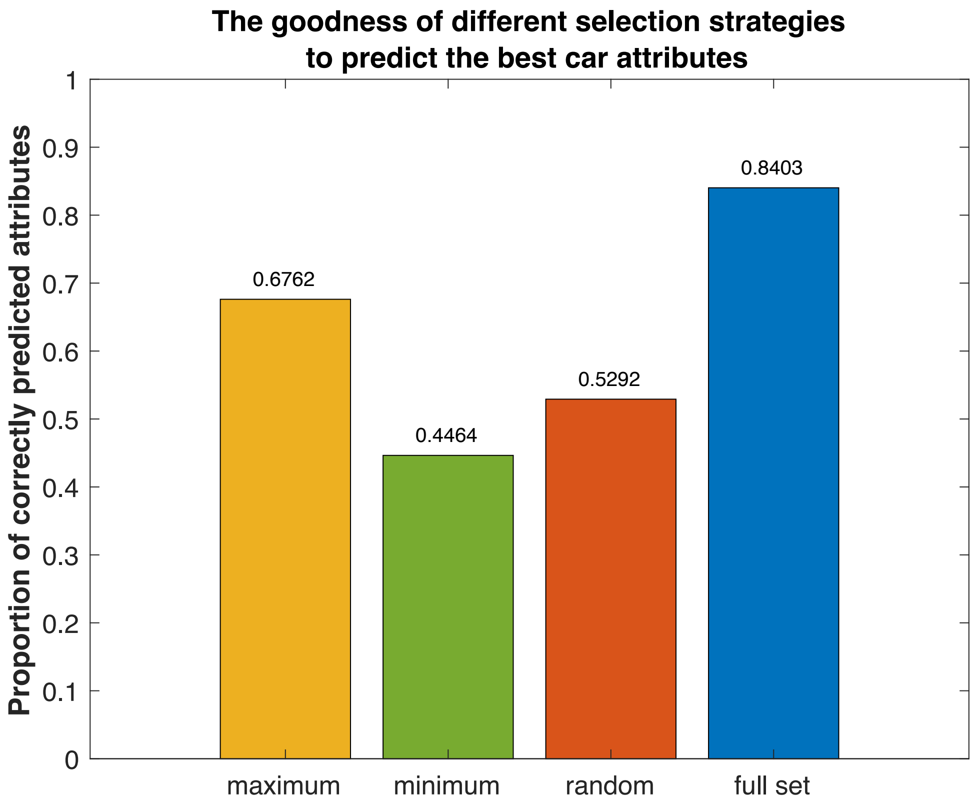


Figure 6.2: Result of the best car attributes prediction

The accuracy of different sorting methods. Our learning algorithm uses a top down approach similar to Aleph/Progol [92]. It builds a hypothesis by combining each attribute for every pair. It starts from the most general (the shortest clause) and stops at a certain level (specified as a parameter). The hypothesis generalisation was done based on the order of the examples (sequentially). This means the rules produced by the algorithm vary depending on how the examples are ordered. It is very important to pay attention to this, especially when there is only a small number of available examples. Another experiment was run on different example orderings and evaluated with 10-fold cross validation. The results of this experiment on different sizes of training examples are shown in Figure 6.3. It is shown that the maximum distance method predicts better than the minimum or random methods.

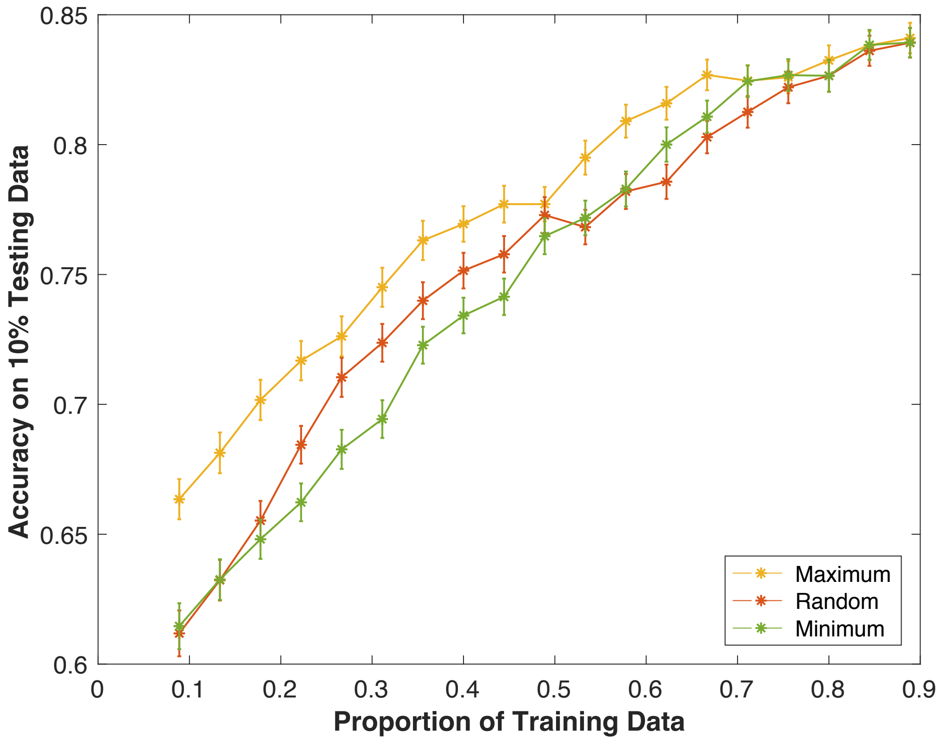


Figure 6.3: Accuracy on the car dataset by different examples ordering

Performance of the proposed AL strategy. In this experiment, the performance of the proposed AL strategy was evaluated. The evaluation was based on how fast it improved in accuracy compared to the random sampling method. The dataset contained 60 different individual preferences. The standard 10-fold cross validation was performed for each user on the dataset and the results are shown in Figure 6.4. In each iteration, the accuracy of a different number of training examples was tested. From the results, it is shown that by sorting the examples using pair distances in descending order, it contributes to better accuracy when compared to random ordering (see the yellow line versus the red line). Even better, as shown in Figure 6.4, the proposed AL strategy can enhance the accuracy of learning from the smallest training size (see the blue line versus the yellow line).

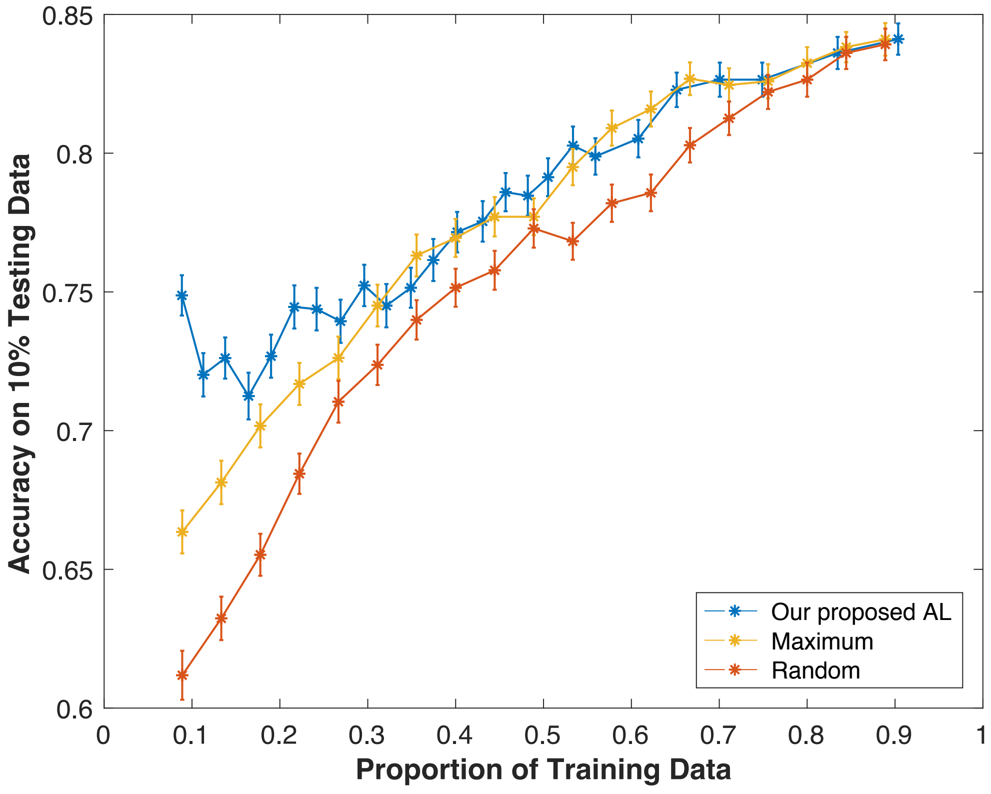


Figure 6.4: Learning curve of our proposed AL strategy

6.4 Summary

In this chapter, a new approach to choosing next pairwise comparisons based on distance measurement between pairs is proposed. This is based on using the same method of greedy ILP systems, where the hypothesis generalisation proceeds sequentially in the order of the examples and then removing the examples which are already covered. The ordering method affected the rules produced by the greedy ILP learner, especially when the number of examples was very limited. From the experiment, it is shown that the proposed AL strategy produces higher accuracy compared to random selection, even in the smallest size of training data.

In the next chapter, the implementation and evaluation of APARELL in a pairwise recommender system is described.

Chapter 7

Pairwise Recommender System Implementation and Evaluation

Despite the practicality of online shopping, in some cases the thousands of available choices can be overwhelming for users. It makes them spend their online shopping time searching for the most suitable items without eventually proceeding to buy the items. Some users may find it easy to find their preferred items online, because they already have a clear preference for what they want, but others may find it difficult. In this chapter, the implementation of a recommender system using pairwise preference elicitation is explained. One of the benefits of using the proposed recommender system is to help users who have difficulty in articulating their preferences, for example, if they are new to the field/shopping category or they want to understand what kind of things they might like.

7.1 Overview of the System

A real-world recommender system application was implemented to help users find their preferred items through a set of pairwise comparisons. The proposed ILP in the DL algorithm which was introduced in Chapter 5 was used to learn the user preferences and finally produce a recommendation. The active learning part was not implemented here, as the system was built interactively, which allowed users to choose the pairs themselves using the filtering system provided in our system. The active learning module would be useful in an offline setting, where there are only datasets without any further interaction with users.

An evaluation of the user experience was done using the ResQue (Recommender systems' Quality of user experience) [102] post-questionnaire and the interface preferences questionnaire. The system is built on a Java platform with a triplestore database, GraphDB, to improve system scalability and performance.

7.2 System Design

This section explains the system design including a system flowchart diagram, system architecture, database design, user interaction and the recommendation algorithm.

7.2.1 System flowchart diagram

The process of the proposed recommender system starts by showing the initial pairs to the user and asking them to see if any of those are interesting to them. The user can use filtering and sorting features to narrow down the pair choices. The system flowchart diagram is shown in Figure 7.1.

7.2.2 System architecture

The proposed system is composed of two main modules. Collaboratively they allow the system to query a set of pairs from the triplestore database to be annotated by the user, so that the system can generate a set of recommendations. Those two main modules are:

1. The learning module.
   The proposed learning algorithm based on ILP in DL used in this module is explained in Chapter 5.

2. Recommender module.
   The system will find the best items for the user based on rules which are produced by the learning algorithm. The steps below are used to produce a recommendation:

   - The output of the learning algorithm is received as an input to the recommendation algorithm.
   - A directed graph is built to express the order of preferences based on the rules produced by the learner.
   - The value attributes in the graph that appear the best (i.e. no values better than them) are obtained.
   - Find a set of items with the above constraints.

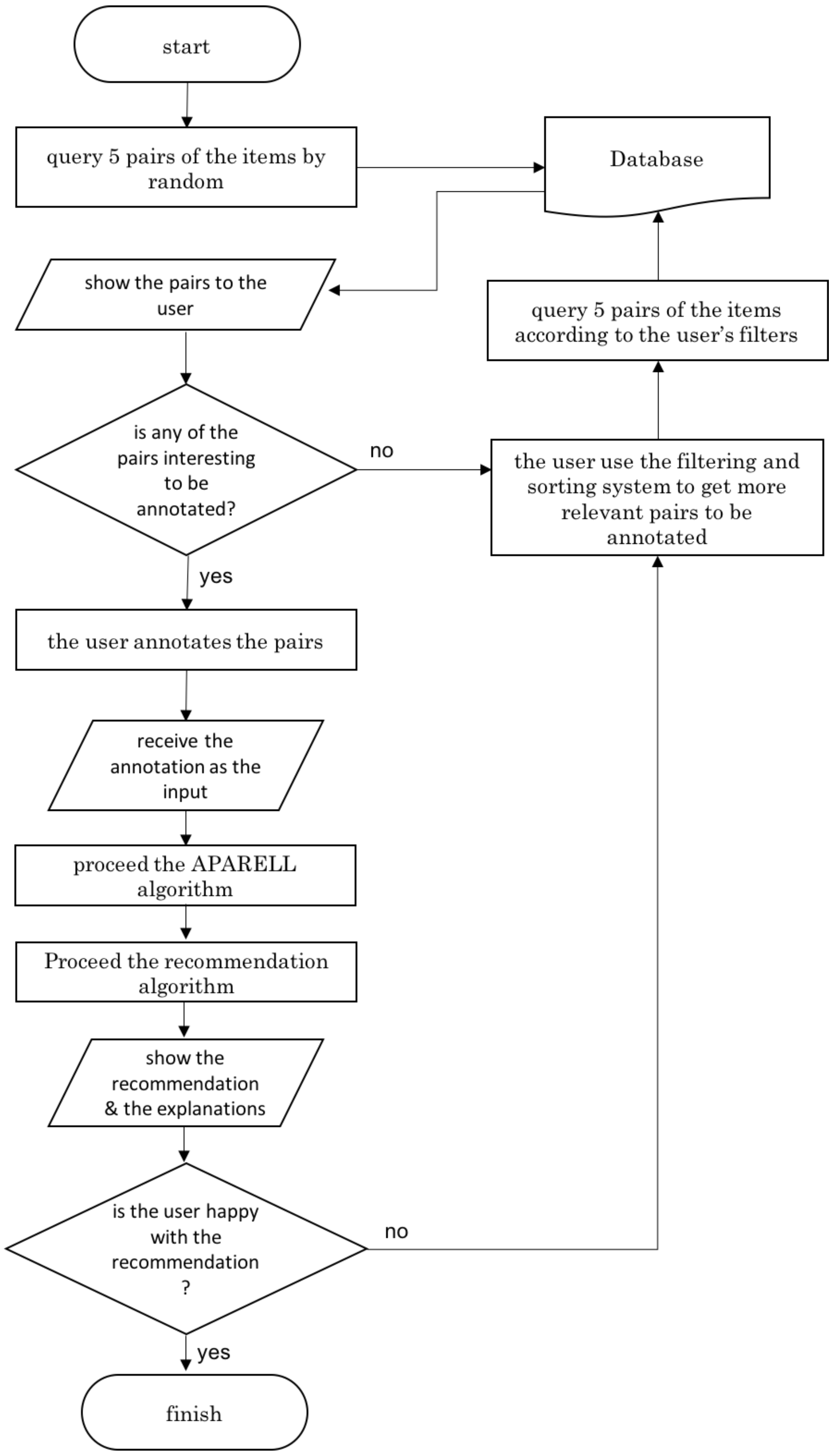

start
query 5 pairs of the items by random
Database
show the pairs to the user
query 5 pairs of the items according to the user's filters
is any of the pairs interesting to be annotated?
no
the user use the filtering and sorting system to get more relevant pairs to be annotated
yes
the user annotates the pairs
receive the annotation as the input
proceed the APARELL algorithm
Proceed the recommendation algorithm
show the recommendation & the explanations
is the user happy with the recommendation ?
no
yes
finish

Figure 7.1: System flowchart diagram

More details about the recommendation algorithm are provided in Section 7.2.3. The architecture of the system is shown in Figure 7.2.

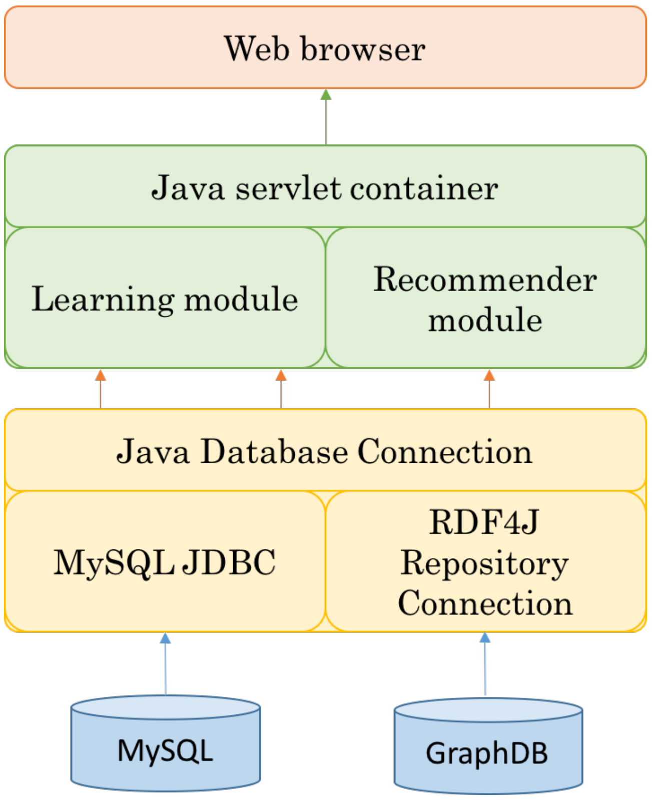


Figure 7.2: Recommender system architecture

7.2.3 Recommendation algorithm

A set of rules produced by the learning algorithm in Chapter 5 shows the attribute values, with one being better than the others, as in the example below:

```
(Suv) betterthan (Sedan)
(Manual) betterthan (Hybrid)
(Suv) betterthan (Automatic)
(Suv) betterthan (MediumCar)
(NonHybrid) betterthan (MediumCar)
(MediumCar) betterthan (Automatic)
```

The recommendation is built based on a set of the best attributes as shown in the above hypotheses. To make this clear, the directed graph in Figure 7.3 shows which attributes are the best. We can see that the node: NonHybrid, SUV, Manual does not have any predecessor nodes (no incoming arrows). These attributes become the criteria for selecting the recommended items and are shown to users as an explanation.

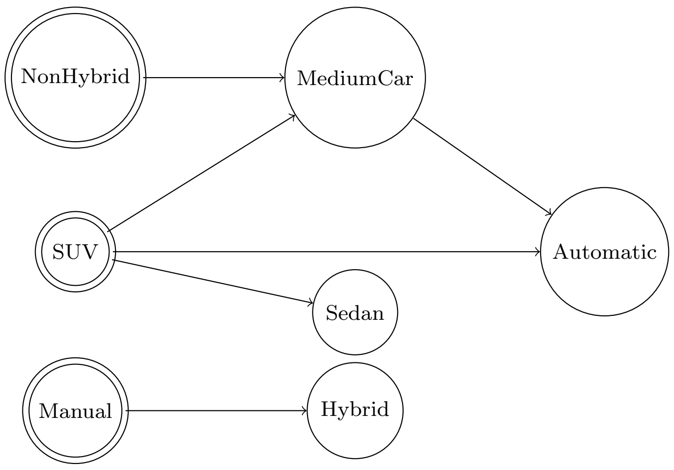


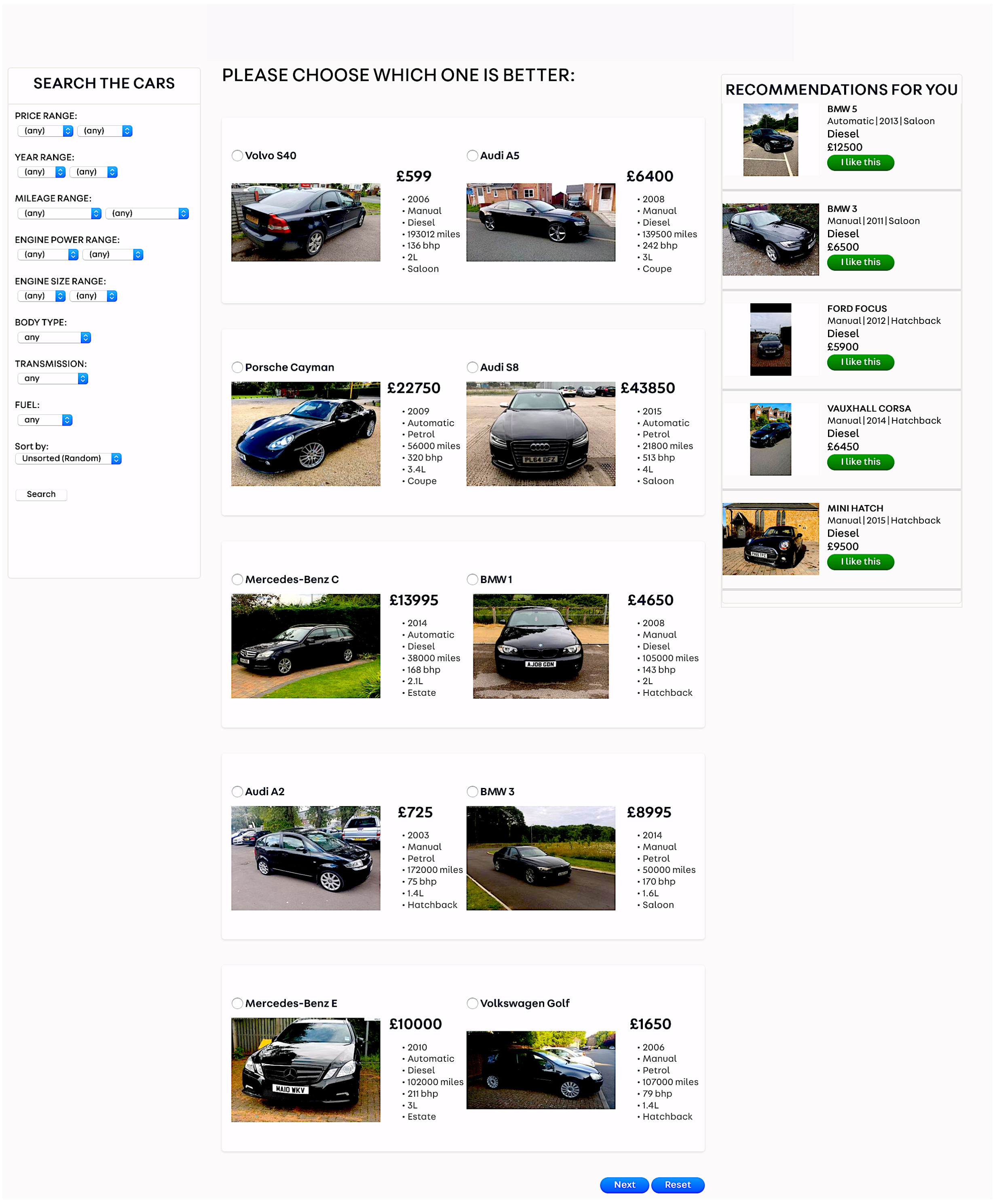
Figure 7.3: Directed graph of hypotheses produced by the learner

7.2.4 Database design

In this recommender application, the dataset was implemented using two different types of database system, relational DBMS (i.e MySQL) for storing user profiles and triplestore DBMS (i.e GraphDB) for storing user preferences and the product knowledge base.

7.2.5 User interaction

A web interface was developed to allow the users to interact with the system easily. The users selected a ‘better item’ from the pairs, based on their preferences. They were allowed to skip pairs in cases where none of the items in the pair matches their preferences. They could also use the filtering system for narrowing down their choices. The pairwise user interface is shown in Figure 7.4. The system showed five pairs on each page, which consisted of 10 items in pairs. The next button was available to see more choices. At the end of the interactive process, the system showed a set of recommended items to the users, along with the reason, based on preferences that have been fed into the system (see Figure 7.5).



SEARCH THE CARS

PRICE RANGE: (any) (any)

YEAR RANGE: (any) (any)

MILEAGE RANGE: (any) (any)

ENGINE POWER RANGE: (any) (any)

ENGINE SIZE RANGE: (any) (any)

BODY TYPE: any

TRANSMISSION: any

FUEL: any

Sort by: Unsorted (Random)

Search

PLEASE CHOOSE WHICH ONE IS BETTER:

Volvo S40 £599 • 2006 • Manual • Diesel • 193012 miles • 136 bhp • 2L • Saloon

Audi A5 £6400 • 2008 • Manual • Diesel • 139500 miles • 242 bhp • 3L • Coupe

Porsche Cayman £22750 • 2009 • Automatic • Petrol • 56000 miles • 320 bhp • 3.4L • Coupe

Audi S8 £43850 • 2015 • Automatic • Petrol • 21800 miles • 513 bhp • 4L • Saloon

Mercedes-Benz C £13995 • 2014 • Automatic • Diesel • 38000 miles • 168 bhp • 2.1L • Estate

BMW 1 £4650 • 2008 • Manual • Diesel • 105000 miles • 143 bhp • 2L • Hatchback

Audi A2 £725 • 2003 • Manual • Petrol • 172000 miles • 75 bhp • 1.4L • Hatchback

BMW 3 £8995 • 2014 • Manual • Petrol • 50000 miles • 170 bhp • 1.6L • Saloon

Mercedes-Benz E £10000 • 2010 • Automatic • Diesel • 102000 miles • 211 bhp • 3L • Estate

Volkswagen Golf £1650 • 2006 • Manual • Petrol • 107000 miles • 79 bhp • 1.4L • Hatchback

Next Reset

RECOMMENDATIONS FOR YOU

BMW 5
Automatic | 2013 | Saloon
Diesel
£12500
I like this

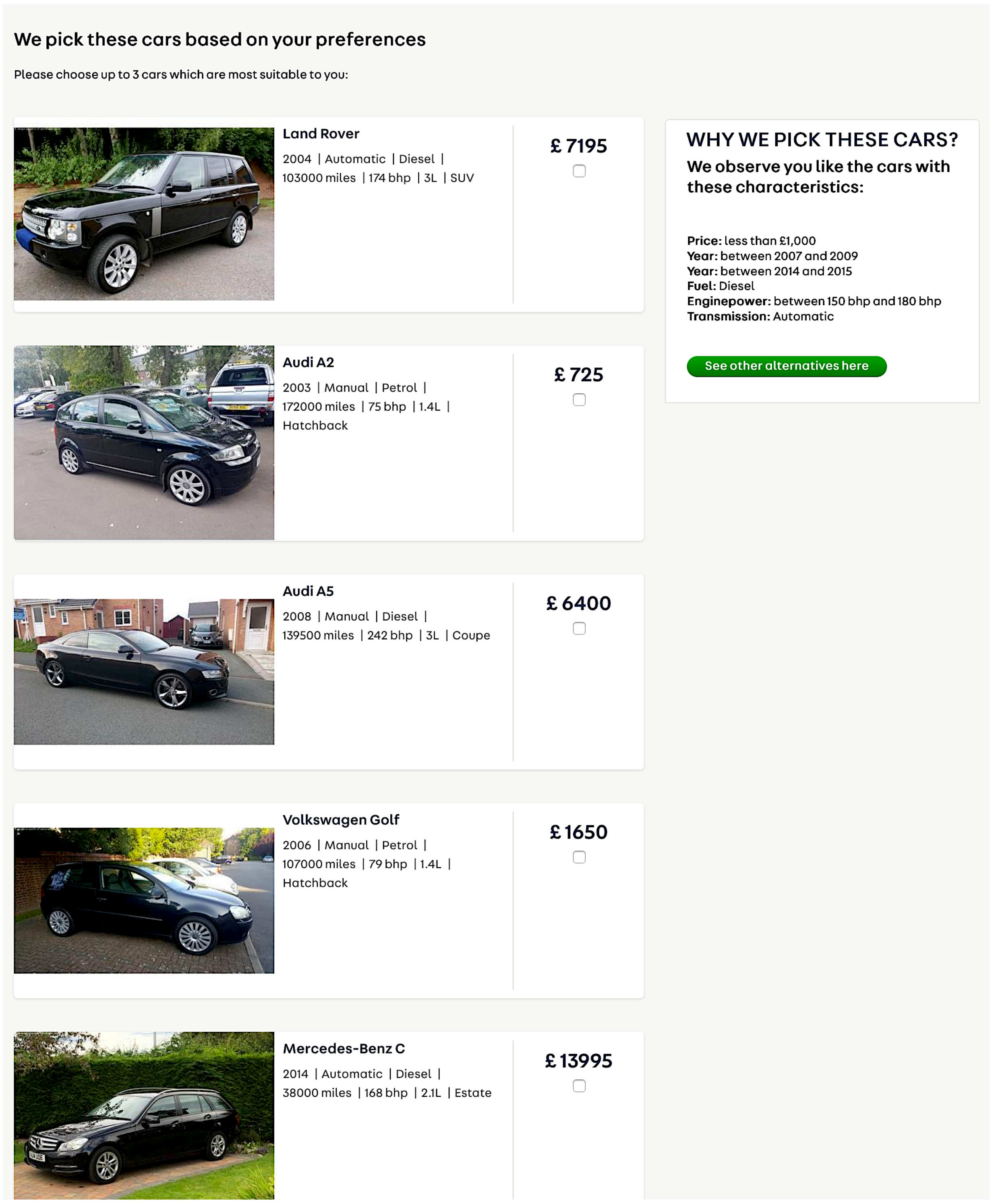
BMW 3
Manual | 2011 | Saloon
Diesel
£6500
I like this

FORD FOCUS
Manual | 2012 | Hatchback
Diesel
£5900
I like this

VAUXHALL CORSA
Manual | 2014 | Hatchback
Diesel
£6450
I like this

MINI HATCH
Manual | 2015 | Hatchback
Diesel
£9500
I like this

Figure 7.4: Pairwise user interface



We pick these cars based on your preferences

Please choose up to 3 cars which are most suitable to you:

Land Rover
2004 | Automatic | Diesel |
103000 miles | 174 bhp | 3L | SUV
£ 7195

Audi A2
2003 | Manual | Petrol |
172000 miles | 75 bhp | 1.4L |
Hatchback
£ 725

Audi A5
2008 | Manual | Diesel |
139500 miles | 242 bhp | 3L | Coupe
£ 6400

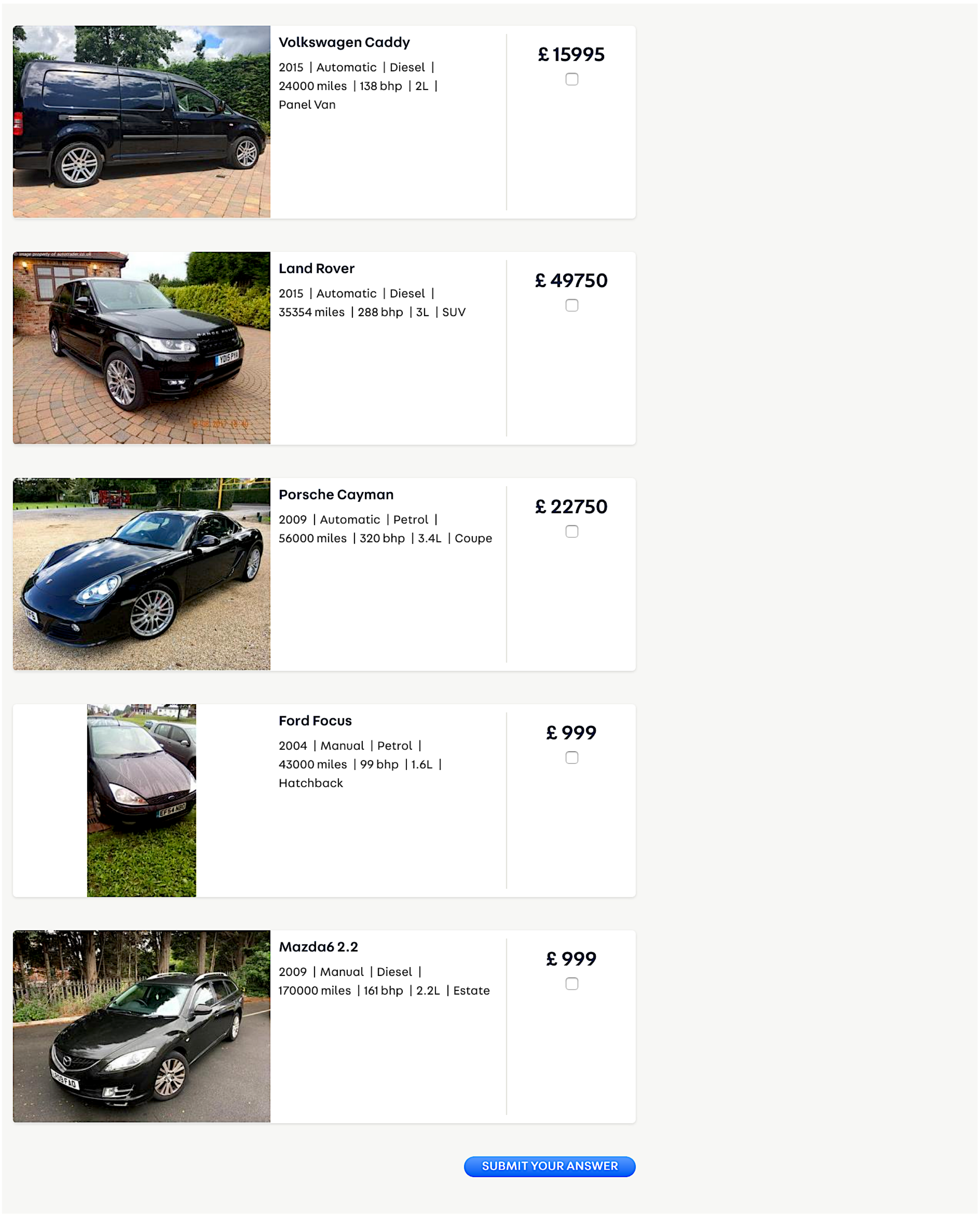
Volkswagen Golf
2006 | Manual | Petrol |
107000 miles | 79 bhp | 1.4L |
Hatchback
£ 1650

Mercedes-Benz C
2014 | Automatic | Diesel |
38000 miles | 168 bhp | 2.1L | Estate
£ 13995

WHY WE PICK THESE CARS?
We observe you like the cars with these characteristics:

Price: less than £1,000
Year: between 2007 and 2009
Year: between 2014 and 2015
Fuel: Diesel
Enginepower: between 150 bhp and 180 bhp
Transmission: Automatic

Figure 7.5: The best items recommendation interface



Volkswagen Caddy
2015 | Automatic | Diesel | 24000 miles | 138 bhp | 2L | Panel Van
£ 15995 ☐

Land Rover
2015 | Automatic | Diesel | 35354 miles | 288 bhp | 3L | SUV
£ 49750 ☐

Porsche Cayman
2009 | Automatic | Petrol | 56000 miles | 320 bhp | 3.4L | Coupe
£ 22750 ☐

Ford Focus
2004 | Manual | Petrol | 43000 miles | 99 bhp | 1.6L | Hatchback
£ 999 ☐

Mazda6 2.2
2009 | Manual | Diesel | 170000 miles | 161 bhp | 2.2L | Estate
£ 999 ☐

Figure 7.6: The best items recommendation interface (cont.)

7.3 Online Evaluation

The study was designed by following the user evaluation framework for recommender systems (**ResQue**), which was introduced by Pu and Chen [102] to evaluate the quality of recommender systems from the user perspective. The participants were recruited at the University of York in November 2017.

The study was conducted using a within-participant design, which means each user performed two different tasks. Users were asked to evaluate the pairwise interface as shown in Figure 7.4 and compare it to the standard list interface as shown in Figure 7.8. Users' click behaviours and the execution time were automatically recorded in log files.

At the beginning, participants were briefed on the upcoming tasks and given instructions for using both interfaces. After the briefing session, participants signed an informed consent form and started the study. In order to clarify the evaluated interfaces to the participants, a user manual was shown and a brief description was given by the experimenter.

The study started with a series of background and demographic questions being answered anonymously, as shown in Figure 7.7. After evaluating both interfaces, participants filled in a post-stage assessment questionnaire for the respective interfaces. At the end, they were asked to fill in an interface preference questionnaire and took part in a short interview about the experience of using the two interfaces. A set of questions in the post-stage questionnaire are shown in Section 7.3.2.1 and the final questionnaire is shown in Section 7.3.2.2. The overall time needed for each participant was between 15 and 30 minutes.

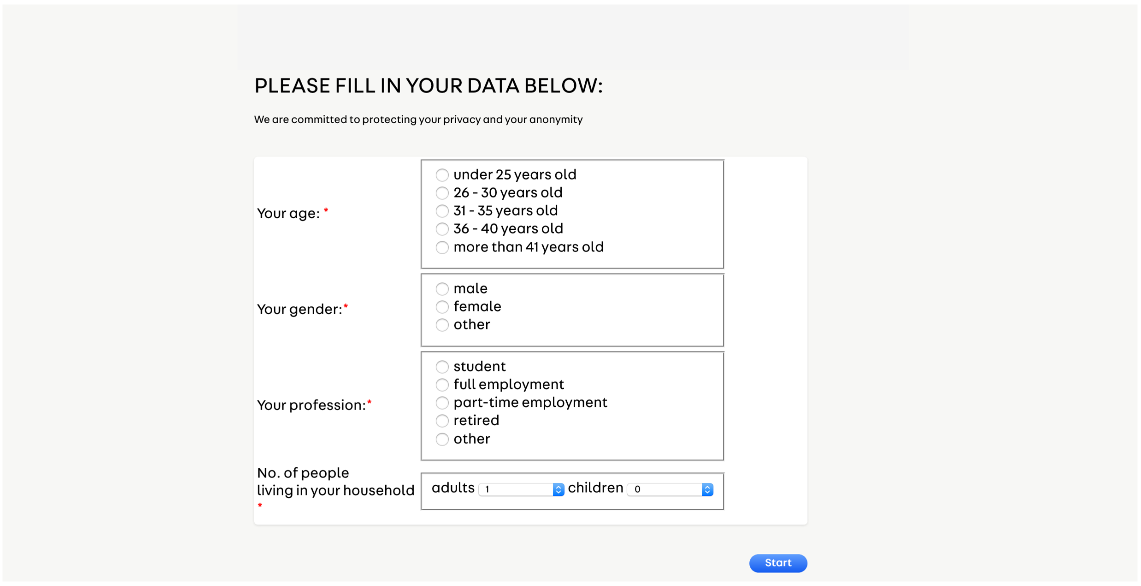
There were 24 participants in this study. The first 20 participants were given a £5 Amazon voucher as a thank you for participating in the study.

Before the user study was conducted, a pilot test was run with three participants, in which some of the system bugs were found and reported. The pilot test is an important step to ensure that the study performs smoothly.

7.3.1 Designing the tasks

The participants were given specific tasks when using each interface. Two similar tasks were given to each user as below:

1. The user needs to find up to three cars suitable for daily commutes between home and office (with a distance of around 5-7 miles each way).


PLEASE FILL IN YOUR DATA BELOW:

We are committed to protecting your privacy and your anonymity

Your age: *	◯ under 25 years old ◯ 26 - 30 years old ◯ 31 - 35 years old ◯ 36 - 40 years old ◯ more than 41 years old
Your gender: *	◯ male ◯ female ◯ other
Your profession: *	◯ student ◯ full employment ◯ part-time employment ◯ retired ◯ other
No. of people living in your household *	adults 1 children 0

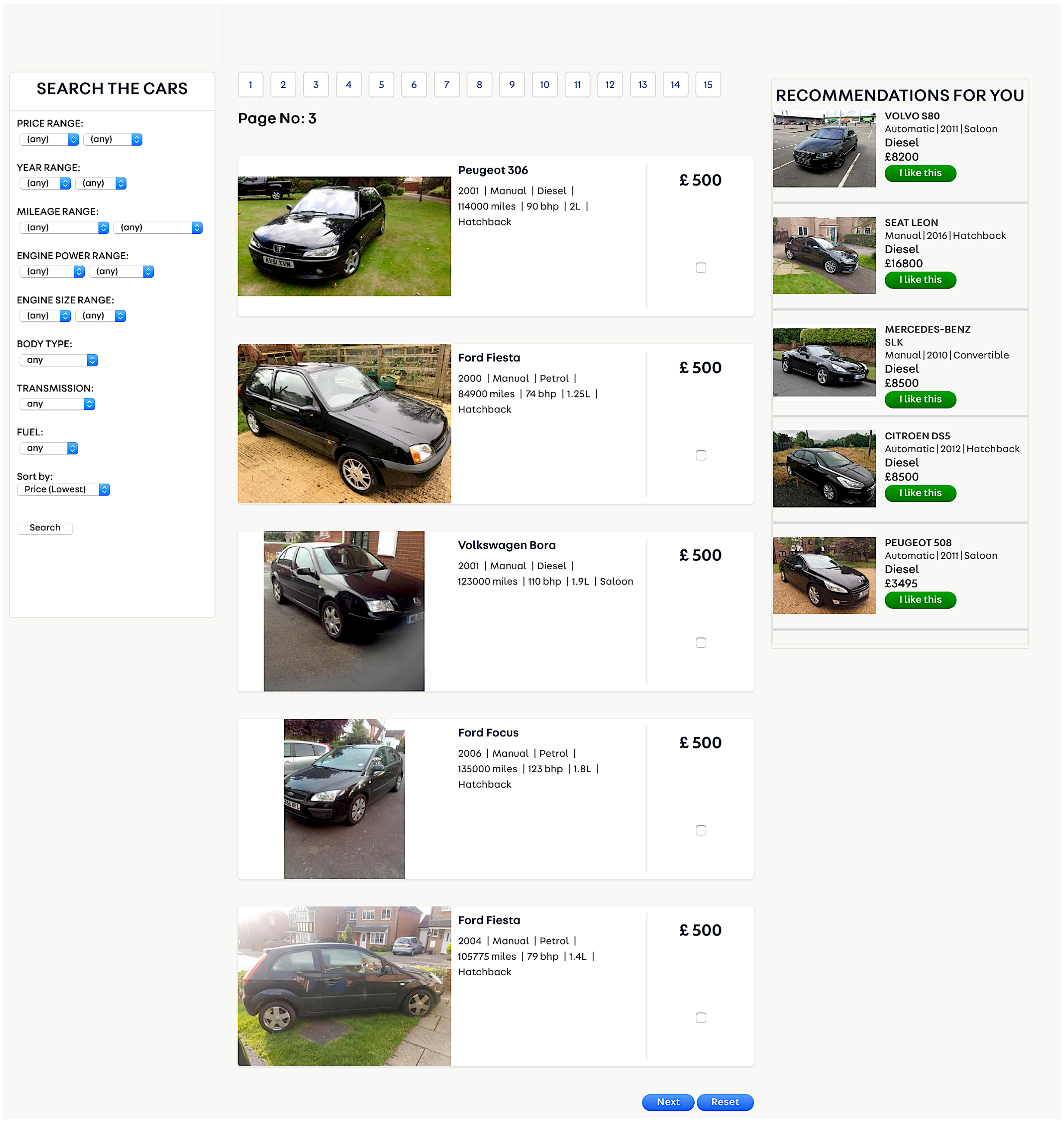
Start

Figure 7.7: User profile interface

2. The user needs to find up to three cars suitable for weekend shopping for a household of four people (with a distance of around 3-5 miles).

The above tasks were designed to be as natural as possible and suitable for all genders and ages. The first task had to be completed using the interface proposed in this thesis i.e. the pairwise comparisons interface. Using this interface, users were also provided with the temporary results of the recommendation (shown at the side of the interface). In the second task, the users had to use the standard list interface with which they are more familiar, as can be found on many e-commerce sites. Using the second interface, users were also provided with a random recommendation (shown on the side of the interface) to help them find the cars. They could click on the 'I like this' button for each option in the side recommendations. Users had to evaluate the quality of the final recommendations, which were shown on the 'Top Picks' page. Users needed to annotate at least 10 pairs (in the first interface) or items (in the second interface) in order to see the recommendation list and finally submit their answer on the 'Top Picks' page. For the pairwise interface, the recommendation was produced using the proposed algorithm explained in Chapter 5, while for the standard list interface, the recommendation was a set of items that had been chosen by the users (no specific algorithm was used here). The users were not informed about the algorithm used by the system, to prevent bias in their answers in the post-stage questionnaire.

There were 7,360 used cars available for participants to choose from. The car dataset



SEARCH THE CARS

PRICE RANGE: (any) (any)

YEAR RANGE: (any) (any)

MILEAGE RANGE: (any) (any)

ENGINE POWER RANGE: (any) (any)

ENGINE SIZE RANGE: (any) (any)

BODY TYPE: any

TRANSMISSION: any

FUEL: any

Sort by: Price (Lowest)

Search

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Page No: 3

Peugeot 306
2001 | Manual | Diesel |
114000 miles | 90 bhp | 2L |
Hatchback
£ 500

Ford Fiesta
2000 | Manual | Petrol |
84900 miles | 74 bhp | 1.25L |
Hatchback
£ 500

Volkswagen Bora
2001 | Manual | Diesel |
123000 miles | 110 bhp | 1.9L | Saloon
£ 500

Ford Focus
2006 | Manual | Petrol |
135000 miles | 123 bhp | 1.8L |
Hatchback
£ 500

Ford Fiesta
2004 | Manual | Petrol |
105775 miles | 79 bhp | 1.4L |
Hatchback
£ 500

Next Reset

RECOMMENDATIONS FOR YOU

VOLVO S80
Automatic | 2011 | Saloon
Diesel
£8200
I like this

SEAT LEON
Manual | 2016 | Hatchback
Diesel
£16800
I like this

MERCEDES-BENZ SLK
Manual | 2010 | Convertible
Diesel
£8500
I like this

CITROEN DS5
Automatic | 2012 | Hatchback
Diesel
£8500
I like this

PEUGEOT 508
Automatic | 2011 | Saloon
Diesel
£3495
I like this

Figure 7.8: Standard list user interface

was crawled from the Autotrader.co.uk website in September 2017. It was relatively up-to-date on the day the study was conducted (in November 2017). The used cars available in this dataset widely varied e.g. they were built between 1965 and 2017, and they were advertised with various prices, the cheapest being £500 and the most expensive £70,000.

7.3.2 Evaluation criteria

7.3.2.1 Post-stage questionnaire

After using each interface, the user was asked to fill in a five-point Likert scale questionnaire (1 strongly disagree up to 5 strongly agree) to evaluate the interface he/she had just tested. Because this questionnaire was a user-centric evaluation, we can also call it a usability and user satisfaction assessment. The evaluation aspects and the questions of the post-stage questionnaire are shown below:

1. Quality of Recommended Items

   (a) Accuracy

   Q1. The items recommended to me matched my interests

   (b) Novelty

   Q2. The items recommended to me are novel and interesting

   (c) Diversity

   Q3. The items recommended to me are diverse

   Q4. The items recommended to me are similar to each other (reverse scale)

2. Interaction Adequacy

   Q5. The recommender explains why the products are recommended to me

3. Interface Adequacy

   Q6. The layout of the recommender interface is attractive and adequate

4. Perceived Ease of Use

   Q7. Finding an item to buy with the help of the recommender is easy

5. Perceived Usefulness

   Q8. I feel supported to find what I like with the help of the recommender

6. Control/Transparency

   Q9. I feel in control of telling the recommender what I want

7. Attitudes

   Q10. Overall, I am satisfied with the recommender

   Q11. I am convinced of the products recommended to me

   Q12. I am confident I will like the items recommended to me

8. Behavioural Intentions

   (a) Continuance and Frequency

      Q13. I will use this recommender again

   (b) Purchase Intention

      Q14. I would buy the items recommended, given the opportunity

7.3.2.2 Interface preferences questionnaire

Finally, all participants were asked to answer a questionnaire about their preferences with the two interfaces in terms of five aspects: general preference, informativeness, usefulness, better at recommending and better at helping perceived diversity. This evaluation follows the interface questionnaire from [59]. The questions used in this questionnaire are shown below:

Q1. Which interface did you prefer overall? (General preference)

Q2. Which interface did you find more informative? (Informativeness)

Q3. Which interface gave you more useful recommendations? (Usefulness)

Q4. Which interface showed the cars you really like? (Better at recommending)

Q5. Which interface was better at showing more diverse choices? (Better at helping perceived diversity)

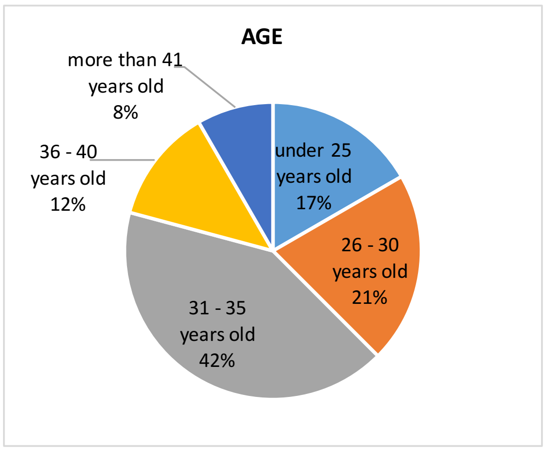
The user manual, the online questionnaire and the consent form are included in Appendix F and E.

7.4 Results Analysis

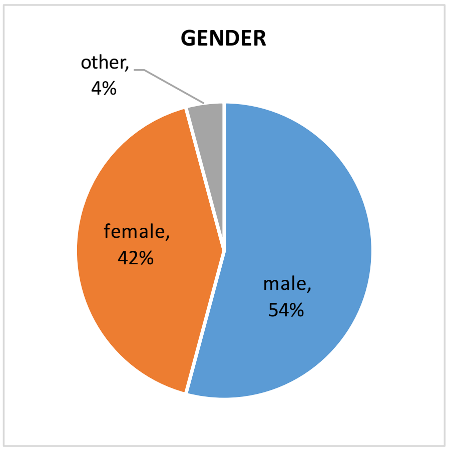
In this section, the experiment's results are reported, which include a demographics report, post-stage questionnaire details and comparisons, an interface preferences questionnaire report, user behaviours (time spent on each interface, number of pages visited, clicks count on the Top Picks page, clicks count for the side recommendations) and short interview results. The details and analysis of each report are also provided.

7.4.1 Demographics report

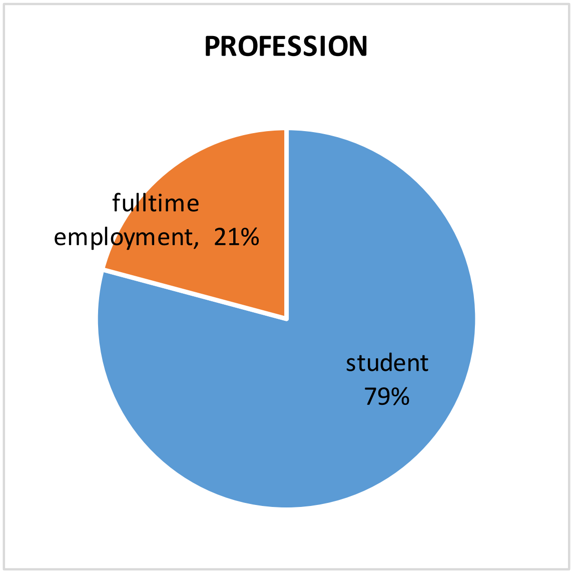
There were 24 participants of different ages, genders and professions in this study. Some of the participants were students (PhD) or in full-time employment (research associates) and their ages ranged from 20 to 45. The proportion of females and males was quite balanced, with 4% identifying as other. The demographic profiles based on age, gender and profession are shown in Figure 7.9.



(a) Participants' profiles based on their age



(b) Participants' profiles based on their gender



(c) Participants' profiles based on their profession

Figure 7.9: Participants' demographics report

7.4.2 Post-stage questionnaire results

The ordinal values of all participants' responses for each question were averaged and the difference between the pairwise and standard list interface responses were tested using a paired sample t-test. The average values of each question are shown side by side between pairwise and standard list to see how they differ. Detailed comparisons on each question are shown in Appendix G.1. The means, standard deviations and paired t-test p-values are shown in Table 7.1.

From all the questions, it can be concluded that the average preferences of the pairwise interface are higher than the standard list interface, except for one question: **Q1:** "The items recommended to me matched my interests", but according to the paired t-test results, the difference was not significant for Q1. Significant differences were found for the questions below:

- **Q3:** "The items recommended to me are diverse"
- **Q5:** "The recommender explains why the products are recommended to me"
- **Q7:** "Finding an item to buy with the help of the recommender is easy"
- **Q11:** "I am convinced of the products recommended to me"
- **Q13:** "I will use this recommender again"
- **Q14:** "I would buy the items recommended, given the opportunity".

While the means of the two questionnaires provide a general comparison of the interfaces, the proportion details for each question show more meaningful results. More details on the response proportions for each question are provided in Figure G.1 using bar charts and Figure G.2 using pie charts. For example, for Q1. "The items recommended to me matched my interests", the means comparison of both interface assessments shows that the means of the list interface are slightly higher than those of the pairwise interface. However, when we assess the details, in the pairwise interface assessment, there were more, "strongly agree", statements than in the list interface assessment (54% vs 50%, resp.). In this case, the difference between the two statements "agree" and "strongly agree" is quite meaningful in drawing a conclusion. For all questions, the pairwise interface got more "strongly agree" statements than the list interface, except in three cases: Q4. "The items recommended to me are similar to each other", where this is the reverse scale (we can assess this case by observing the opposite statement: disagree vs strongly disagree); Q6. “The layout of the recommender interface is attractive and adequate”; and Q9. “I feel in control of telling the recommender what I want”, where for the latter two questions, they are a draw (the proportions of “strongly agree” statements are the same for both interface assessments).

7.4.3 Interface preferences questionnaire results

The final questionnaire consisted of an evaluation of five different factors of the interface preferences. These were, general preference, informativeness, usefulness, better at recommending and better at helping perceived diversity. From those five factors, the number of participants who preferred the pairwise interface over the list interface was higher than the opposite preference, except in one aspect, better at recommending, where the pairwise interface got the same votes as the list interface. The results of the interface preferences are presented in Figure 7.11. The dotted line at 50% is drawn to make differences between votes clearer.

For the first factor, preferred interface, the results show that 54% of participants preferred the pairwise interface while 42% of the participants preferred the standard list interface. This means that the pairwise interface got slightly higher votes than the standard list interface. For the second factor, informativeness, 38% of the participants chose the pairwise interface while only 21% of the participants chose the standard list interface. However, for this factor, participants who chose both interfaces as the more informative interface reported as high as 42%. For the third factor, usefulness, the pairwise interface gained 46% of the votes; this is higher than that for the standard list interface, which only got 38%. For the fourth factor, better at recommending, the pairwise and the standard list interfaces got the same votes at 38%. This is in line with the previous section on the post-stage questionnaire results, where participants were asked about the accuracy of the recommendation (Q1. The items recommended to me matched my interests). The results also show almost the same number of voters between the pairwise and standard list interfaces. For the last factor, better at helping perceived diversity, the participants who chose the pairwise interface were significantly higher than the standard list interface with 67% vs 17%.

Table 7.1: Mean and standard deviation of post-stage questionnaire
Q with * mark is a question with the significant difference at $\alpha = 5\%$ ($p - value < 0.05$)

	Q1 Accuracy	Q2 Novelty	Q3* Diversity	Q4 Diversity (reverse scale)	Q5* Interaction	Q6 Interface	Q7* Ease of Use	Q8 Usefulness	Q9 Control	Q10 Satisfaction	Q11* Influence	Q12 Confidence	Q13* Intention to reuse	Q14* Intention to buy
Pairwise	4.38±0.92	4.25±0.79	3.88±0.99	2.75±1.26	4.25±0.79	4.08±0.58	4.33±0.64	4.21±0.66	4.04±0.75	4.38±0.58	4.21±0.72	4.17±0.87	4.50±0.59	4.29±0.62
List	4.42±0.65	3.79±0.83	3.17±1.13	3.25±1.15	3.08±1.35	3.83±0.92	3.96±0.81	4.04±0.81	3.79±0.98	4.08±0.78	3.92±0.72	4.00±0.78	4.13±0.90	4.00±0.72
$p - value$	0.8619	0.0694	**0.0208**	0.1035	**0.0019**	0.1853	**0.0359**	0.3824	0.1617	0.1292	**0.0499**	0.2567	**0.0359**	0.0256

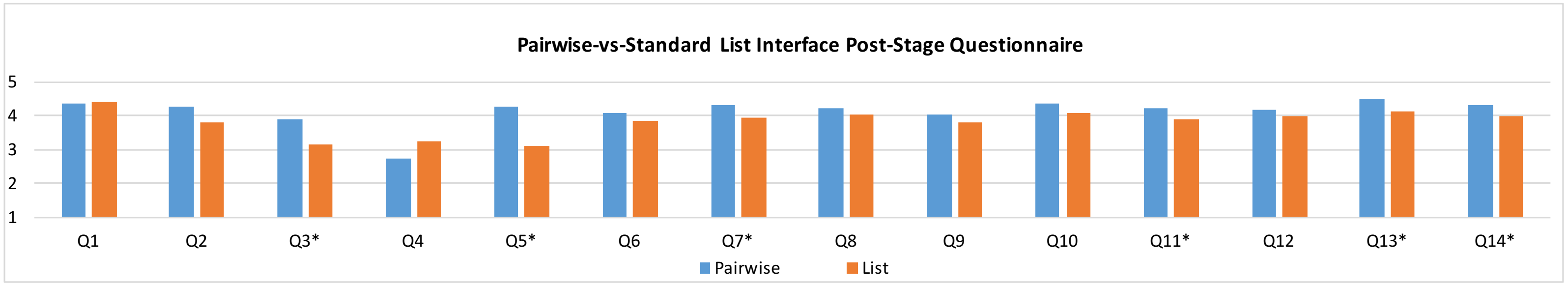


Figure 7.10: Usability and user satisfaction assessment results
Q with * mark is a question with the significant difference at $\alpha = 5\%$ ($p - value < 0.05$)

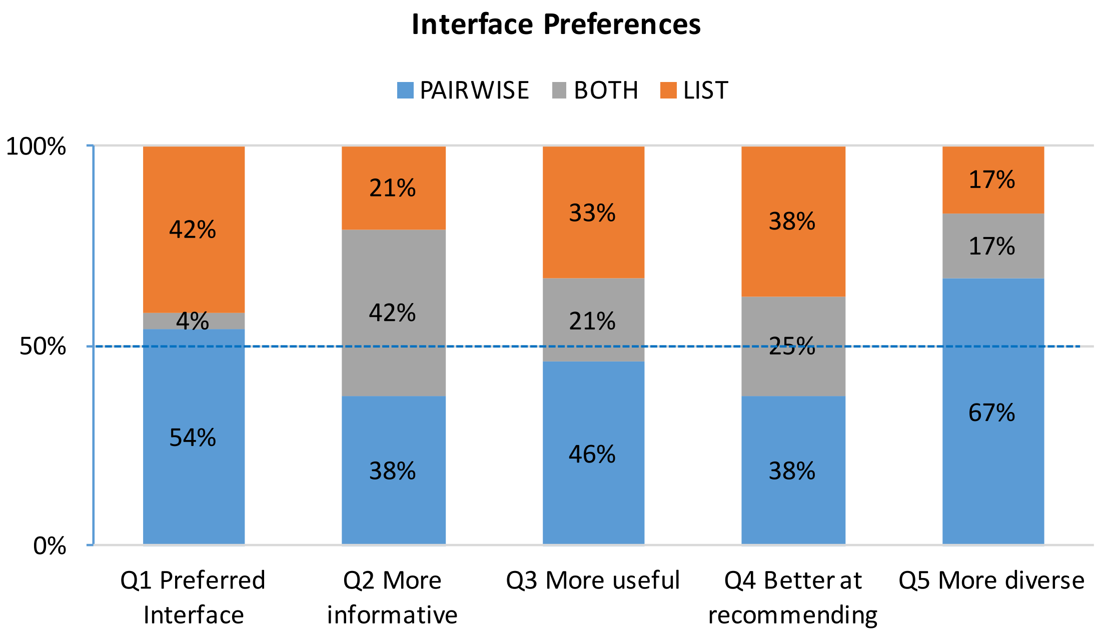


Figure 7.11: Interface preferences questionnaire results

Details of the interface questionnaire is provided in Appendix G.2. The composition of participant profiles based on their ages and who gave votes to three of the options pairwise, list and both interfaces is reported in Figure G.3. The voters' details based on their genders are shown in Figure G.5 and the voters' details based on their professions are shown in Figure G.7. From those three figures, it needs to be emphasised that the pairwise interface got votes from different types of participants. There was no dominant user type who liked the pairwise interface.

7.4.4 User behaviours

7.4.4.1 Time spent

Participants completed all of the tasks in about 15 - 30 minutes. The average time needed to complete the task using the pairwise interface was 7 minutes and 28 seconds. To complete the task using the standard list interface, they needed slightly longer, at 8 minutes and 20 seconds on average. According to the paired sample t-test p-value at the 5% level, there is no significant difference between the time spent on the pairwise and the standard list interfaces. The number of participants who worked longer on the pairwise interface was 10 people, while the number of participants who worked longer on the standard list interface was 14 people. Details of the time spent on the study are reported in Table 7.2.

Table 7.2: Users' time spent (in minutes)

	pairwise interface	list interface
user1	08:09	18:02
user2	09:04	13:02
user3	05:52	16:04
user4	07:42	08:08
user5	07:22	04:42
user6	06:27	06:08
user7	07:19	02:57
user8	09:01	06:10
user9	06:50	12:27
user10	06:54	10:39
user11	09:18	07:53
user12	08:14	07:38
user13	17:28	10:41
user14	09:39	09:17
user15	04:19	06:26
user16	08:59	09:12
user17	03:21	05:23
user18	05:28	08:55
user19	16:50	06:26
user20	04:04	04:25
user21	05:14	05:17
user22	03:38	02:46
user23	01:31	02:29
user24	06:40	14:55
average	07:28	08:20

7.4.4.2 Pages visited

The users' behaviours when interacting with the system were also observed, i.e. how many pages they opened to complete each task. In the pairwise interface, five pairs of cars (10 individual cars) were shown on each page, while in the standard list interface, there were five used cars on each page. The users' behaviours when exploring the pages while searching for their favourite cars are quite interesting. The number of pages visited by each user is presented in Table 7.3. The average number of pages visited using the pairwise interface was 12.08, while for the standard list interface it was reported as high as 49.96 on average. There is a significant difference in the number of pages visited between the pairwise and list interfaces with the p-value of the paired t-test ($\alpha = 0.05$) being 0.0089. From this report, we can conclude that using the pairwise interface can reduce the number of pages visited by up to 76%, which can be interpreted as simplifying the task.

Table 7.3: Number of pages visited

	pairwise interface	list interface
user1	3	28
user2	7	68
user3	5	46
user4	3	30
user5	4	8
user6	9	36
user7	17	22
user8	3	10
user9	7	32
user10	4	52
user11	5	32
user12	114	429
user13	32	52
user14	9	26
user15	2	40
user16	18	32
user17	2	10
user18	8	16
user19	14	18
user20	3	12
user21	10	32
user22	5	14
user23	3	18
user24	3	136
average	12.08	49.96

7.4.4.3 The number of cars submitted as the answers

In each task, the participants needed to find up to three cars they liked. The average number of cars submitted as answers for the pairwise interface was 2.29 and it was 2.42 for the standard list interface. The details of these numbers are shown in Table 7.4.

7.4.4.4 Clicks count on the side recommendation

A side recommendation was provided as an additional feature of the system to help the users find suitable cars. It seems that not many participants took advantage of the side recommendation, as this was only an additional feature. Most of the time, the participants focused on the main part of the page to complete the tasks. The total clicks count on the side recommendation of the pairwise interface was 11 and for the standard list, it was only 4.

Table 7.4: Number of cars submitted as the answers

	pairwise interface	list interface
user1	3	3
user2	1	1
user3	3	3
user4	2	3
user5	3	3
user6	2	1
user7	1	2
user8	1	3
user9	2	3
user10	1	2
user11	3	3
user12	3	1
user13	1	3
user14	3	1
user15	1	3
user16	3	3
user17	3	3
user18	3	1
user19	3	3
user20	3	2
user21	3	2
user22	3	3
user23	1	3
user24	3	3
average	2.29	2.42

7.4.5 Short interview results

Short interviews were conducted after each participant finished the tasks, including filling in the three questionnaires. The questions were: "How do you feel after evaluating these two interfaces? Which one do you like and why?" And "what about the recommendation that the system gave to you?". Participants talked about their opinions and the sessions were recorded. The full interview transcripts are presented in Table H.1 in Appendix H. Please note that the interview IDs are provided to make the points in the coding table clearer and they are not related to the user IDs in the previous table.

Transcripts were coded manually by annotating the important keywords mentioned by the respondents in the short interview process. Each keyword is classified into three relevant categories: (1) interface preference, (2) reason, and (3) general comment. The second category, reason, is divided into two more specific classifications: (a) reasons for preferring the pairwise interface, and (b) reasons for preferring the standard list view. The transcript codes and categories are shown in Table H.2 in Appendix H. The keywords found in each category and the count for each keyword (shown as a number in brackets) are given below:

1. Interface preference
    - preferring to the pairwise (14)
    - preferring to the list view (10)
2. Reason
    (a) reasons for preferring the pairwise interface
    - correct explanation (match the interest) (6)
    - comparing is easy (4)
    - more diverse choices (4)
    - helpful recommendation explanation (5)
    - helpful to understand the expectations (3)
    - displaying more cars (4)
    - more informative (2)
    - interesting (2)
    - relevant (1)
    - specifications is shown next to picture (1)
    - helpful (1)
    - forcing me to compare (1)
    - capability to compare similar cars (1)
    - recommending some surprising items (1)
    - helpful side recommendation (1)
    - helpful to find the preferred cars (1)
    - better design (1)
    - less often to click the ‘next’ button and scroll down (1)
    - correct recommendation (1)
    - ability to narrow down the choices (1)

- feel in control (1)
- helpful filtering feature (1)

(b) reasons for preferring the standard list view

- the pairs are confusing (5)
- dislike of comparing (3)
- easier to use (3)
- freedom to choose (2)
- not aware of filtering in the pairwise interface (2)
- more familiar (2)
- comfortability on the screen (1)
- having a clear idea on what to buy (1)
- filtering is more useful with the list view (1)
- more relevant (1)
- more convincing (1)
- attractive recommendation (1)
- more diverse (1)
- expecting to compare more similar cars (1)
- can stay focus on searching (1)

3. General comment

- good (8)
- correct recommendations (6)
- useful (3)
- helpful (3)
- expecting more informative pictures of cars (3)
- expecting more items to compare (rather than in pairs) (2)
- comparisons is suitable for someone who unsure about the choices (2)
- list interface is suitable for someone who already knows his/her expectations (1)
- expecting more data in the system (1)
- expecting most popular items (1)

- expecting more options in the filtering (1)
- knowing the reason of the recommendation is helpful (1)
- concern to use pairwise for actually buying a car (1)
- pairwise interface showing my preferred car in the first page (1)
- list interface showing my preferred car in the second page (1)
- easy to use (1)
- interactive (1)
- interesting (1)
- informative (1)
- good interface (1)
- convincing (1)
- expecting user reviews in each car (1)
- recommendation explanation can be improved (1)
- specific range in the explanation is not really helpful (1)
- filtering in the pairwise interface can be improved (1)
- design of the explanation can be improved (1)

From the interviews, we can conclude that participants' responses align with the questionnaire results where more than half of them showed an interest in the pairwise interface. In summary, there were a number of reasons of why some participants found that the pairwise interface was interesting, such as they did not really understand what type of car they liked, so the comparisons helped them to know about types of car more easily. They also liked the explanation given by the system to help them understand their preferences. Some of them liked a surprise and felt more excited when the recommendation shown to them was different from the one that they had chosen before. In the recommender system research, this is called the *novelty* of the recommendation. Discussing the interface, they found an advantage in seeing more cars displayed on a page, so there was no need for them to click the 'next' button and scroll down too often.

On the other side, we observe the people who prefer the standard list interface over the pairwise gave reasons such as it was more familiar, they already knew what they really wanted to buy, and they did not like to be forced to compare cars. Sometimes, the pairs were more confusing for them. We also highlighted some important feedback about the interfaces in general, such as the pictures could be more informative, the filtering feature could be improved, and the system could show the most popular cars and user reviews of each car.

7.5 Summary

In this chapter, the learning algorithm proposed in Chapter 5 was implemented in a real-world used car recommender system and the evaluation was reported. The system was evaluated using both a user-centric evaluation framework, ResQue, and an interface preference questionnaire. There were 24 participants, recruited from the University of York in this study. Some interesting findings are outlined in this chapter which include the fact that our proposed pairwise interface gained more votes in the ResQue than the common list interface on all aspects of the evaluation, except on the first aspect about the accuracy of the recommendation, which was a draw. Some of them also had significant differences according to the paired t-test results. In the interface preference questionnaire, the pairwise interface also showed a better result compared to the standard list interface, except for one factor on usefulness, which was again a draw. The short interview results are also interesting, where the real experience on using the interfaces was observed in more detail. The results of our experiment show that the use of a pairwise interface in a recommender system offered an interesting new experience to the user.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

This thesis provides an overview of machine learning techniques to solve the problem of learning from pairwise comparisons. It studies the comparison between statistical and logic-based methods. The benefits of using logic-based methods with their richer representations are also discussed. The contributions in this thesis are in the area of machine learning and recommender systems.

In the machine learning area, a novel approach of Inductive Logic Programming (ILP) in Description Logic (DL), called APARELL (Active PAirwise RELation Learner), is proposed. Existing work on ILP in DL is still limited to learning class descriptions. Here we propose an implementation of ILP in DL to learn binary relations from pairwise comparisons. The capability of the system to read from a remote Resource Description Framework (RDF) data is also presented in order to gain the benefit of using the DL representation with a standardised syntax. A command line tool was developed and made available to the public to be used under the General Public License (GPL). Several experiments to evaluate the accuracy and performance of the proposed algorithm were conducted. The accuracy was measured using four different aspects, on relatively small but complete and very detailed datasets, on a larger size dataset, on a dataset with a more complex class hierarchy and on various training example sizes. The evaluation shows that in terms of accuracy, APARELL outperformed other baseline machine learning algorithms.

When examining the existing algorithms to address the pairwise preference learning, it could be argued that ILP system based on First Order Logic (FOL) performance is sufficient, as well as using the propositional logic and statistics-based systems. However, some studies on DLs [12] claim that by using DL representation, the knowledge can be represented in a more human-friendly and readable way than the other representations. It can be integrated easily into an online e-commerce platform such as the recommender system presented in this thesis. Another advantage of using DL representation is that background knowledge, which usually needs to be reformatted from the original data to be understood by the learner, is already available as an ontology. It can be easily generated using an Integrated Development Environments (IDEs) with a Graphical User Interface (GUI), e.g. Protégé or even easier for domains in which they are available as linked open data. Not only representing the problem in different logic representation, but our proposed algorithm is also improving the accuracy by producing a complete set of consistent hypotheses. Our algorithm uses Closed World Assumption (CWA), which makes it easier to find a consistent hypothesis. For the task of learning strict order, using CWA or Open World Assumption (OWA) produces very similar results.

As an additional contribution in this area, a new approach in Active Learning (AL) has been proposed supporting the learning process given limited data. The experiment shows that the set of hypotheses/rules produced by APARELL depends on the ordering method when processing the training examples and which example are processed first, especially when the number of examples and the attributes of each item are very limited. The proposed AL strategy selects the next pairwise comparison based on the distance measurement between pairs. It is shown that the proposed AL strategy can produce higher accuracy compared to random selection, in a smaller number of examples.

In the recommender system area, the novel approach, APARELL, was implemented in a real-world used car recommender system and the user evaluation study was reported. We propose a new method to produce recommendations with APARELL. The logic approach is still not quite common in recommender systems. Most studies apply a statistics-based approach to produce recommendations, as it is easier and more practical to implement. We show that a logic-based approach can be implemented effectively in a recommender system. A used car dataset collected from Autotrader.co.uk on September 2017 was used in a user study with 24 participants from the University of York. From the online experiments, we achieve a satisfactory result where the majority of participants preferred the pairwise comparisons interface over the standard list and they agree that the use of pairwise comparisons in a recommender system is helpful to find their preferred cars. The pairwise interface also offers more diverse recommendations than the standard list interface, which can be helpful for anyone who wants to explore their preferences. In addition, although the standard list interface is more familiar and easier for some participants, the introduction of pairwise comparisons can bring an exciting experience for an online shopper.

8.2 Future Work

With the completion of this thesis, there are many possible areas of improvement that can be explored considering that the work done in this thesis constitutes only a starting point for a wider research line.

As a general relation learner in DL, APARELL can be improved by expanding the refinement operator to allow the use of different logic operators (e.g. union and negation) and universal quantifiers to improve the accuracy. To do this, we need an evaluation on a different dataset. The expansion of APARELL could also include a method of handling mixed-type data in the learning process. A method to incorporate numerical features in the hypothesis search can be borrowed from FOIL-$\mathcal{DL}$ proposed by Lisi and Straccia [77].

Considering applications in the recommender system area, the use of probability and weighted rules to deal with the uncertainty problem could be an advantage. A recent study by Riguzzi et al. [108] can be used to improve the proposed approach. The implementation of the recommender system can then be generalised and tested using different purchase domains, such as housing, short-term accommodation or hotels, travel and books.

Finally, there is a large potential in the proposed ILP in DL method in this thesis, as well as using the logic-based methods in e-commerce research. The current research in the e-commerce area, specifically in recommender systems, is looking for an effective and richer representation to solve the problem with explainable systems to ensure a stronger level of engagement between the system and the users.

8.3 Final Remarks

Some of the data, software and source code developed as part of this thesis will be made available on the author's GitHub page at `https://github.com/nnqomariyah/`

Appendix A

Bradley-Terry Experiment

Table A.1: BT coefficients on car dataset [1]

Car ID	BT Coefficients									
	user1	user2	user3	user4	user5	user6	user7	user8	user9	user10
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	6.30×10^{1}	1.92	-1.14×10^{2}	6.74×10^{1}	2.14	3.94×10^{1}	4.48×10^{1}	1.00×10^{2}	8.85×10^{-1}	−4.39
3	5.20×10^{-16}	-9.96×10^{-1}	-4.57×10^{1}	-2.29×10^{1}	1.23	-2.72×10^{-16}	-4.55×10^{1}	6.05×10^{1}	−1.64	1.94×10^{-1}
4	-4.39×10^{1}	1.92	-2.33×10^{1}	-4.64×10^{1}	2.44	-2.93×10^{-16}	-6.90×10^{1}	4.09×10^{1}	−2.51	1.91×10^{1}
5	2.26×10^{-16}	2.09×10^{1}	2.37×10^{1}	2.26×10^{1}	2.28×10^{-15}	7.00×10^{-1}	-2.26×10^{1}	6.09×10^{-16}	5.47×10^{-16}	-7.58×10^{-1}
6	2.18×10^{1}	3.98×10^{1}	-6.82×10^{1}	1.35×10^{2}	1.23	2.33	1.13×10^{2}	2.14×10^{1}	-6.51×10^{-2}	−4.39
7	1.05×10^{2}	-9.96×10^{-1}	-1.37×10^{2}	1.12×10^{2}	2.71	5.81×10^{1}	6.73×10^{1}	8.01×10^{1}	1.99×10^{1}	−3.40
8	8.39×10^{1}	9.25×10^{-1}	-9.08×10^{1}	1.59×10^{2}	1.15×10^{-15}	2.13×10^{1}	1.36×10^{2}	1.10	3.87×10^{1}	-2.33×10^{1}
9	4.24×10^{1}	-2.09×10^{1}	-2.80×10^{-10}	4.50×10^{1}	1.21	2.22	2.24×10^{1}	1.21×10^{2}	-2.22×10^{1}	−1.47
10	-2.22×10^{1}	3.98×10^{1}	4.73×10^{1}	8.99×10^{1}	2.59	7.49×10^{-2}	8.99×10^{1}	1.10	−2.59	−2.50

Car ID	BT Coefficients									
	user11	user12	user13	user14	user15	user16	user17	user18	user19	user20
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	2.24×10^{1}	-4.20×10^{-1}	6.09×10^{-16}	-2.07×10^{1}	2.05×10^{2}	3.87	6.73×10^{1}	−1.01	6.73×10^{1}	6.01×10^{1}
3	6.73×10^{1}	-2.42×10^{-1}	-2.11×10^{1}	-2.18×10^{1}	6.90×10^{1}	2.30	-4.55×10^{1}	-4.05×10^{1}	-4.55×10^{1}	3.79×10^{1}
4	8.99×10^{1}	6.47×10^{-2}	-4.08×10^{1}	-2.18×10^{1}	4.64×10^{1}	1.15	-6.90×10^{1}	-6.01×10^{1}	-6.90×10^{1}	4.08×10^{1}
5	4.48×10^{1}	9.43×10^{-1}	-9.11×10^{-1}	1.96×10^{1}	2.35×10^{1}	3.16×10^{-15}	-2.26×10^{1}	−2.02	-2.26×10^{1}	1.87×10^{1}
6	-4.55×10^{1}	-2.42×10^{-1}	3.98×10^{1}	8.00×10^{1}	1.36×10^{2}	3.16	1.13×10^{2}	2.02×10^{1}	1.36×10^{2}	3.89×10^{1}
7	-2.26×10^{1}	-8.55×10^{-1}	9.11×10^{-1}	-2.07×10^{1}	1.82×10^{2}	3.87	8.99×10^{1}	-2.29×10^{-16}	4.48×10^{1}	3.98×10^{1}
8	-6.90×10^{1}	-4.20×10^{-1}	5.94×10^{1}	5.93×10^{1}	1.14×10^{2}	2.47	1.36×10^{2}	3.99×10^{1}	1.13×10^{2}	3.79×10^{1}
9	1.36×10^{2}	-4.20×10^{-1}	2.67×10^{-16}	-4.36×10^{1}	1.59×10^{2}	2.26×10^{1}	4.48×10^{1}	-2.15×10^{1}	2.24×10^{1}	7.88×10^{1}
10	1.13×10^{2}	-4.20×10^{-1}	2.07×10^{1}	3.93×10^{1}	9.14×10^{1}	4.71	2.24×10^{1}	−2.02	8.99×10^{1}	4.08×10^{1}

Table A.1: BT coefficients on car dataset [1] (cont.)

Car ID	BT Coefficients									
	user21	user22	user23	user24	user25	user26	user27	user28	user29	user30
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	-2.24×10^1	-5.61×10^{-1}	1.38×10^2	4.03×10^1	5.84×10^1	8.99×10^1	7.78×10^1	-8.10×10^1	-4.14×10^1	4.40×10^{-1}
3	6.75×10^1	-2.05×10^1	1.15×10^2	-3.96×10^1	1.90×10^1	-4.64×10^1	7.67×10^1	-7.99×10^1	-3.94×10^1	-4.23×10^{-1}
4	4.49×10^1	-3.93×10^1	1.15×10^2	-6.01×10^1	-1.96×10^1	-2.29×10^1	7.78×10^1	-7.99×10^1	-4.14×10^1	3.10×10^{-1}
5	2.24×10^1	1.72×10^{-1}	2.35×10^1	-1.97×10^1	3.78×10^1	2.26×10^1	3.83×10^1	2.06×10^1	-1.93×10^1	-8.63×10^{-1}
6	-6.79×10^1	1.86	6.90×10^1	4.14×10^1	6.00×10^1	1.36×10^2	1.96×10^1	-1.99×10^1	-1.93×10^1	9.18×10^{-16}
7	-4.50×10^1	1.86	1.61×10^2	4.03×10^1	5.92×10^1	6.74×10^1	7.67×10^1	-8.10×10^1	-1.93×10^1	-4.23×10^{-1}
8	-9.15×10^1	2.18×10^1	4.64×10^1	6.32×10^1	5.92×10^1	1.13×10^2	9.00×10^{-16}	-3.96×10^1	1.87×10^1	-8.63×10^{-1}
9	9.12×10^1	-1.18	1.85×10^2	1.96×10^1	5.76×10^1	4.50×10^1	5.69×10^1	-1.03×10^2	-3.94×10^1	-6.07×10^{-1}
10	9.12×10^1	-5.61×10^{-1}	9.15×10^1	4.14×10^1	5.84×10^1	1.36×10^2	8.16×10^{-15}	-5.92×10^1	-4.04×10^1	3.10×10^{-1}

Car ID	BT Coefficients									
	user31	user32	user33	user34	user35	user36	user37	user38	user39	user40
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	4.21×10^1	4.49×10^1	-6.29×10^1	8.87×10^{-1}	1.61×10^2	4.23×10^1	-3.93×10^1	8.99×10^1	1.12×10^2	6.05×10^1
3	1.90×10^1	-4.56×10^1	-6.29×10^1	-3.89×10^1	4.65×10^1	8.01×10^{-16}	2.21×10^1	-2.29×10^1	-4.64×10^1	-9.13×10^{-16}
4	2.14×10^1	-6.91×10^1	-4.11×10^1	-9.02×10^{-1}	7.00×10^1	-2.29×10^1	2.21×10^1	-4.64×10^1	-2.29×10^1	1.10
5	1.90×10^1	-2.26×10^1	4.16×10^1	-2.02×10^1	2.35×10^1	6.58×10^{-16}	2.07×10^1	2.26×10^1	2.26×10^1	1.10
6	2.31×10^1	1.13×10^2	2.07×10^1	2.76	9.33×10^1	1.26×10^2	-1.96×10^1	1.59×10^2	1.59×10^2	6.05×10^1
7	2.31×10^1	9.01×10^1	-8.58×10^1	-9.75×10^{-1}	1.16×10^2	8.34×10^1	-7.98×10^1	1.12×10^2	8.99×10^1	5.94×10^1
8	2.07×10^1	1.37×10^2	-2.05×10^1	2.76	7.00×10^1	1.04×10^2	-5.93×10^1	1.35×10^2	1.35×10^2	5.94×10^1
9	2.07×10^1	6.74×10^1	-6.29×10^1	1.72	1.85×10^2	2.18×10^1	2.24×10^1	4.50×10^1	4.50×10^1	2.09×10^1
10	2.14×10^1	2.25×10^1	6.31×10^1	2.27×10^1	1.38×10^2	6.28×10^1	2.09×10^1	6.74×10^1	6.74×10^1	3.96×10^1

Table A.1: BT coefficients on car dataset [1] (cont.)

Car ID	BT Coefficients									
	user41	user42	user43	user44	user45	user46	user47	user48	user49	user50
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	1.01	-6.74×10^{1}	2.05×10^{2}	2.24×10^{1}	-9.05×10^{1}	1.85×10^{2}	4.06×10^{1}	4.11×10^{1}	-1.00×10^{-15}	1.06
3	-1.65×10^{-16}	-9.03×10^{1}	4.64×10^{1}	1.14×10^{2}	-6.77×10^{1}	1.14×10^{2}	3.78×10^{1}	-1.10	3.79×10^{1}	-2.02×10^{1}
4	-1.99×10^{1}	-1.14×10^{2}	6.90×10^{1}	1.14×10^{2}	-4.51×10^{1}	1.37×10^{2}	3.71×10^{1}	-1.10	3.79×10^{1}	-3.99×10^{1}
5	2.02	2.24×10^{1}	2.35×10^{1}	4.49×10^{1}	2.28×10^{1}	4.64×10^{1}	1.77×10^{1}	2.10×10^{1}	1.90×10^{1}	5.28×10^{-1}
6	2.21×10^{1}	9.14×10^{1}	1.59×10^{2}	-4.55×10^{1}	4.59×10^{1}	1.37×10^{2}	3.85×10^{1}	7.98×10^{1}	-1.92	4.10×10^{1}
7	2.32×10^{1}	-2.24×10^{1}	1.82×10^{2}	-2.26×10^{1}	-1.38×10^{2}	1.61×10^{2}	4.06×10^{1}	2.10×10^{1}	-2.92	2.10×10^{1}
8	2.32×10^{1}	6.79×10^{1}	1.36×10^{2}	-6.90×10^{1}	-2.25×10^{1}	6.90×10^{1}	3.60×10^{1}	6.01×10^{1}	-2.28×10^{1}	4.10×10^{1}
9	2.21×10^{1}	-4.48×10^{1}	1.14×10^{2}	6.73×10^{1}	-1.14×10^{2}	2.35×10^{1}	3.96×10^{1}	-5.02×10^{-16}	-2.92	5.28×10^{-1}
10	2.02	4.50×10^{1}	9.14×10^{1}	9.00×10^{1}	6.96×10^{1}	9.15×10^{1}	3.60×10^{1}	2.10×10^{1}	-9.96×10^{-1}	4.14×10^{1}

Car ID	BT Coefficients									
	user51	user52	user53	user54	user55	user56	user57	user58	user59	user60
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	2.05×10^{2}	1.82×10^{2}	1.36×10^{2}	6.42×10^{-1}	6.42×10^{-1}	2.80×10^{-16}	-8.55×10^{1}	3.23×10^{-16}	3.70	-3.94×10^{1}
3	2.35×10^{1}	4.64×10^{1}	-2.26×10^{1}	-1.86	-1.86	-7.99×10^{1}	-2.19×10^{1}	1.67	2.73	-5.89×10^{1}
4	1.82×10^{2}	6.90×10^{1}	-6.92×10^{1}	-1.86	-1.86	-1.00×10^{2}	-2.19×10^{1}	-3.38×10^{-1}	2.27×10^{1}	-2.07×10^{1}
5	9.14×10^{1}	2.35×10^{1}	4.49×10^{1}	-1.22	-1.22	-2.07×10^{1}	4.26×10^{1}	-1.41	3.94×10^{-1}	9.74×10^{-1}
6	1.36×10^{2}	1.14×10^{2}	1.13×10^{2}	1.34	1.34	1.10	-4.37×10^{1}	3.90×10^{-1}	1.13	7.97×10^{-1}
7	1.14×10^{2}	1.59×10^{2}	8.99×10^{1}	2.18	2.18	1.10	-1.07×10^{2}	2.41	1.13	-5.89×10^{1}
8	4.64×10^{1}	9.14×10^{1}	6.73×10^{1}	1.34	1.34	2.29×10^{1}	-6.44×10^{1}	6.68×10^{-1}	-1.93×10^{1}	-9.74×10^{-1}
9	6.90×10^{1}	2.05×10^{2}	-4.55×10^{1}	-1.22	-1.22	-5.99×10^{1}	-2.19×10^{1}	2.30	-1.32×10^{-15}	-7.93×10^{1}
10	1.59×10^{2}	1.36×10^{2}	2.24×10^{1}	-1.86	-1.86	-4.03×10^{1}	2.10×10^{1}	3.23	3.70	-7.97×10^{-1}

Appendix B

DL-Learner Experiment

Table B.1: DL-Learner experiment result

user ID	the best carID	DL-Learner result with 100% accuracy
user1	car7	Automatic and Hybrid and MediumCar and Sedan
user2	car6,car10	Automatic and MediumCar and Suv
user3	car1	Manual and SmallCar
user4	car8	Automatic and SmallCar
user5	car7	Automatic and Hybrid and MediumCar and Sedan
user6	car7	Automatic and Hybrid and MediumCar and Sedan
user7	car8	Automatic and SmallCar
user8	car9	Automatic and NonHybrid and Sedan
user9	car8	Automatic and SmallCar
user10	car4	LargeCar and Manual
user11	car9	Automatic and NonHybrid and Sedan
user12	car5	Manual and MediumCar and Suv
user13	car8	Automatic and SmallCar
user14	car6	Automatic and Hybrid and MediumCar and Suv
user15	car2	Automatic and LargeCar
user16	car9	Automatic and NonHybrid and Sedan
user17	car8	Automatic and SmallCar
user18	car8	Automatic and SmallCar
user19	car6	Automatic and Hybrid and MediumCar and Suv
user20	car9	Automatic and NonHybrid and Sedan

Table B.2: DL-Learner experiment result (cont.)

user ID	the best carID	DL-Learner result with 100% accuracy
user21	car10,car9	Automatic and NonHybrid
user22	car8	Automatic and SmallCar
user23	car9	Automatic and NonHybrid and Sedan
user24	car8	Automatic and SmallCar
user25	car6	Automatic and Hybrid and MediumCar and Suv
user26	car10,6	Automatic and MediumCar and Suv
user27	car2,car4	LargeCar
user28	car5	Manual and MediumCar and Suv
user29	car8	Automatic and SmallCar
user30	car2	Automatic and NonHybrid and Sedan
user31	car2	Automatic and NonHybrid and Sedan
user32	car8	Automatic and SmallCar
user33	car10	Automatic and NonHybrid and Suv
user34	car10	Automatic and NonHybrid and Suv
user35	car9	Automatic and NonHybrid and Sedan
user36	car6	Automatic and Hybrid and MediumCar and Suv
user37	car9	Automatic and NonHybrid and Sedan
user38	car6	Automatic and Hybrid and MediumCar and Suv
user39	car6	Automatic and Hybrid and MediumCar and Suv
user40	car2,car6	Hybrid and (LargeCar or (MediumCar and Suv))
user41	car7,car8	Hybrid and (SmallCar or (MediumCar and Sedan))
user42	car6	Automatic and Hybrid and MediumCar and Suv
user43	car2	Automatic and NonHybrid and Suv
user44	car3,car4	Manual and Sedan
user45	car10	Automatic and NonHybrid and Suv
user46	car2	Automatic and NonHybrid and Suv
user47	car2,car7	Automatic and Hybrid and Sedan
user48	car6	Automatic and Hybrid and MediumCar and Suv
user49	car3,car4	Manual and Sedan
user50	car10	Automatic and NonHybrid and Suv
user51	car2	Automatic and NonHybrid and Suv
user52	car9	Automatic and NonHybrid and Sedan
user53	car2	Automatic and NonHybrid and Suv
user54	car7	Automatic and Hybrid and MediumCar and Sedan
user55	car7	Automatic and Hybrid and MediumCar and Sedan
user56	car8	Automatic and SmallCar
user57	car5	Manual and MediumCar and Suv
user58	car10	Automatic and NonHybrid and Suv
user59	car4	LargeCar and Manual
user60	car5	Manual and MediumCar and Suv

Appendix C

Aleph Background Knowledge

C.1 Aleph settings for car dataset

```
% specify the hypothesis language
:- modeh(1,bt(+car,+car)).
:- modeb(1,carfuel(+car,#fuelconsumption,+car,#fuelconsumption)).
:- modeb(1,carbodytype(+car,#bodytype,+car,#bodytype)).
:- modeb(1,cartransmission(+car,#transmission,+car,#transmission)).
:- modeb(1,carenginegreaterthan(+car,+car,-float,-float)).
:- modeb(1,carenginelessthan(+car,+car,-float,-float)).

:- determination(bt/2,carfuel/4).
:- determination(bt/2,carbodytype/4).
:- determination(bt/2,cartransmission/4).
:- determination(bt/2,carenginegreaterthan/4).
:- determination(bt/2,carenginelessthan/4).

% type of categorical attributes
fuelconsumption(hybrid).
fuelconsumption(nonhybrid).
transmission(automatic).
transmission(manual).
bodytype(sedan).
bodytype(suv).

% type of individual car
car(car1).
car(car2).
car(car3).
car(car4).
car(car5).
car(car6).
car(car7).
car(car8).
car(car9).
car(car10).
```

Figure C.1: Aleph's background knowledge for car dataset in setting 3

```
% relations of each car with its attribute values
hasbodytype(car1,suv).
hasbodytype(car2,sedan).
hasbodytype(car3,sedan).
hasbodytype(car4,sedan).
hasbodytype(car5,suv).
hasbodytype(car6,suv).
hasbodytype(car7,sedan).
hasbodytype(car8,suv).
hasbodytype(car9,sedan).
hasbodytype(car10,suv).

hasfuelcons(car1,nonhybrid).
hasfuelcons(car2,hybrid).
hasfuelcons(car3,nonhybrid).
hasfuelcons(car4,nonhybrid).
hasfuelcons(car5,nonhybrid).
hasfuelcons(car6,hybrid).
hasfuelcons(car7,hybrid).
hasfuelcons(car8,hybrid).
hasfuelcons(car9,nonhybrid).
hasfuelcons(car10,nonhybrid).

hastransmission(car1,manual).
hastransmission(car2,automatic).
hastransmission(car3,manual).
hastransmission(car4,manual).
hastransmission(car5,manual).
hastransmission(car6,automatic).
hastransmission(car7,automatic).
hastransmission(car8,automatic).
hastransmission(car9,automatic).
hastransmission(car10,automatic).


hasenginesize(car1, 2.5).
hasenginesize(car8, 2.5).
hasenginesize(car10, 4.5).
hasenginesize(car5, 3.5).
hasenginesize(car7, 3.5).
hasenginesize(car6, 3.5).
hasenginesize(car9, 3.5).
hasenginesize(car3, 4.5).
hasenginesize(car2, 5.5).
hasenginesize(car4, 6.2).
```

Figure C.1: Aleph's background knowledge for car dataset in setting 3 (cont.)

```
% specify different rules for categorical attributes
carfuel(A,X,B,Y):- hasfuelcons(A,X), car(A), car(B), hasfuelcons(B,Y), X\=Y.
carbodytype(A,X,B,Y):- hasbodytype(A,X), car(A), car(B), hasbodytype(B,Y), X\=Y.
cartransmission(A,X,B,Y):- hastransmission(A,X), car(A), car(B),
   hastransmission(B,Y), X\=Y.

% specify rules for numerical attribute
carenginegreaterthan(A,B,X,Y):- car(A), car(B),
   hasenginesize(A,X), hasenginesize(B,Y), X>Y.
carenginelessthan(A,B,X,Y):- car(A), car(B), hasenginesize(A,X),
   hasenginesize(B,Y), X<Y.
```

Figure C.1: Aleph's background knowledge for car dataset in setting 3 (cont.)

C.2 Aleph setting for sushi dataset

```
% hypothesis language
:- modeh(1,bt(+sushi,+sushi)).
:- modeb(1,sushistyle(+sushi,#style,+sushi,#style)).
:- modeb(1,sushimajor(+sushi,#major,+sushi,#major)).
:- modeb(1,sushiminor(+sushi,#minor,+sushi,#minor)).
:- modeb(1,sushiheavinesslessthan(+sushi,+sushi,-float,-float)).
:- modeb(1,sushiheavinessgreaterthan(+sushi,+sushi,-float,-float)).
:- modeb(1,sushiconsumedlessthan(+sushi,+sushi,-float,-float)).
:- modeb(1,sushiconsumedgreaterthan(+sushi,+sushi,-float,-float)).
:- modeb(1,sushipricelessthan(+sushi,+sushi,-float,-float)).
:- modeb(1,sushipricegreaterthan(+sushi,+sushi,-float,-float)).
:- modeb(1,sushisoldlessthan(+sushi,+sushi,-float,-float)).
:- modeb(1,sushisoldgreaterthan(+sushi,+sushi,-float,-float)).

:- determination(bt/2,sushistyle/4).
:- determination(bt/2,sushimajor/4).
:- determination(bt/2,sushiminor/4).
:- determination(bt/2,sushiheavinesslessthan/4).
:- determination(bt/2,sushiheavinessgreaterthan/4).
:- determination(bt/2,sushiconsumedlessthan/4).
:- determination(bt/2,sushiconsumedgreaterthan/4).
:- determination(bt/2,sushipricelessthan/4).
:- determination(bt/2,sushisoldlessthan/4).
:- determination(bt/2,sushipricegreaterthan/4).
:- determination(bt/2,sushisoldgreaterthan/4).

% type for categorical attributes
style(maki).
style(notmaki).
major(seafood).
major(notseafood).
minor(aomono).
minor(tare).
minor(squid).
minor(shrimp).
minor(roe).
minor(otherseafood).
minor(egg).
minor(veggie).

% type of individual sushi
sushi(sushi0).
sushi(sushi1).
sushi(sushi2).
sushi(sushi3).
sushi(sushi4).
sushi(sushi5).
sushi(sushi6).
sushi(sushi7).
sushi(sushi8).
sushi(sushi9).
```

Figure C.2: Aleph's background knowledge for sushi dataset in setting 3

```
%relations of each sushi with its attribute values
hasstyle(sushi0,notmaki).
hasstyle(sushi1,notmaki).
hasstyle(sushi2,notmaki).
hasstyle(sushi3,notmaki).
hasstyle(sushi4,notmaki).
hasstyle(sushi5,notmaki).
hasstyle(sushi6,notmaki).
hasstyle(sushi7,notmaki).
hasstyle(sushi8,maki).
hasstyle(sushi9,maki).


hasmajor(sushi0,seafood).
hasmajor(sushi1,seafood).
hasmajor(sushi2,seafood).
hasmajor(sushi3,seafood).
hasmajor(sushi4,seafood).
hasmajor(sushi5,seafood).
hasmajor(sushi6,notseafood).
hasmajor(sushi7,seafood).
hasmajor(sushi8,seafood).
hasmajor(sushi9,notseafood).

hasminor(sushi0,shrimp).
hasminor(sushi1,tare).
hasminor(sushi2,akami).
hasminor(sushi3,squid).
hasminor(sushi4,otherseafood).
hasminor(sushi5,roe).
hasminor(sushi6,egg).
hasminor(sushi7,akami).
hasminor(sushi8,akami).
hasminor(sushi9,veggie).

hasheaviness(sushi0,2.728978008).
hasheaviness(sushi1,0.926384365).
hasheaviness(sushi2,1.769559033).
hasheaviness(sushi3,2.688400824).
hasheaviness(sushi4,0.813043478).
hasheaviness(sushi5,1.264872521).
hasheaviness(sushi6,2.368070953).
hasheaviness(sushi7,0.551854656).
hasheaviness(sushi8,2.247133758).
hasheaviness(sushi9,3.73054755).
```

Figure C.2: Aleph's background knowledge for sushi dataset in setting 3 (cont.)

```
hasconsumed(sushi0,2.138421734).
hasconsumed(sushi1,1.990228013).
hasconsumed(sushi2,2.348506401).
hasconsumed(sushi3,2.043239533).
hasconsumed(sushi4,1.643478261).
hasconsumed(sushi5,1.979461756).
hasconsumed(sushi6,1.866223208).
hasconsumed(sushi7,2.057532173).
hasconsumed(sushi8,1.878980892).
hasconsumed(sushi9,1.456772334).

hasprice(sushi0,1.838419913).
hasprice(sushi1,1.992458678).
hasprice(sushi2,1.874724518).
hasprice(sushi3,1.515151515).
hasprice(sushi4,3.28728191).
hasprice(sushi5,2.695362718).
hasprice(sushi6,1.032467532).
hasprice(sushi7,4.485454545).
hasprice(sushi8,1.57983683).
hasprice(sushi9,1.02).

hassold(sushi0,0.84).
hassold(sushi1,0.88).
hassold(sushi2,0.88).
hassold(sushi3,0.92).
hassold(sushi4,0.88).
hassold(sushi5,0.88).
hassold(sushi6,0.84).
hassold(sushi7,0.8).
hassold(sushi8,0.44).
hassold(sushi9,0.4).

% specify different rules for categorical attributes
sushistyle(A,X,B,Y):- hasstyle(A,X),sushi(A),sushi(B),hasstyle(B,Y),X\=Y .
sushimajor(A,X,B,Y):- hasmajor(A,X),sushi(A),sushi(B),hasmajor(B,Y),X\=Y .
sushiminor(A,X,B,Y):- hasminor(A,X),sushi(A),sushi(B),hasminor(B,Y),X\=Y .

% specify rules for numerical attribute
sushiheavinesslessthan(A,B,X,Y):- sushi(A), sushi(B), hasheaviness(A,X),
  hasheaviness(B,Y), X<Y .
sushiheavinessgreaterthan(A,B,X,Y):- sushi(A), sushi(B), hasheaviness(A,X),
  hasheaviness(B,Y), X>Y .
sushiconsumedlessthan(A,B,X,Y):- sushi(A), sushi(B), hasconsumed(A,X),
  hasconsumed(B,Y), X<Y .
sushiconsumedgreaterthan(A,B,X,Y):- sushi(A), sushi(B), hasconsumed(A,X),
  hasconsumed(B,Y), X>Y .
sushipricelessthan(A,B,X,Y):- sushi(A), sushi(B), hasprice(A,X),
  hasprice(B,Y), X<Y .
sushipricegreaterthan(A,B,X,Y):- sushi(A), sushi(B), hasprice(A,X),
  hasprice(B,Y), X>Y .
sushisoldlessthan(A,B,X,Y):- sushi(A), sushi(B), hassold(A,X),
  hassold(B,Y), X<Y .
sushisoldgreaterthan(A,B,X,Y):- sushi(A), sushi(B), hassold(A,X), hassold(B,Y), X>Y .
```

Figure C.2: Aleph's background knowledge for sushi dataset in setting 3 (cont.)

Appendix D

APARELL User Manual

D.1 Introduction

APARELL stands for Active PAirwise RELation Learner, is a supervised machine learning software which can learn a relation in OWL/Description Logics. It implements the framework explains in this thesis.The guidance on how to use the software is provided in this document.

D.2 Getting started

APARELL is implemented in Java and can be used in almost all platform. It requires Java version 8 or higher. It can be downloaded in the software section in here[1] as a zip file which contains the main software in a compiled "aparell.jar" file; an example of configuration file; and a set of working examples in a folder 'dataset'. Before running the software, we need to prepare these three basic things, which are:

1. a relation name to be learned,
   for example, we want to learn "betterthan" relationship,
2. a set of positive and negative examples,
   this set will contain examples in form of "individual1,individual2" in which individual1 is related to individual2 through a relation in point 1, and
3. a class hierarchy in the ontology,

[1]https://www-users.cs.york.ac.uk/∼nnq/

the class hierarchy as knowledge source is necessary to be specified as the learner uses it as a base to build the hypotheses.

To run the software, it requires all necessary parameters to be specified in a configuration file stored as a text file. We will explain the parameter configuration in the next section. The software can be simply run from a terminal as:

```
$ java -jar aparell.jar \path\to\configuration\file.txt
```

D.3 Configuration file

There are five required parameters need to be specified in this file. It is required to provide the full path to the specified file.

- **kbfile** or **tripledb_server**
  this parameter is used to specify a knowledge source. APARELL can read both an OWL file and a triple database server. We need to specify one at a time, whether we want to use an OWL file or a remote/local triple store database server. All formats supporting by OWL API[2] can be used in here.
- **prefix**
  prefix is used to reference IRI (Internationalized Resource Identifier) of the ontology.
- **relation**
  currently, APARELL can be used to learn one relation at a time. We need to specify the relation name in the ontology that we want to learn.
- **pos_example**
  we need to specify the positive examples in a text file using a format given in the next section. The possible value is the full path to the positive training file.
- **neg_example**
  we also need to specify the negative examples in a separated text file. The possible value is the full path to the negative training file.

There are three optional parameters can be specified in a configuration file, they are:

[2]http://owlapi.sourceforge.net

- **pos_test**
  we can specify the positive examples test file if we want to evaluate the learner model's accuracy. The possible value is the full path to the positive test file.

- **neg_test**
  if the positive test is specified, the negative test examples is also need to be specified. The possible value is the full path to the negative test file.

- **literal_depth_limit**
  this setting is used to limit the depth of the search with the default value is 4. Currently, our system can handle the search until the depth value=5. We can specify any positive integer number between 2 and 5.

- **include_inferred_class** this setting is used to specify whether we want to include the inferred class in the search or not. The possible value is: YES or NO, with the default value of this setting is NO.

D.4 An example

An example of a complete configuration file is shown in Figure D.1a. All the examples, i.e. training positive, training negative, testing positive and testing negative, is using the format shown in Figure D.1b as the pairs of `individual - individual` separated by a comma. All the input files is stored as textfiles.

D.5 Software architecture

APARELL is built by using two types of Java ontology handling library: OWL API for processing an OWL file and RDF4J for processing an ontology from an RDF database server. For the ability to read databases from a remote server, this software works well with GraphDB [3].

[3]https://ontotext.com/products/graphdb/

```
% knowledge base resource
% tripledb_server=http://localhost:7200/repositories/mycarsontology
kbfile = car.owl
prefix = http://www.mycars.org/ontology#
relation = betterthan

% training examples
pos_examples = trainpos.txt
neg_examples = trainneg.txt

% test examples (optional)
pos_test = testpos.txt
neg_test = testneg.txt

% parameter settings (optional)
literal_depth_limit=3
include_inferred_class=no
```

(a) A sample of a configuration file

```
car7,car6
car7,car8
car3,car1
car2,car3
```

(b) A sample of a positive examples file

Figure D.1: APARELL configurations

Appendix E

User Consent Form

UNIVERSITY of York

INFORMED CONSENT FORM

Title of Project:	Utilizing Pairwise Comparisons and Active Learning for User Preferences Elicitation in Recommender System
Principal Investigator:	Nunung Nurul Qomariyah

We invite you to take part in this research study at Computer Science Department University of York which aims to learn user preferences regarding the purchase of used cars. Please note that this study cannot be seen as providing any practical advice on this matter and the University of York cannot be held liable for the consequence of any purchase made by yourself.

Before you participate in the design evaluation study, please read all the sections in this consent form, printing your name and then sign at the end.

Section 1. VOLUNTARY PARTICIPATION

Taking part in this research study is voluntary. You do not have to participate in this research. If you choose to take part, you have the right to stop at any time. If you decide not to participate or if you decide to stop taking part in the research at a later date, there will be no penalty or loss of benefits to which you are otherwise entitled.

Your investigator may take you out of the research study without your permission. Some possible reasons for this are: you did not follow the study instructions or your data seemed to be invalid.

Section 2. PROCEDURES

Your investigator will guide you through the online system. You will be asked to answer all the questions. We do not store any personal data related to your email account.

You are given two tasks to finish as below:

1. Please find up to three cars suitable for daily commute between home and office (within the distance around 5-7 miles each way).
2. Please find up to three cars suitable for weekend shopping for a household of 4 people (within the distance around 3-5 miles each way).

You have to finish each of the tasks by using one of these interfaces, which are:

- Pairwise interface
- Standard list interface

There are three questionnaires you need to fill in, they are:

- Post-stage questionnaire for pairwise interface. After you finish working with pairwise interface, you will be asked to complete a post-stage questionnaire to evaluate the recommender system.
- Post-stage questionnaire for standard list interface. You also need to complete the post-stage questionnaire for evaluating this interface.
- After you complete your assessment with both interface, you will be asked to fill in the final questionnaire to compare between those two interfaces.

The principal investigator will help you with any inquiry regarding this experiment. You can ask any question or seek further clarification about this study if you need to.

Section 3. STATEMENT OF CONFIDENTIALITY

All information collected is confidential and anonymous. Any information from the study will only be made public in an anonymous group format, so that individuals will not be identifiable.

Section 4 COMPENSATION FOR PARTICIPATION

You will be given an Amazon voucher worth £5 to compensate you for participating in this study.

By signing below, you indicate that you have read the information written above, agree to participate in this study and give permission for the study to be recorded.

______________________	_____________	_________________________
Signature of Participant	Date	Printed Name

Appendix F

User Study Manual

UNIVERSITY *of York*

USER MANUAL

Title of Project:	Learning User Preferences for Recommender Systems from Pairwise Comparisons using Inductive Logic Programming in Description Logics
Principal Investigator:	Nunung Nurul Qomariyah

In this experiment, you are given two tasks to finish as below:

1. Please find up to three cars suitable for daily commute between home and office (within the distance around 5-7 miles each way).
2. Please find up to three cars suitable for weekend shopping for a household of 4 people (within the distance around 3-5 miles each way).

You have to finish each of the tasks by using one of these interfaces, which are:

- Pairwise interface.
- Standard list interface

We provide the screenshot and guidance in the next page.

There are three questionnaires you need to fill in, they are:

1. Post-stage questionnaire for pairwise interface. After you finish working with pairwise interface, you will be asked to complete a post-stage questionnaire to evaluate the recommender system.
2. Post-stage questionnaire for standard list interface. You also need to complete the post-stage questionnaire for evaluating this interface.
3. After you complete your assessment with both interfaces, you will be asked to fill in the final questionnaire to compare between those two interfaces.

The principal investigator will help you with any inquiry regarding this experiment. You can ask any question or seek further clarification about this study if you need to.

PAIRWISE INTERFACE

There are three steps to use this interface:

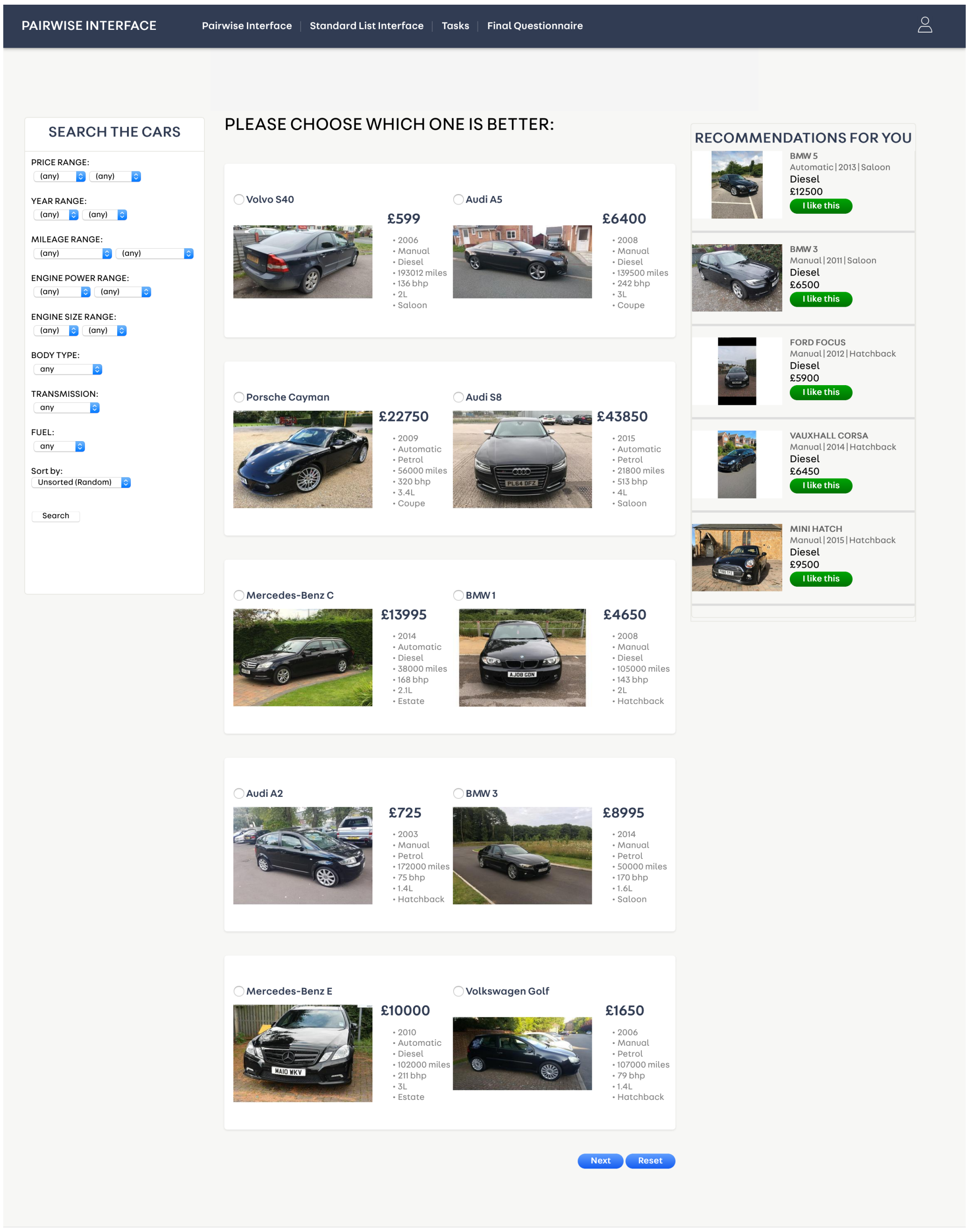
1. Annotating the pairs.
   In this interface, you can search the cars using filtering in the left side, we suggest you to not sort the items in order to allow you to get more diverse choices. Please note that we provide the image of the cars for illustration only. Some of the items don't have an image available. **Please don't use the image solely** for guiding you through the search. We want you to annotate at least **10 pairs** to complete this part.

   You can start to choose one of items you like in each pair (please compare the items in rows). For examples, in the first row you compare between Volkswagen CC and Volkswagen Passat. Please consider all the attribute values carefully. You can pass the pairs if none of the items match your preferences. After you finish choosing in pairs, please click blue button "Save" in the bottom of the page. You can continue to annotate the pairs as many as you like.

2. Recommendations.
   We provide a set of recommendation for you in the right side, if you like any of them, please click the green button "I like this".

3. Submitting your answer and result of your search.
   After you finish selecting the cars you like, you can submit your answer through YOUR TOP PICKS page (click the green button in your top right).

   In the TOP PICKS page, you will be asked to choose up to three cars that suitable for you, and then click the SUBMIT YOUR ANSWER. You can also see alternatives of the recommendation by click the button "See other alternatives here". In this step, you can always go back and annotate more pairs if you want to see different results.



PAIRWISE INTERFACE | Pairwise Interface | Standard List Interface | Tasks | Final Questionnaire

SEARCH THE CARS

PRICE RANGE: (any) (any)

YEAR RANGE: (any) (any)

MILEAGE RANGE: (any) (any)

ENGINE POWER RANGE: (any) (any)

ENGINE SIZE RANGE: (any) (any)

BODY TYPE: any

TRANSMISSION: any

FUEL: any

Sort by: Unsorted (Random)

Search

PLEASE CHOOSE WHICH ONE IS BETTER:

Volvo S40 £599 • 2006 • Manual • Diesel • 193012 miles • 136 bhp • 2L • Saloon	Audi A5 £6400 • 2008 • Manual • Diesel • 139500 miles • 242 bhp • 3L • Coupe
Porsche Cayman £22750 • 2009 • Automatic • Petrol • 56000 miles • 320 bhp • 3.4L • Coupe	Audi S8 £43850 • 2015 • Automatic • Petrol • 21800 miles • 513 bhp • 4L • Saloon
Mercedes-Benz C £13995 • 2014 • Automatic • Diesel • 38000 miles • 168 bhp • 2.1L • Estate	BMW 1 £4650 • 2008 • Manual • Diesel • 105000 miles • 143 bhp • 2L • Hatchback
Audi A2 £725 • 2003 • Manual • Petrol • 172000 miles • 75 bhp • 1.4L • Hatchback	BMW 3 £8995 • 2014 • Manual • Petrol • 50000 miles • 170 bhp • 1.6L • Saloon
Mercedes-Benz E £10000 • 2010 • Automatic • Diesel • 102000 miles • 211 bhp • 3L • Estate	Volkswagen Golf £1650 • 2006 • Manual • Petrol • 107000 miles • 79 bhp • 1.4L • Hatchback

Next Reset

RECOMMENDATIONS FOR YOU

BMW 5
Automatic | 2013 | Saloon
Diesel
£12500
I like this

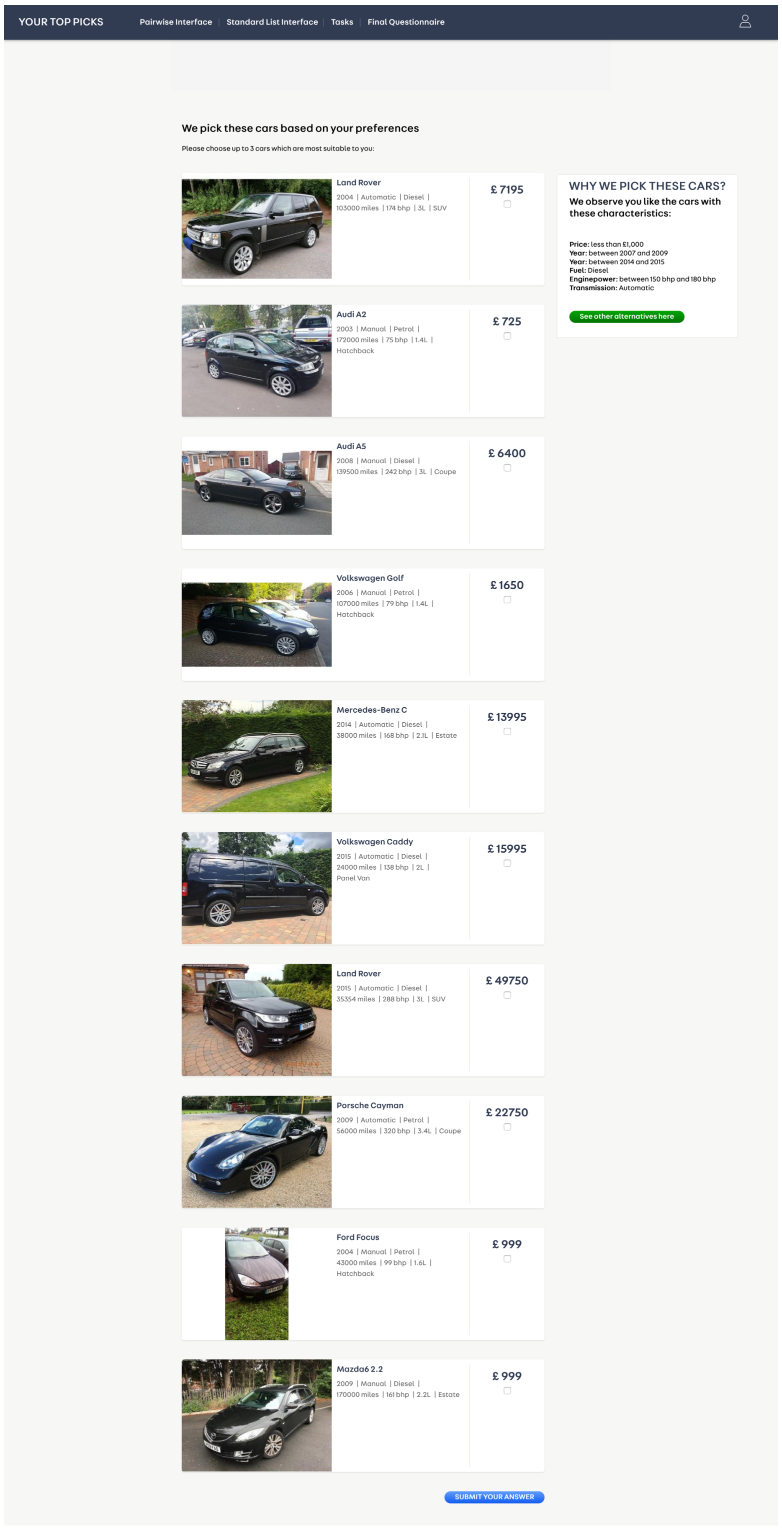
BMW 3
Manual | 2011 | Saloon
Diesel
£6500
I like this

FORD FOCUS
Manual | 2012 | Hatchback
Diesel
£5900
I like this

VAUXHALL CORSA
Manual | 2014 | Hatchback
Diesel
£6450
I like this

MINI HATCH
Manual | 2015 | Hatchback
Diesel
£9500
I like this

Figure 1. Pairwise interface



We pick these cars based on your preferences

Please choose up to 3 cars which are most suitable to you:

Land Rover
2004 | Automatic | Diesel |
103000 miles | 174 bhp | 3L | SUV
£ 7195

Audi A2
2003 | Manual | Petrol |
172000 miles | 75 bhp | 1.4L |
Hatchback
£ 725

Audi A5
2008 | Manual | Diesel |
139500 miles | 242 bhp | 3L | Coupe
£ 6400

Volkswagen Golf
2006 | Manual | Petrol |
107000 miles | 79 bhp | 1.4L |
Hatchback
£ 1650

Mercedes-Benz C
2014 | Automatic | Diesel |
38000 miles | 168 bhp | 2.1L | Estate
£ 13995

Volkswagen Caddy
2015 | Automatic | Diesel |
24000 miles | 138 bhp | 2L |
Panel Van
£ 15995

Land Rover
2015 | Automatic | Diesel |
35354 miles | 288 bhp | 3L | SUV
£ 49750

Porsche Cayman
2009 | Automatic | Petrol |
56000 miles | 320 bhp | 3.4L | Coupe
£ 22750

Ford Focus
2004 | Manual | Petrol |
43000 miles | 99 bhp | 1.6L |
Hatchback
£ 999

Mazda6 2.2
2009 | Manual | Diesel |
170000 miles | 161 bhp | 2.2L | Estate
£ 999

WHY WE PICK THESE CARS?

We observe you like the cars with these characteristics:

Price: less than £1,000
Year: between 2007 and 2009
Year: between 2014 and 2015
Fuel: Diesel
Enginepower: between 150 bhp and 180 bhp
Transmission: Automatic

Figure 2. Top picks page

STANDARD LIST INTERFACE

There are three steps to use this interface:

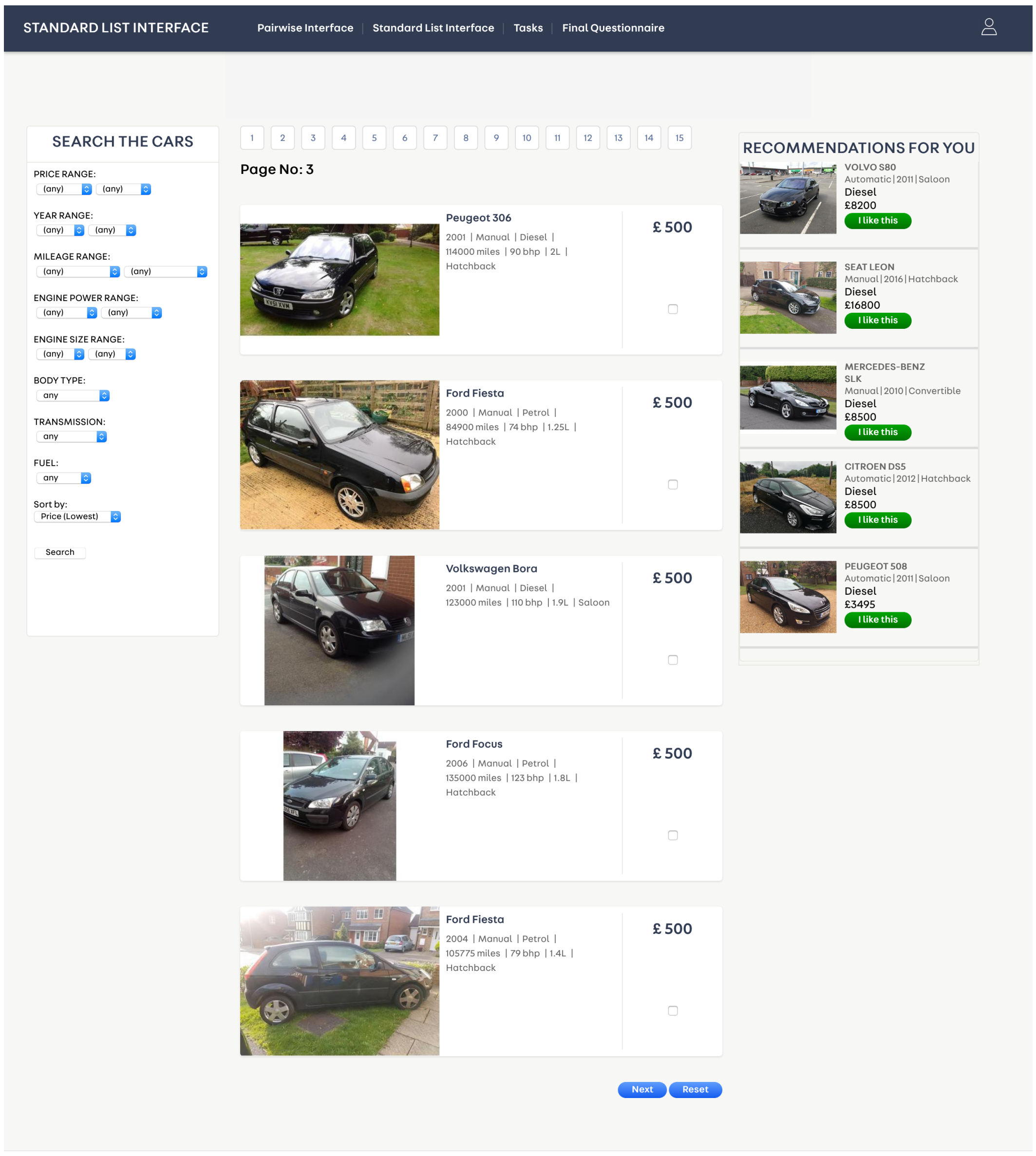
1. Selecting the items.
   In this interface, you can search the cars using filtering in the left side. Please note that we provide the image of the cars for illustration only. Some of the items don't have an image available. **Please don't use the image solely** for guiding you through the search.

   You can start to choose one of items you like in each page. You can pass the annotation on a page if none of the items match your preferences. After you finish choosing your preferred items, please click blue button "Save" in the bottom of the page. You can continue to select the items as many as you like.

2. Recommendations.
   We provide a set of recommendation for you in the right side, if you like any of them, please click the green button "I like this".

3. Submitting your answer and result of your search.
   After you finish selecting the cars you like, you can submit your answer through YOUR TOP PICKS page (click the green button in your top right).

   In the TOP PICKS page, you will be asked to choose up to three cars that suitable for you, and then click the SUBMIT YOUR ANSWER.


STANDARD LIST INTERFACE
Pairwise Interface | Standard List Interface | Tasks | Final Questionnaire

SEARCH THE CARS

PRICE RANGE: (any) (any)

YEAR RANGE: (any) (any)

MILEAGE RANGE: (any) (any)

ENGINE POWER RANGE: (any) (any)

ENGINE SIZE RANGE: (any) (any)

BODY TYPE: any

TRANSMISSION: any

FUEL: any

Sort by: Price (Lowest)

Search
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Page No: 3

Car	Price
Peugeot 306 2001 | Manual | Diesel | 114000 miles | 90 bhp | 2L | Hatchback	£ 500
Ford Fiesta 2000 | Manual | Petrol | 84900 miles | 74 bhp | 1.25L | Hatchback	£ 500
Volkswagen Bora 2001 | Manual | Diesel | 123000 miles | 110 bhp | 1.9L | Saloon	£ 500
Ford Focus 2006 | Manual | Petrol | 135000 miles | 123 bhp | 1.8L | Hatchback	£ 500
Ford Fiesta 2004 | Manual | Petrol | 105775 miles | 79 bhp | 1.4L | Hatchback	£ 500

RECOMMENDATIONS FOR YOU

VOLVO S80
Automatic | 2011 | Saloon
Diesel
£8200

SEAT LEON
Manual | 2016 | Hatchback
Diesel
£16800

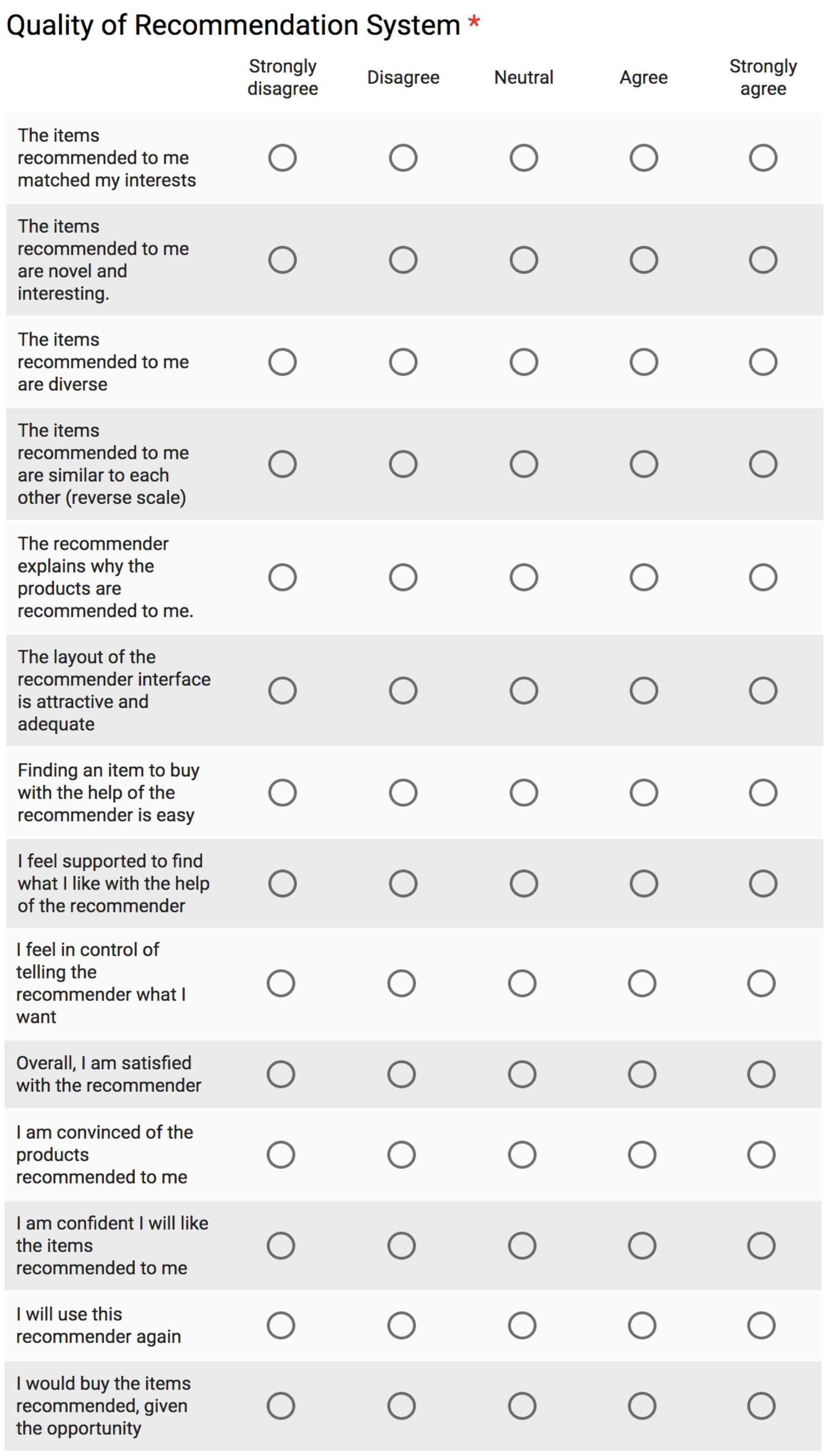
MERCEDES-BENZ SLK
Manual | 2010 | Convertible
Diesel
£8500

CITROEN DS5
Automatic | 2012 | Hatchback
Diesel
£8500

PEUGEOT 508
Automatic | 2011 | Saloon
Diesel
£3495

Figure 3. Standard list interface

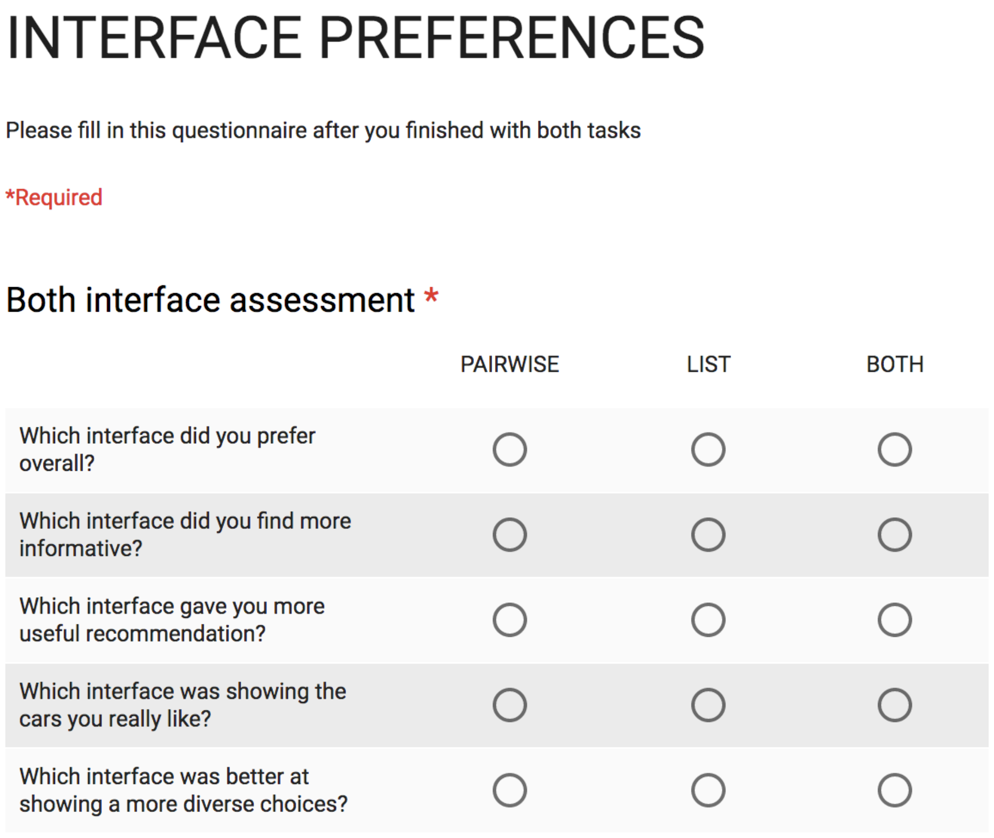
POST-STAGE QUESTIONNAIRE



Quality of Recommendation System *

	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
The items recommended to me matched my interests	○	○	○	○	○
The items recommended to me are novel and interesting.	○	○	○	○	○
The items recommended to me are diverse	○	○	○	○	○
The items recommended to me are similar to each other (reverse scale)	○	○	○	○	○
The recommender explains why the products are recommended to me.	○	○	○	○	○
The layout of the recommender interface is attractive and adequate	○	○	○	○	○
Finding an item to buy with the help of the recommender is easy	○	○	○	○	○
I feel supported to find what I like with the help of the recommender	○	○	○	○	○
I feel in control of telling the recommender what I want	○	○	○	○	○
Overall, I am satisfied with the recommender	○	○	○	○	○
I am convinced of the products recommended to me	○	○	○	○	○
I am confident I will like the items recommended to me	○	○	○	○	○
I will use this recommender again	○	○	○	○	○
I would buy the items recommended, given the opportunity	○	○	○	○	○

INTERFACE PREFERENCE



INTERFACE PREFERENCES

Please fill in this questionnaire after you finished with both tasks

*Required

Both interface assessment *

	PAIRWISE	LIST	BOTH
Which interface did you prefer overall?	○	○	○
Which interface did you find more informative?	○	○	○
Which interface gave you more useful recommendation?	○	○	○
Which interface was showing the cars you really like?	○	○	○
Which interface was better at showing a more diverse choices?	○	○	○

Appendix G

Questionnaire Result Details

G.1 Post-stage questionnaire results

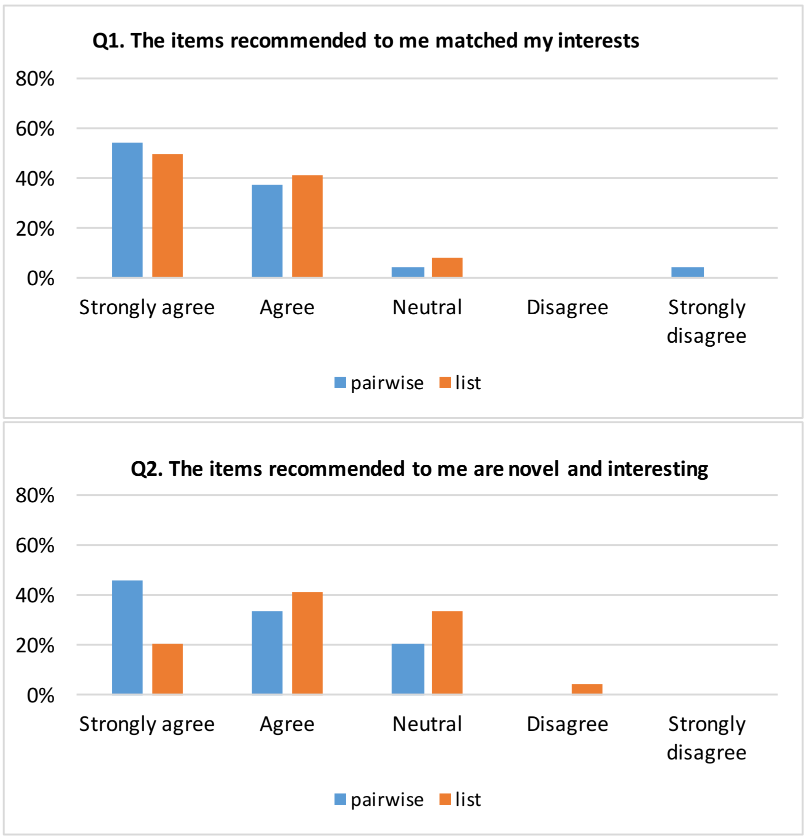
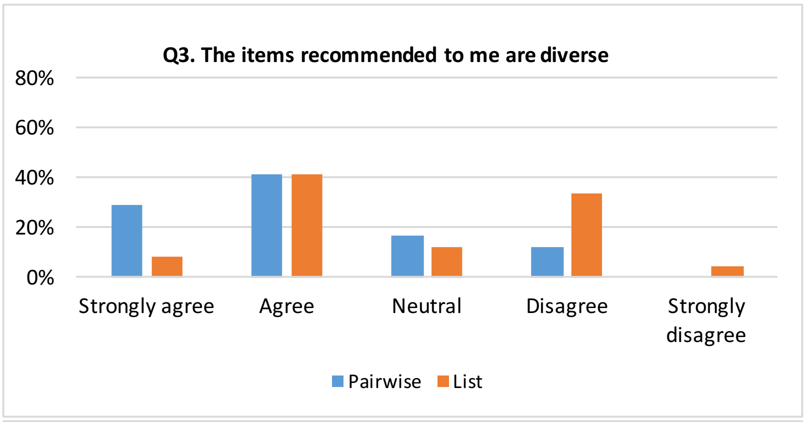
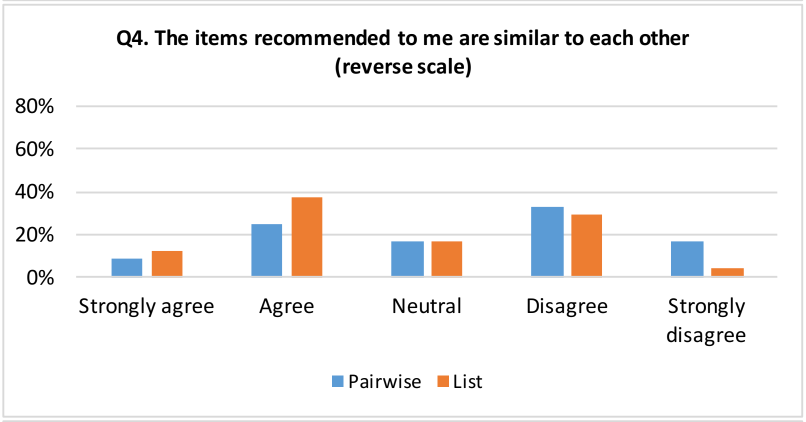
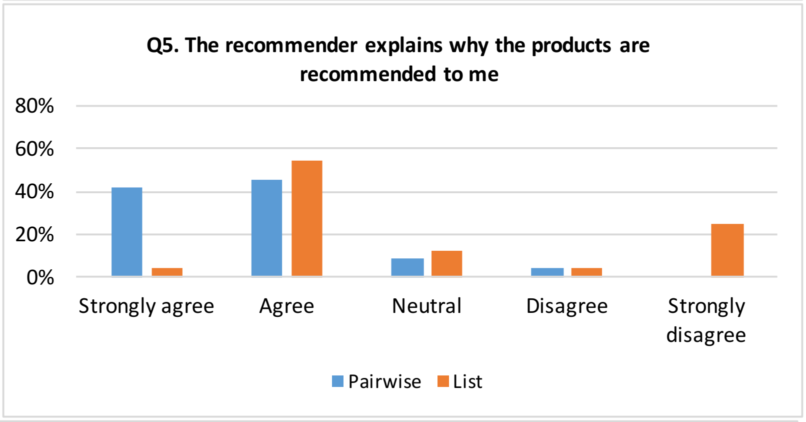


Figure G.1: Proportion details of post-stage questionnaire responses







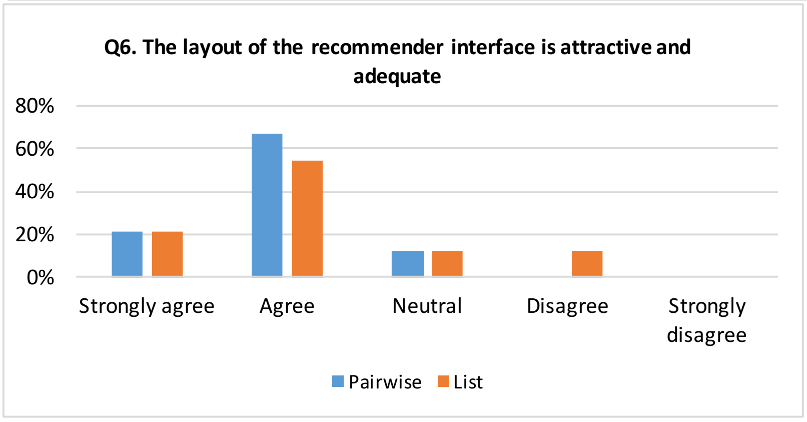
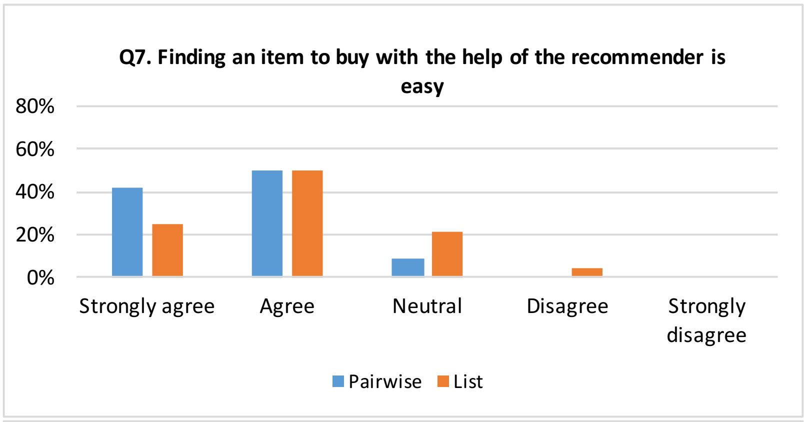
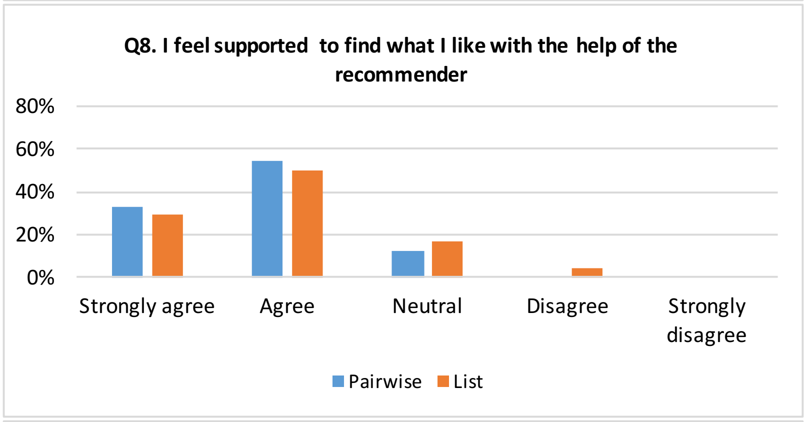
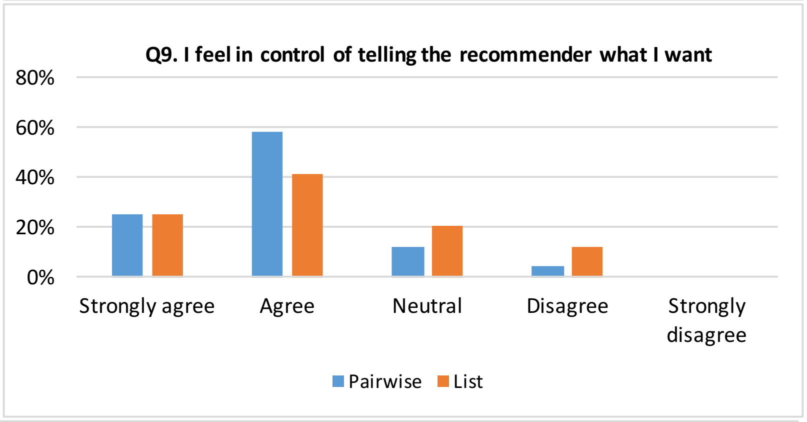


Figure G.1: Proportion details of post-stage questionnaire responses (cont.)







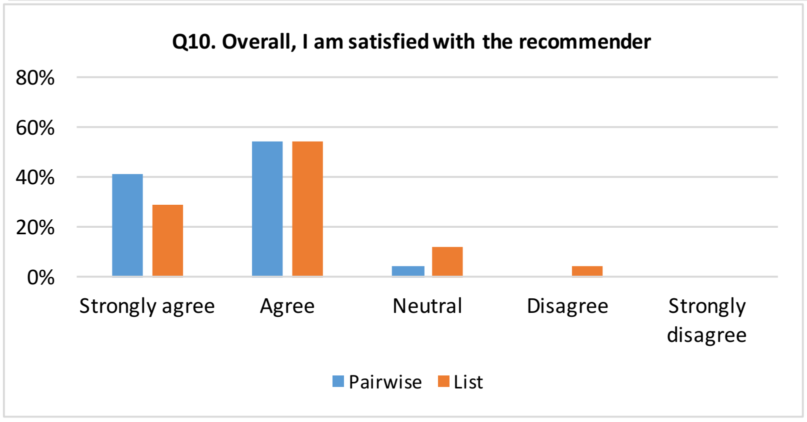
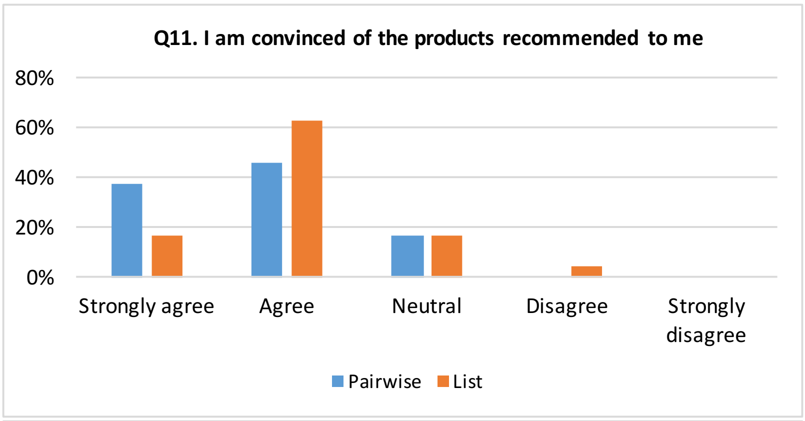
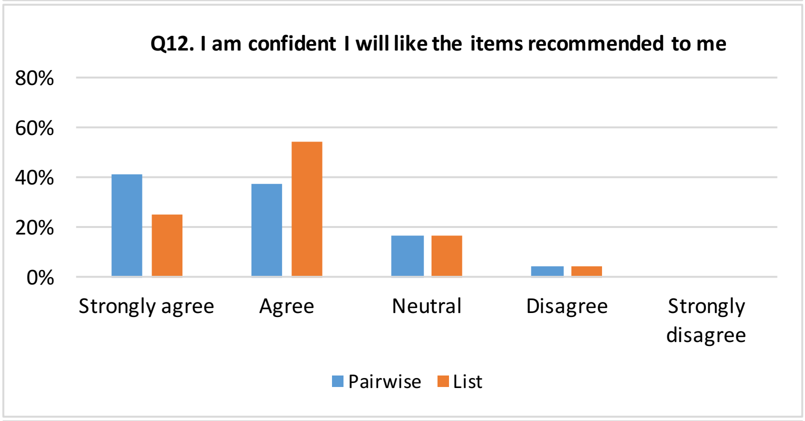
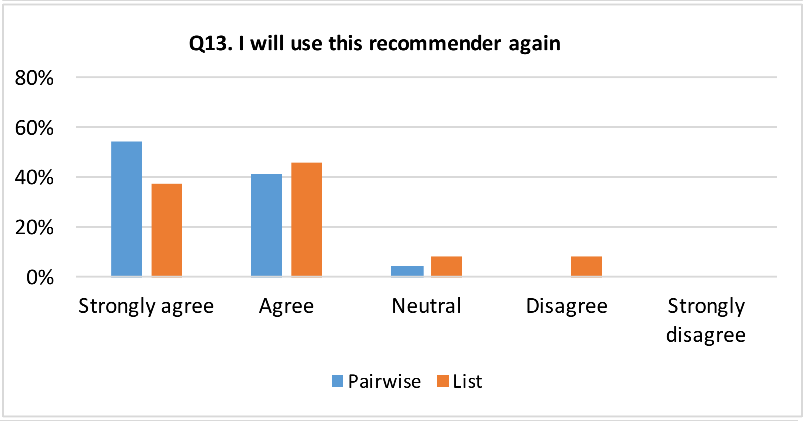


Figure G.1: Proportion details of post-stage questionnaire responses (cont.)







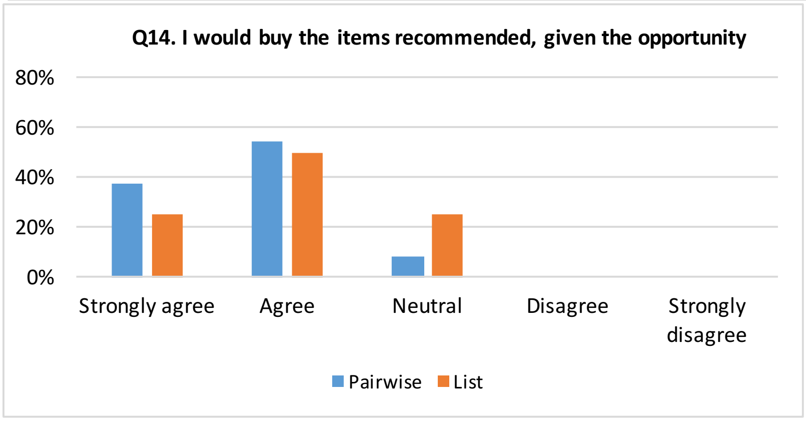
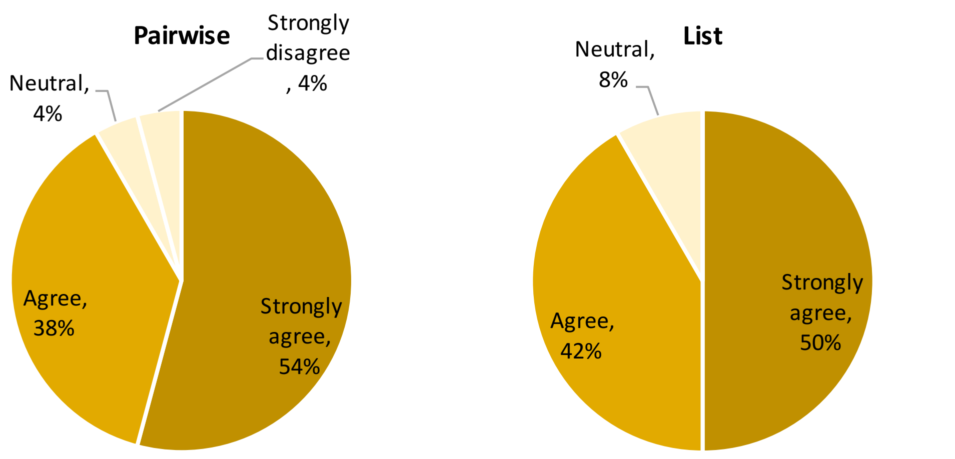
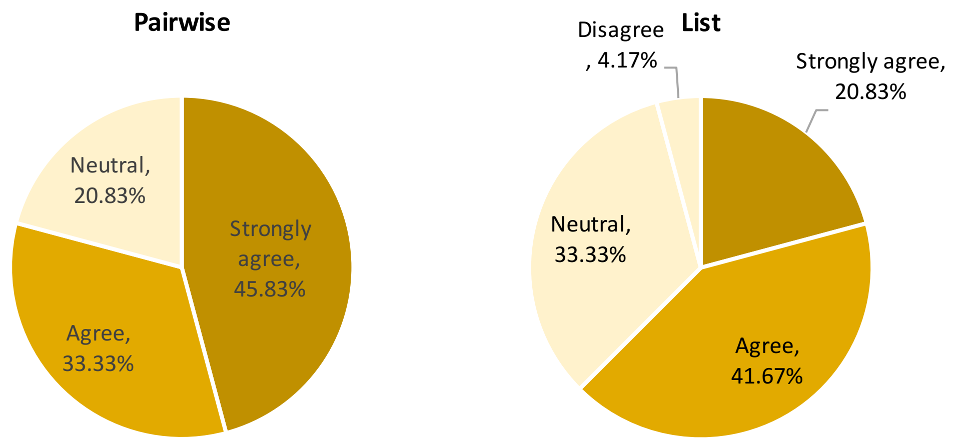


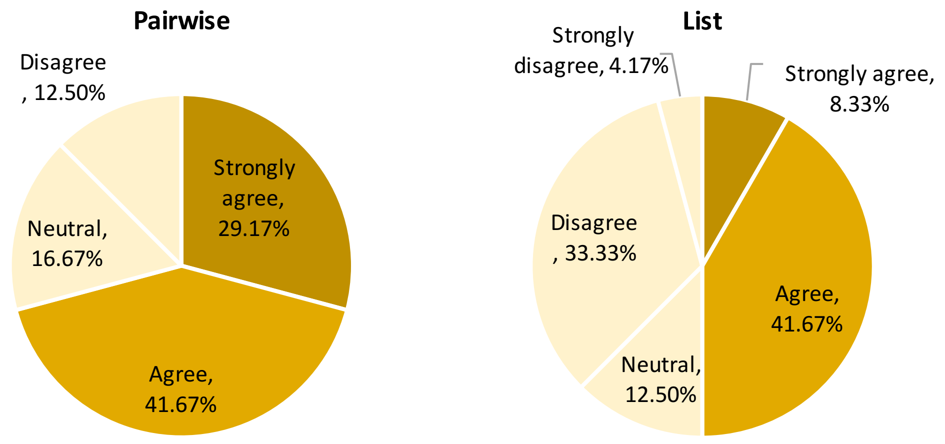
Figure G.1: Proportion details of post-stage questionnaire responses (cont.)



Q1. The items recommended to me matched my interests

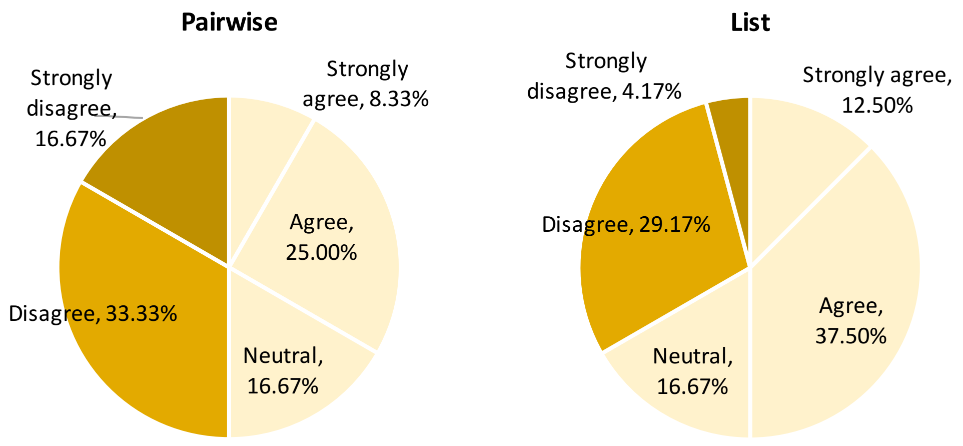


Q2. The items recommended to me are novel and interesting

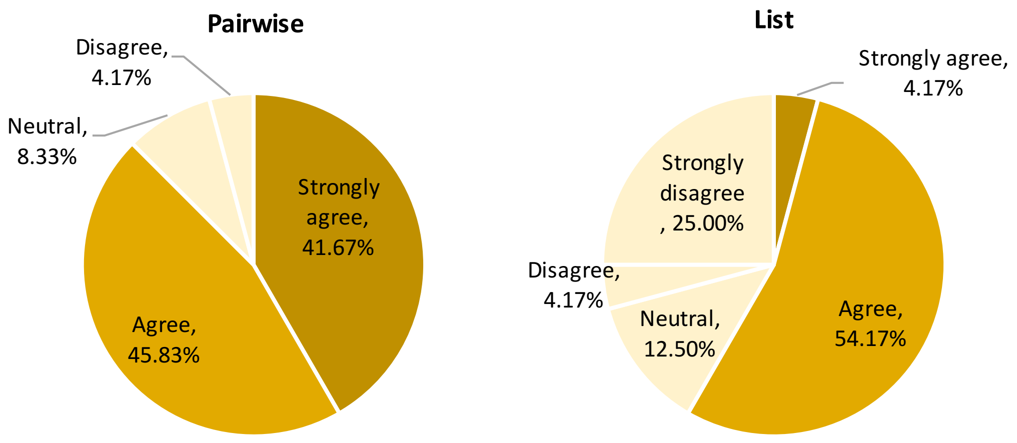


Q3. The items recommended to me are diverse

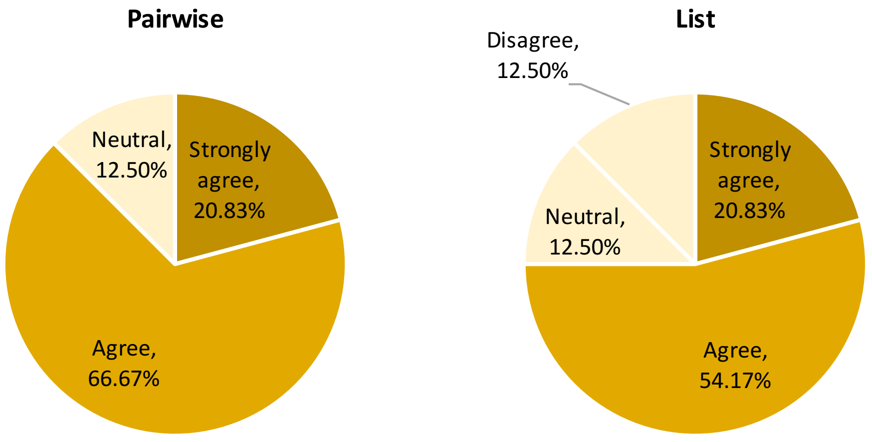
Figure G.2: The pie charts of the post-stage questionnaire responses



Q4. The items recommended to me are similar to each other (reverse scale)

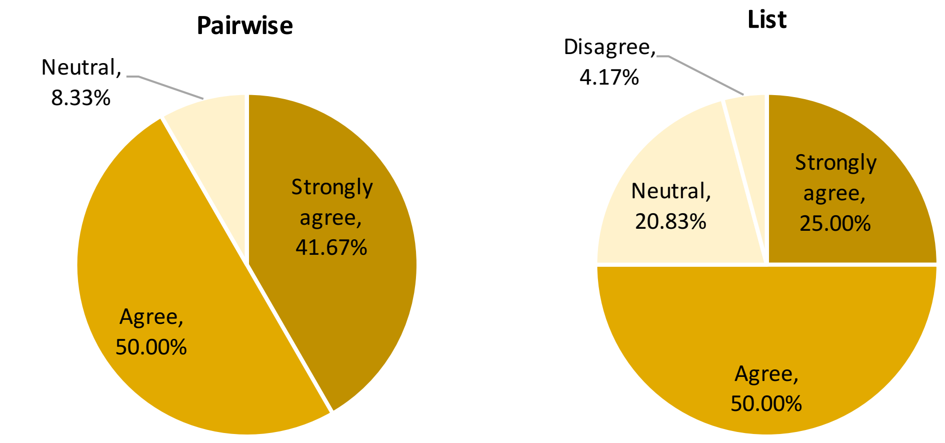


Q5. The recommender explains why the products are recommended to me

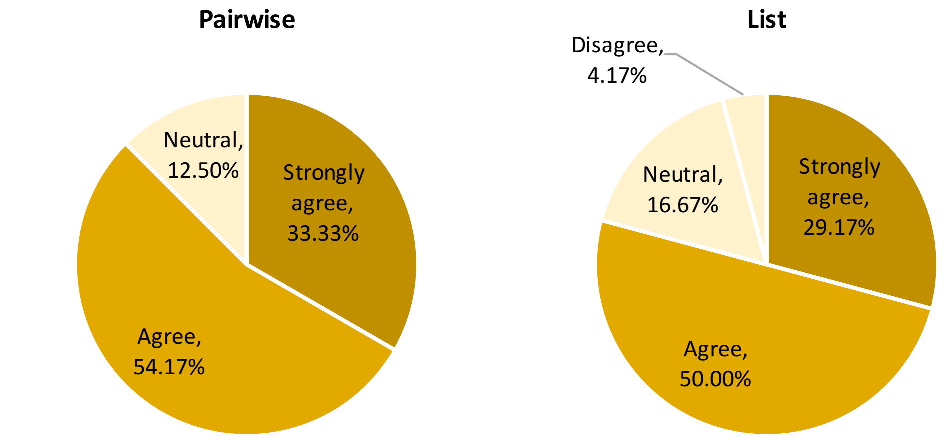


Q6. The layout of the recommender interface is attractive and adequate

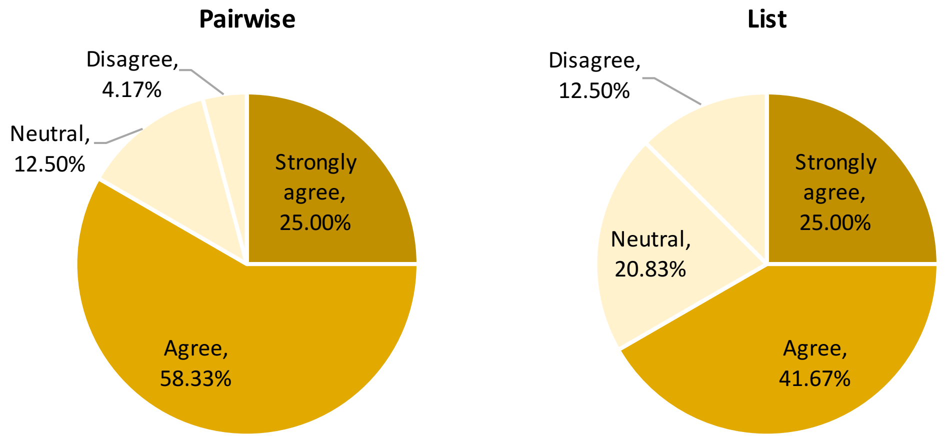
Figure G.2: The pie charts of the post-stage questionnaire responses (cont.)



Q7. Finding an item to buy with the help of the recommender is easy

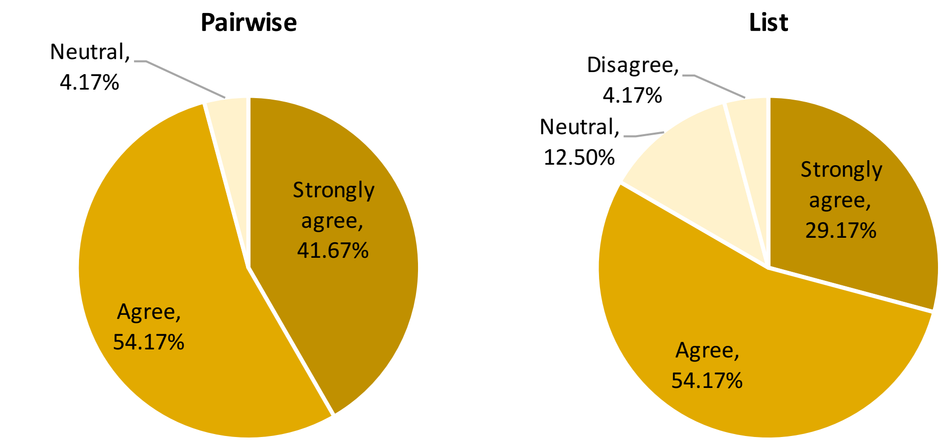


Q8. I feel supported to find what I like with the help of the recommender

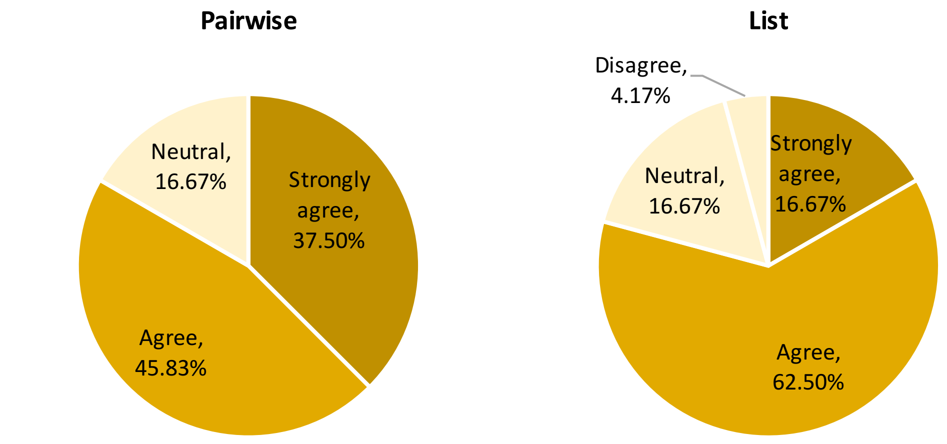


Q9. I feel in control of telling the recommender what I want

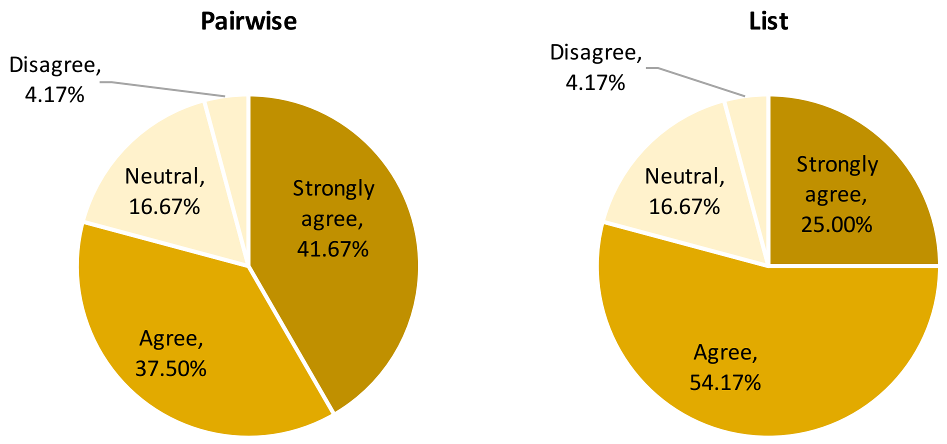
Figure G.2: The pie charts of the post-stage questionnaire responses (cont.)



Q10. Overall, I am satisfied with the recommender

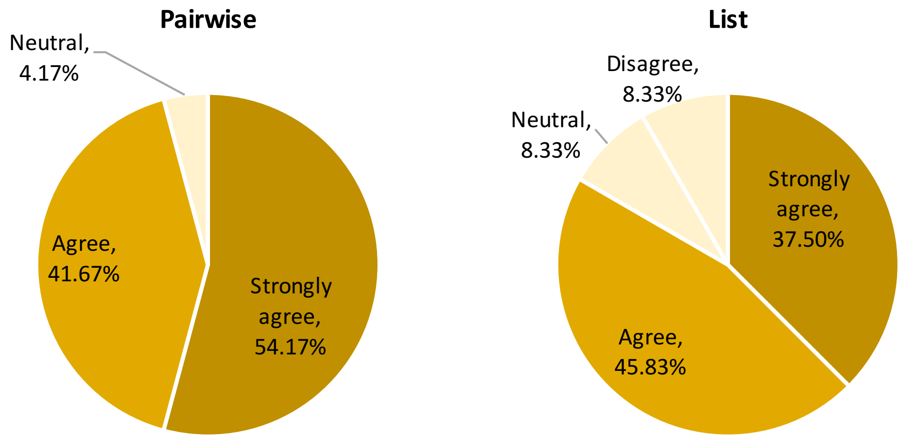


Q11. I am convinced of the products recommended to me

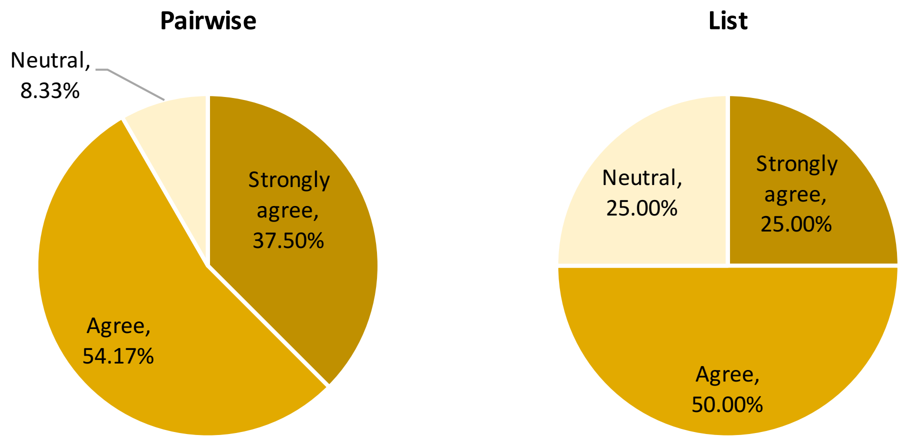


Q12. I am confident I will like the items recommended to me

Figure G.2: The pie charts of the post-stage questionnaire responses (cont.)



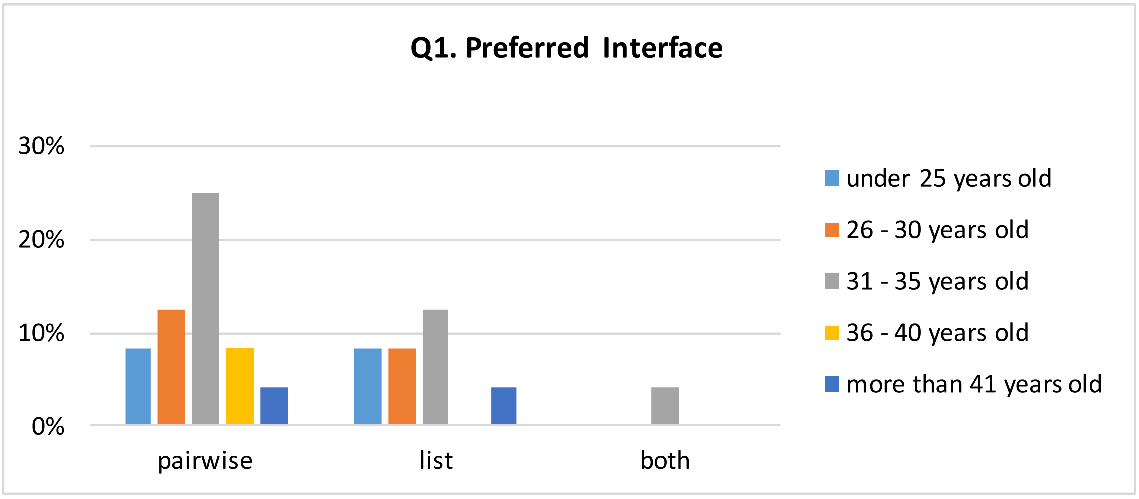
Q13. I will use this recommender again

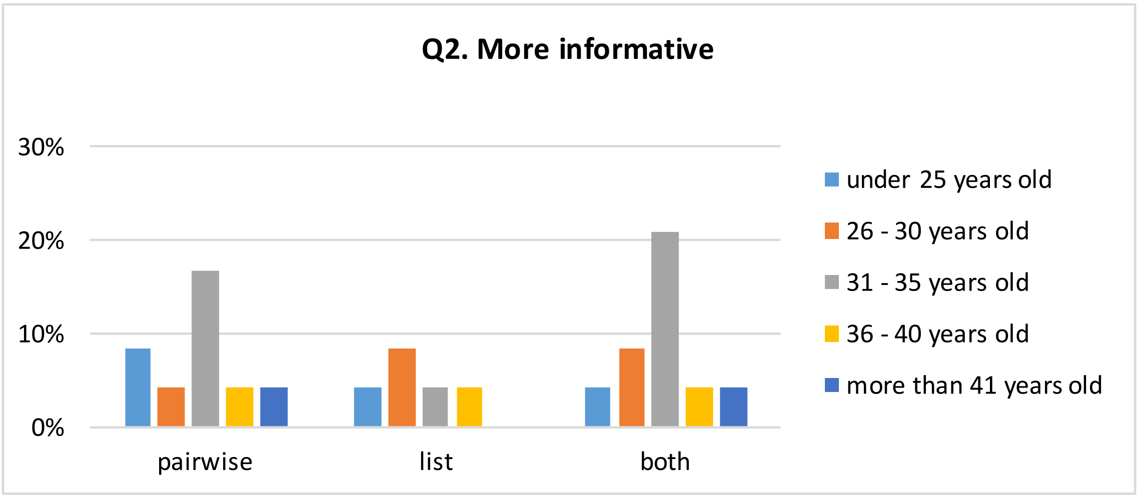


Q14. I would buy the items recommended, given the opportunity

Figure G.2: The pie charts of the post-stage questionnaire responses (cont.)

G.2 Interface preferences questionnaire results





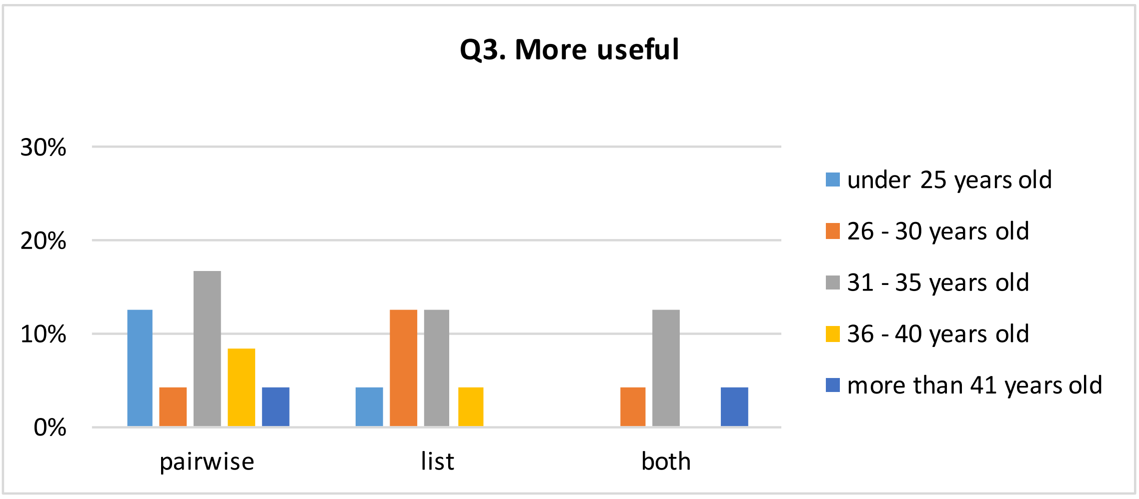
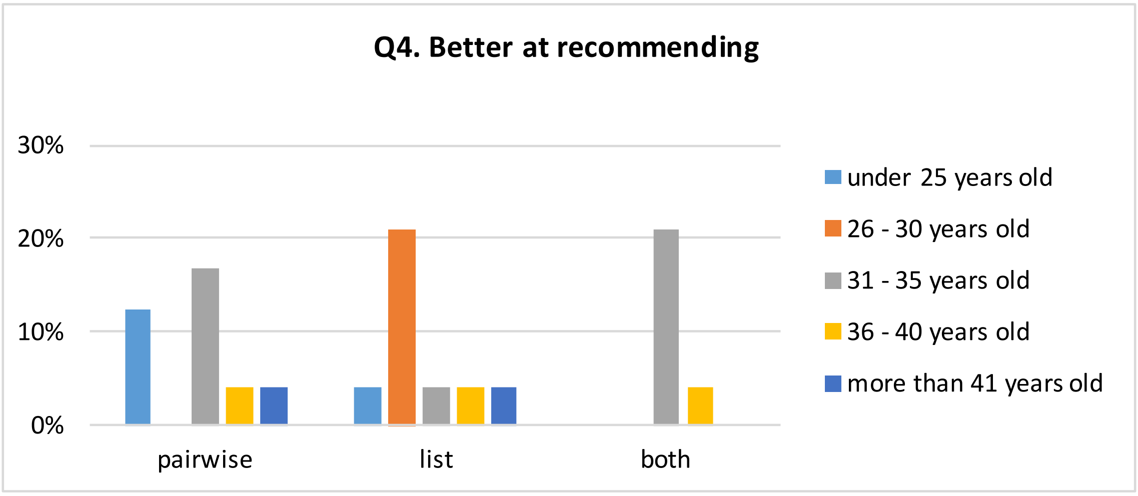


Figure G.3: Voters' details based on the ages



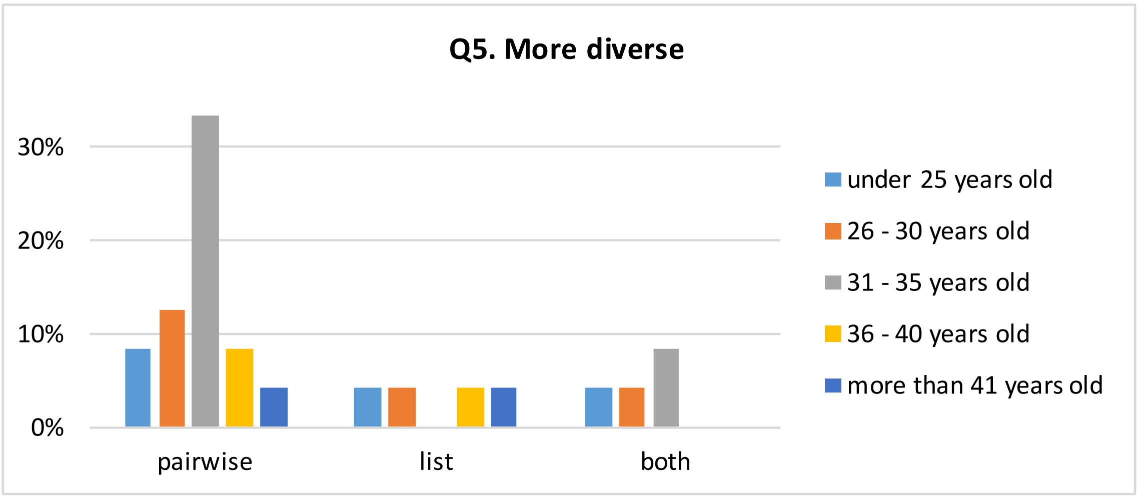
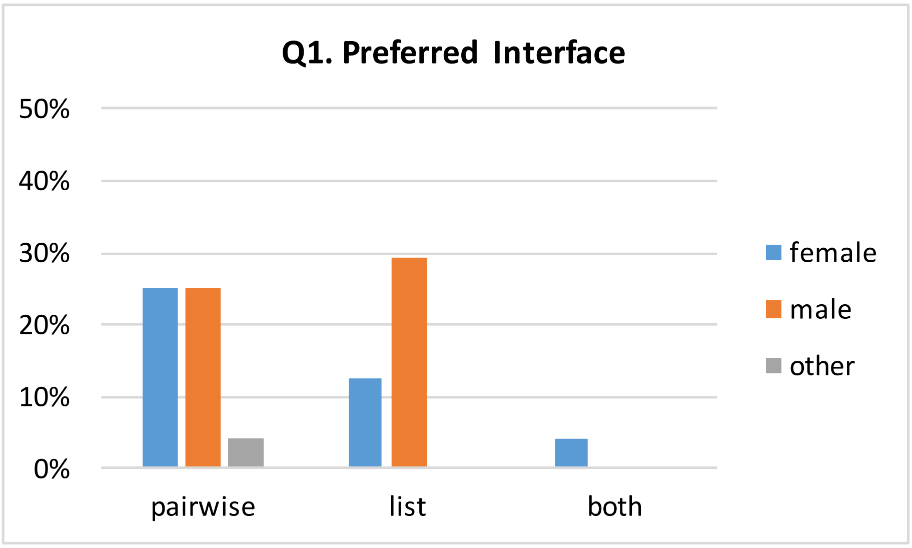
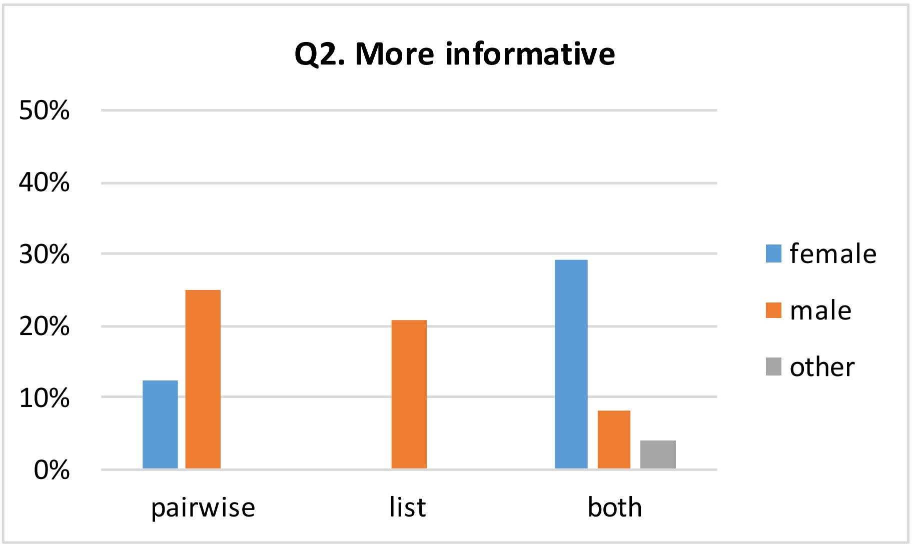


Figure G.4: Voters' details based on the ages (cont.)





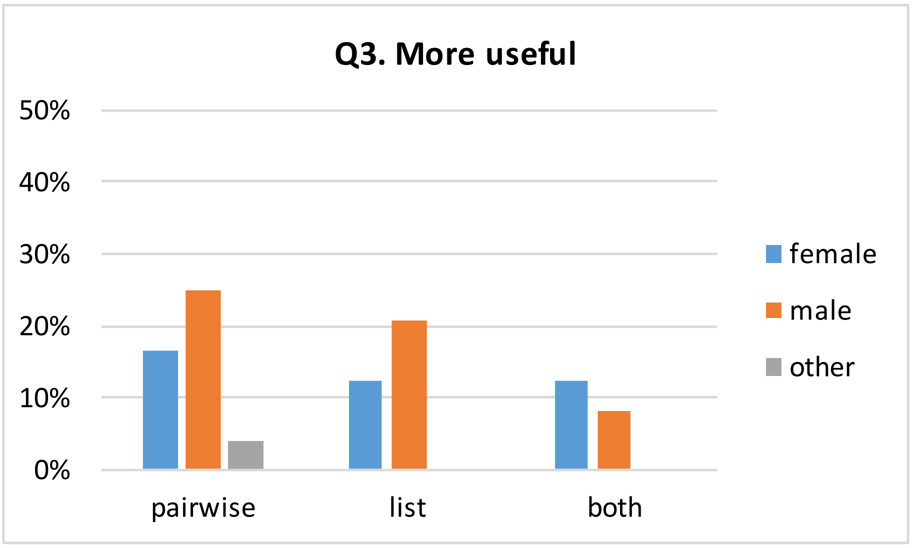
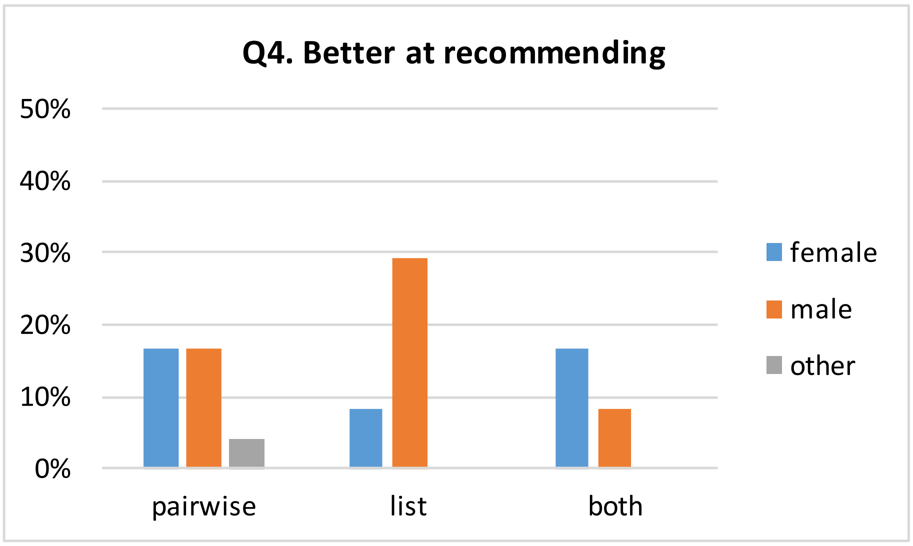


Figure G.5: Voters' details based on the genders



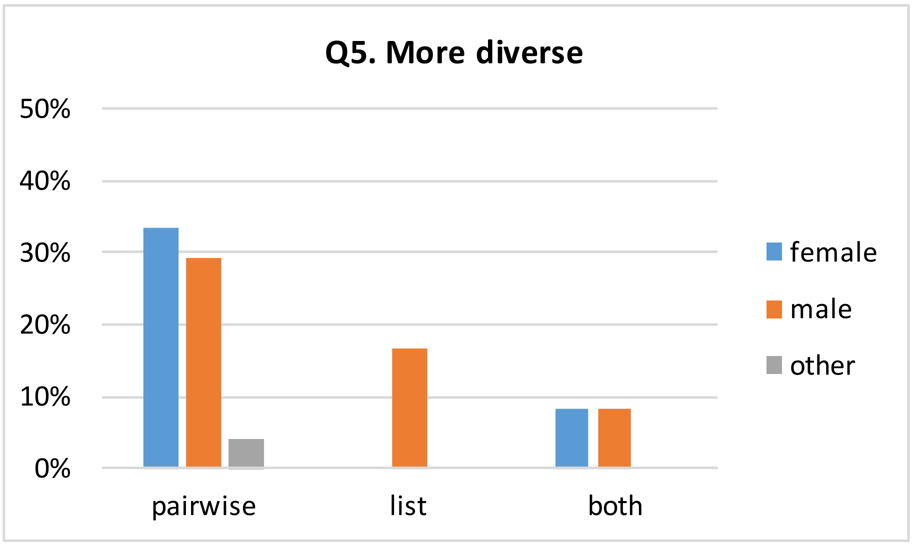
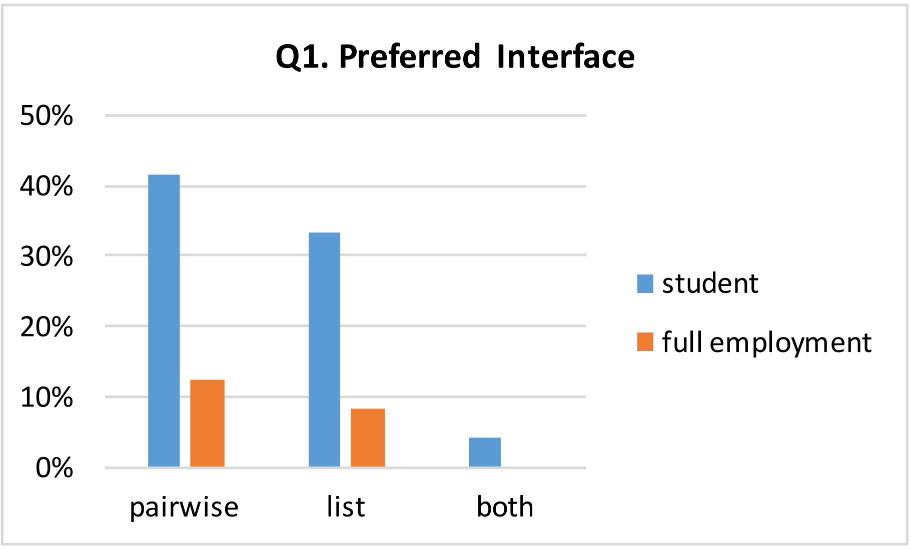
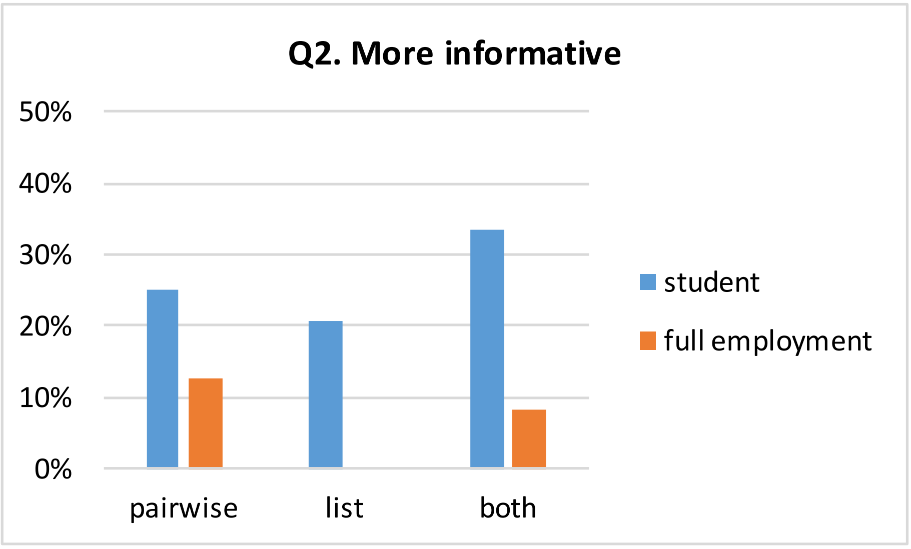


Figure G.6: Voters' details based on the genders (cont.)





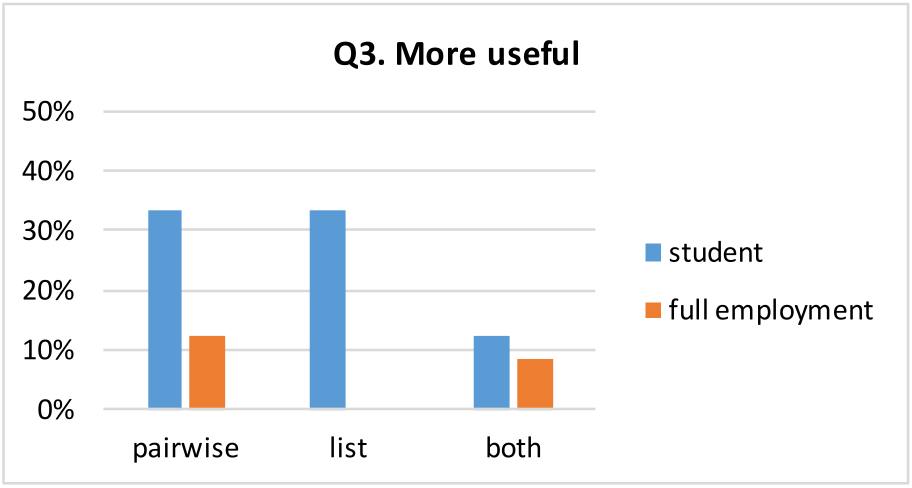
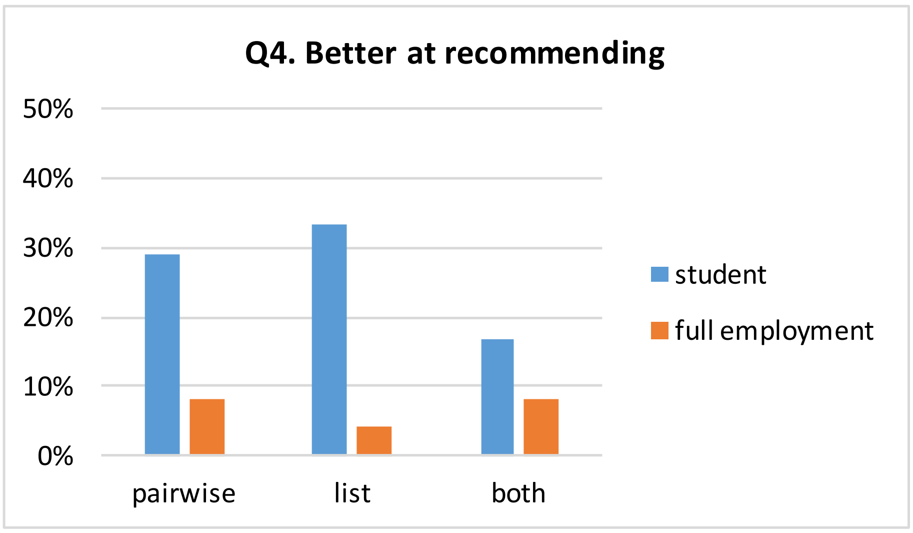


Figure G.7: Voters' details based on the professions



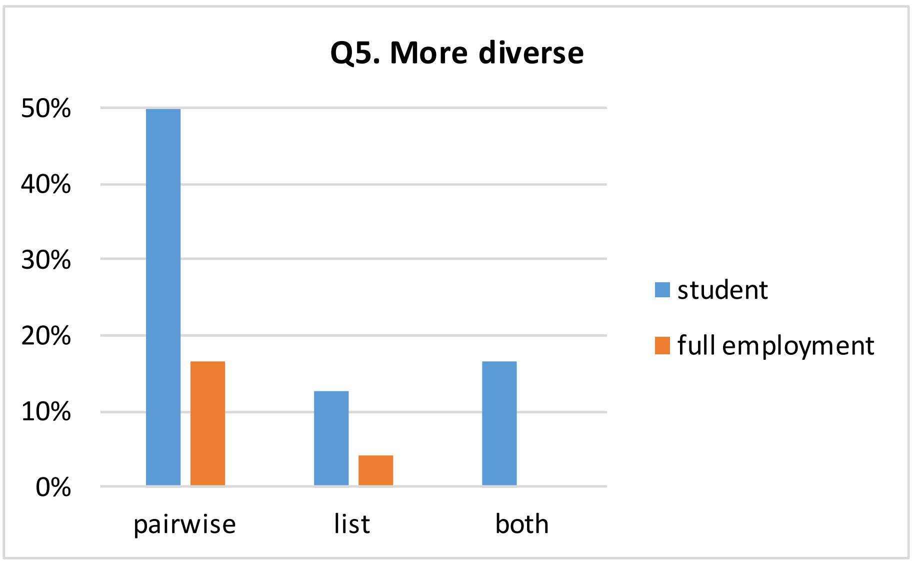


Figure G.8: Voters' details based on the professions (cont.)

Appendix H

Interview Transcripts

Table H.1: Interview transcripts

Interview ID	Transcript
1	“I think I prefer to choose pairwise comparions because I can compare two items easily. The system is really useful and helpful for customers. As a customer, we can choose the items so that the system can give us some items related to the one that we want to buy.”
2	“I prefer the single list rather than the comparisons between pairs of cars because I find it more comfortable on interface like in this computer screen, the fact that you can have as many options as possible all together rather than go step by step with two or three or multiple comparisons, with the filter, if you have in mind what you want it is easier for me to choose with the filter and give me a splash page with 20 or 50 options that I can compare all together. Not just the two and then the two. If you do not have the filter, probably you need step-by-step filtering but it’s a bit redundant. But I don’t know if making a comparison between pairs of cars could be useful for people that have no real idea of what to choose.”
3	“When I was filtering the types of cars according to mileage and price, there was no car available, it was a surprise for me, because there are so many cars in all the ranges of mileages and engine power. So it will be better and interesting for the person who is using the website if they get more choices to select from. In terms of filter, I think there can be more options because people like to play around first and then go to their final choices, so there could be more options on what people like most or most popular. I like the pairwise one better. It was easier for me because when I was filtering the cars, it showed me the car with the specifications on the side of the picture. It was very easy to choose from in terms of the information that had been given.”
4	“Well the interface is good, but from both of them I prefer the list, because I don’t have to think to choose one of the available choices so I can focus on each of the items. The pairwise just forces me to choose between two options, it is difficult for me. The recommendation is good, at least I know why the system recommended the cars for me.”
5	“Pairwise was better in my opinion. The pictures should show the different sides of the cars not only front side, some pictures only showing the back, I am confused. I like pairwise maybe because it’s more informative and the choices there are more diverse compared to the list one.”
6	“I like the pairwise interface because it gives me the benefit to compare instantly with other cars instead of comparing the homogeneous list of similar cars. I think it helped me to understand and to know my expectations. (Talking about the quality of the recommendation) Actually both of them were almost the same but the way it gave me the best cars, pairwise did a better job.”
7	“I like the pairwise one, because I guess it can help you, kind of forces you to compare the cars. So I sorted them by price, it tended to show me quite similar cars, they came up in each pair, it kind of forces me to choose between the similar cars. The standard list is OK because I am more familiar with this type on every online website. The recommendation given by the system is quite interesting because it came up with some things that I haven’t seen before, I don’t know whether I’ve seen them or not, I’ve just forgotten about it. It was certainly interesting. If I want to use it to buy a car probably I am a bit worried because it gives me slightly different choices but with kind of explanations why it showed you these cars, so it’s a quite good result.”
8	“I would say I like the pairwise better because it gives more choices and it chose everything on the side (side recommendation) instead of just disappear; you just keep an eye on whatever you choose. It helped me to find the cars I liked and also because it gives you summary about what you like. The explanations matched my preferences.”

Table H.1: Interview transcripts (cont.)

Interview ID	Transcript
9	“It is kind of a bit difficult, because in terms of the interface, I think the pairwise interface looks better and is very informative because in the section where I click Your Top Picks the interface gave me information on why we picked those cars. In the standard list interface I couldn’t find why we picked the cars. In terms of the design, the pairwise is better but in terms of the informativeness also pairwise is better, but in terms of the diversity of the choices I guess I think the pairwise is very diverse... which is sometimes, if I am not really sure what I’m looking for, that’s good but if I’m really sure what I want and then which car, which type, what kind of transmission system that I like, I think the standard list interface gave me a better choice of cars that I like. (About the recommendation) I mean it’s a matter of personal choice, so I set in the beginning that I like these very specific cars with very specific characteristics so when I expect something in the recommendation I expect something that OK, fits with my profile with what I set before in the beginning so I don’t really pay much attention to something new that happens as far as I remember, all the cars that I clicked on appeared in the recommendation section, I didn’t click on ‘see other alternatives’ for the pairwise interface, I only clicked the ‘see other alternatives’ in the standard list interface because on the first page the pairwise interface represented what I really needed so although I didn’t click three cars, I just clicked on two cars but it seems that’s OK it was quite representative of what I need. But in the standard list interface, I needed to click on ‘see other alternatives’ and then on the second page I found all the cars that really suit me.”
10	“With the pairwise it seems to me because you’re forced to compare two different cars so.. to make like a choice whether you like a large car or maybe a newer car or.. so it’s make more, that you are forced to make a choice between something... which is good. But also the good part of the other... the list interface that you are not forced to make a choice between two, so you have more freedom of looking around. Maybe I like more the first interface, maybe just because I am used to it... oh sorry, the list one, I feel more comfortable.”
11	“I prefer the pairwise interface simply because it’s the fact that at the end of the day when you’ve done choosing your cars and you move on to the recommendation provided it gives you the reason why it’s providing the recommendation compared to the standard one. (About the quality of the recommendation) Yes, I think that compared to the standard list interface, the pairwise interface gives me a more diverse output to some extent. I think this is what really got me because this is what I am looking for, I was purposely trying to get a hatchback between hundred.. well over 100 bhp, and it’s all picked up on that feature highlight.”
12	“Well actually I found I had interest in both of the interfaces so... specifically I want the cars and... in that pairs and there is a... both of them, but I found out that... the standard list interface is likely to recommend the cars that I might purchase if I don’t want these specific cars. Both of the interfaces are very easy to use and very interactive.”

Table H.1: Interview transcripts (cont.)

Interview ID	Transcript
13	"Actually both are good, but for me I think... the pairwise is good to see two cars but sometimes it's not that similar for me so I got confused, I need to choose both. Because we used to use the standard list, that's why maybe it's easier, but both are good. I think pairwise, the most advanced job using pairwise is you can choose cars on the same page... you can see more cars on the same page. I think the quality of the recommendation in both is good. I can find my choices in both interfaces. It would be good if the pictures can have more than one side of the same car, maybe if you have some customer reviews, something like that."
14	"I prefer the list interface of the recommendation, obviously making the preference better than the pairwise, the more complex one, you need to choose one, but I think the quality of the recommended cars in the end is not diverse for me at all and they are kind of similar to each other. But the list one, I think the recommendation is all attractive to me. I think it is sometimes, it's not fair if you compare two cars with very different prices or if you prefer the high price car, it makes me a lot more confused cause obviously the car with the higher price is of higher quality but if you take into account the price.. is that really worth that price, so it is hard to make a decision. If you compare the cars with similar prices is a lot more easier, I was not using the sorting method in both interfaces, I didn't notice. The results here (pairwise interface) are quite limited I think as, for example, the price is limited to 5000 to 6000 and... mileage I'm not sure.. to me it's the small it is the better for me, I really do not prefer this range. Actually I just prefer the smaller mileage but you need to compare also the price and features. I like the small engine size and yes... it is here in the explanation. And also the year... the newer the better for me, not only this range and I found the cars I like in here (pairwise). But you see these cars are very similar to each other, isn't it? But in the list I found it more diverse, you got SUV, you got a small car."
15	"I think the search filter in the pairwise, whenever I click next it goes back, it's a bit annoying for me. More or less I like the pairwise better than the list, you find more choices on a page, I don't need to scroll down to see more cars. I can get more cars quicker, the other one (list) you need to click next more often to see more cars. Well I think this pairwise recommendation is better than the list for me. I found the cars here, to some extent, yes, this explanation matched my preferences. I submit three cars each time, the same as the list one."
16	"I like standard list interface more actually, well in the standard list interface I can check the cars one by one, in the pairwise interface I have to choose one of them, but with some cars, I didn't like both, so it was actually not very meaningful to me. In the standard list interface, you know... I feel more independent to choose one. This (pairwise) felt like I had to choose one, I felt pressured. The filtering system is helpful. In the standard list interface, the recommendation was much more relevant. In the pairwise interface, the recommendation was more different from what I like. The explanation is representative but it could be.. I don't know... maybe more.. the design could be more different, I think. The attributes in the explanation are related to what I like."
17	"Both interfaces were good and interesting, but I like the pairwise one compared to the list. In pairwise I can see more available cars without using the filtering, but in the list, I have to use the filtering otherwise it will be too cheap and random. Yes, I like the recommendation given by the system and I found my preferences in there."

Table H.1: Interview transcripts (cont.)

Interview ID	Transcript
18	"I think the list one is better because the pairwise is comparing the cars and there were different cars that we couldn't compare each other, some of them are cheap, some of them are expensive, some of them are sports cars. I didn't find it suitable to compare them, therefore I choose the list one. The list one is also easy to use and remembers my choice. The recommendation quality is good, because it gives everything, I can see all the information just by looking at the small picture. I don't need to go inside of the advert therefore the interface was useful for me just looking at the information. And the price is suitable to my budget. I choose new cars mostly and it recommends me the new cars and the cheap ones and those suitable for families, I choose five-door cars because it will be better for my children and the recommendation is suitable with my choices. The application is useful, but some of the pictures just show the inside of the car and therefore I want to see the outside picture of the car, otherwise it is useful, it gives me the opportunity to choose anything I want, price, year and mileage. It is easier to look for cars with this application. Yes, it's helped me to find the cars I need, actually I found two or three cars that suit me."
19	"I think the list interface is much easier because I think pairwise got me something like confused so in my opinion I choose the list one. In pairwise interface, sometimes there is no match between this pairwise, for example, here is automatic and here is manual so I think, if it is a similar pattern in pairwise it will be better. I was not using the sorting filter, if I was using the sorting maybe I got similar car comparisons. I did find the cars I like in the recommendation and the explanations were good because I prefer the automatic over manual. The recommendation gave me automatic cars so that's OK."
20	"I think I like the interfaces. I am still wondering how this search functionality, sorting functionality can be used, but yeah, I like the interface, I like the recommendation part as well because some of those are quite useful. It was difficult to choose in the pairwise because the choices were quite random but you know it was still giving me better options. I think I found something in the recommendation which I would like to buy. The standard list interface does not distract me from choosing the cars."
21	"The pairwise interface has limited choices. I found the cars that I like in the recommendation. Actually when comparing the cars, I am considering the price and age of the cars. I prefer the newer and the cheaper car."
22	"They are both good, they bring quite good information, but I think the pairwise is more useful because it is narrowing down the options, options, options. I think I like pairwise better. The explanation is interesting and matches my preferences. It is very nice, I like it. Maybe for the pairwise, if you can make it three options instead of two that would be great. I feel it isn't difficult for me to choose between two."
23	"I think the pairwise interface is quite interesting because I can see more options than the list one. And also this is new, I have never seen it before, so I think it would be useful for a person like me who didn't know much about cars. The list one is OK for me. Yes, the recommendation helps me find the car I like."
24	"I would say that the pairwise helps me to understand what I would like to look for. I like the facts that we can go back and add more choices to get a better recommendation from the system, so I feel more in control with the system. I like the explanations, some of the features are exacly what I like, just like this body type, I prefer hatchback than any other else."

Table H.2: Transcripts codes and categories

Interview ID	Codes	Categories
1	preferring to the pairwise	interface preference
	comparing is easy; relevant	reasons for preferring the pairwise interface
	useful; helpful	general comment
2	preferring to the list view	interface preference
	comfortability on the screen; having a clear idea on what to buy; filtering is more useful with the list view	reasons for preferring the standard list view
	comparisons is suitable for someone who unsure about the choices; expecting more items to compare (rather than in pairs)	general comment
3	preferring to the pairwise	interface preference
	specifications is shown next to picture; comparing is easy	reasons for preferring the pairwise interface
	expecting more data in the system; expecting most popular items; expecting more options in the filtering	general comment
4	preferring to the list view	interface preference
	the pairs are confusing; easier to use	reasons for preferring the standard list view
	helpful recommendation explanation	reasons for preferring the pairwise interface
	good	general comment
5	preferring to the pairwise	interface preference
	more informative; more diverse choices	reasons for preferring the pairwise interface
	expecting more informative pictures of cars	general comment
6	preferring to the pairwise	interface preference
	comparing is easy; helpful to understand the expectations	reasons for preferring the pairwise interface
	good	general comment

Table H.3: Transcripts codes and categories (cont.)

Interview ID	Codes	Categories
7	preferring to the pairwise	interface preference
	helpful; forcing me to compare; capability to compare similar cars; recommending some surprising items; interesting; helpful recommendation explanation	reasons for preferring the pairwise interface
	more familiar	reasons for preferring the standard list view
	good; concern to use pairwise for actually buying a car	general comment
8	preferring to the pairwise	interface preference
	more diverse choices; helpful side recommendation; helpful to find the preferred cars; helpful recommendation explanation; correct explanation (match the interest)	reasons for preferring the pairwise interface
9	preferring to the pairwise	interface preference
	better design; more informative; helpful recommendation explanation; more diverse choices	reasons for preferring the pairwise interface
	comparisons is suitable for someone who unsure about the choices; list interface is suitable for someone who already knows his/her expectations; pairwise interface showing my preferred car in the first page; list interface showing my preferred car in the second page	general comment
10	preferring to the list view	interface preference
	dislike of comparing; freedom to choose	reasons for preferring the standard list view
11	preferring to the pairwise	interface preference
	helpful recommendation explanation; more diverse choices; correct explanation (match the interest)	reasons for preferring the pairwise interface
12	preferring to the list view	interface preference
	more convincing	reasons for preferring the standard list view
	correct recommendations; easy to use; interactive	general comment

Table H.4: Transcripts codes and categories (cont.)

Interview ID	Codes	Categories
13	preferring to the list view	interface preference
	more familiar; the pairs are confusing	reasons for preferring the standard list view
	displaying more cars	reasons for preferring the pairwise interface
	good; correct recommendations; expecting more informative pictures of cars; expecting user reviews in each car	general comment
14	preferring to the list view	interface preference
	attractive recommendation; more diverse the pairs are confusing; not aware of filtering in the pairwise interface	reasons for preferring the standard list view
	recommendation explanation can be improved; specific range in the explanation is not really helpful	general comment
15	preferring to the pairwise	interface preference
	displaying more cars; lless often to click the 'next' button and scroll down; correct explanation (match the interest)	reasons for preferring the pairwise interface
	filtering in the pairwise interface can be improved	general comment
16	preferring to the list view	interface preference
	dislike of comparing; freedom to choose; more relevant	reasons for preferring the standard list view
	correct explanation(match the interest); helpful filtering feature	reason to prefer the pairwise interface
	design of the explanation can be improved	general comment
17	preferring to the pairwise	interface preference
	displaying more cars; correct recommendation	reasons for preferring the pairwise interface
	good; interesting	general comment
18	preferring to the list view	interface preference
	the pairs are confusing; easier to use	reasons for preferring the standard list view
	good; correct recommendation; informative; useful; helpful; expecting more informative pictures of cars	general comment

Table H.5: Transcripts codes and categories (cont.)

Interview ID	Codes	Categories
19	preferring to the list view	interface preference
	easier to use; the pairs are confusing; expecting to compare more similar cars; not aware of the filtering in the pairwise interface	reasons for preferring the standard list view
	good; correct recommendation	general comment
20	preferring to the list view	interface preference
	dislike of comparing; can stay focus on searching	reasons for preferring the standard list view
	good interface; useful; correct recommendation; convincing	general comment
21	preferring to the pairwise	interface preference
	comparing is easy	reasons for preferring the pairwise interface
	correct recommendation	general comment
22	preferring to the pairwise	interface preference
	ability to narrow down the choices; correct explanation (match the interest); interesting; comparing is easy	reasons for preferring the pairwise interface
	good; expecting more items to compare (rather than in pairs)	general comment
23	preferring to the pairwise	interface preference
	displaying more cars; helpful to understand the expectations	reasons for preferring the pairwise interface
	helpful	general comment
24	preferring to the pairwise	interface preference
	helpful to understand the expectations; feel in control; correct explanation (match the interest)	reasons for preferring the pairwise interface

Abbreviations

ABox	Assertional Box
ACF	Automated Collaborative Filtering
ACM	Association for Computing Machinery
AI	Artificial Intelligence
AL	Active Learning
ALC	Attributive Language with Complement
ANOVA	Analysis of Variance
AOPRS	Adaptive Ontology Based Personalised Recommender System
API	Application Programming Interface
APARELL	Active PAirwise RELation Learner
ASP	Answer Set Programming
BTM	Bradley Terry Model
CELOE	Class Expression Learner for Ontology Engineering
DL	Description Logics
DT	Decision Tree
FOL	First Order Logic
GPL	General Public License

GUI	Graphical User Interface
HCI	Human Computer Interaction
IDE	Integrated Development Environment
ILP	Inductive Logic Programming
IMDb	Internet Movie Database
LOD	Linked Open Data
MCDM	Multi-Criteria Decision Making
ML	Machine Learning
OWL	Web Ontology Language
PC	Pairwise Comparison
PL	Preference Learning
PORE	Personal Ontology-based REcommender
PROMETHEE	Preference Ranking Organization Method for Enrichment Evaluations
QBC	Query by Committee
RDF	Resource Description Framework
RecSys	Conference on Recommender System
ROC	Receiver Operating Characteristic
RS	Recommender System
SVM	Support Vector Machine
TBox	Terminology Box
UCO	Used Cars Ontology
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
XML	Extensible Markup Language

XSL	Extensible Stylesheet Language

References

[1] Ehsan Abbasnejad, Scott Sanner, Edwin V Bonilla, Pascal Poupart, et al. Learning community-based preferences via dirichlet process mixtures of gaussian processes. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI-13)*, 2013.

[2] Behnoush Abdollahi and Olfa Nasraoui. Using explainability for constrained matrix factorization. In *Proceedings of the 11th ACM Conference on Recommender Systems*, pages 79–83. ACM, 2017.

[3] Andrejs Abele, John P. McCrae, Paul Buitelaar, Anja Jentzsch, and Richard Cyganiak. Linking open data cloud diagram 2017. URL: http://lod-cloud.net. Accessed: 7 December 2017.

[4] Brian Ackerman and Yi Chen. Evaluating rank accuracy based on incomplete pairwise preferences. In *Procedings of Workshop on UCERSTI Recsys*, volume 11, 2011.

[5] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734 – 749, 2005.

[6] Amir Ahmad and Lipika Dey. A k-mean clustering algorithm for mixed numeric and categorical data. *Data and Knowledge Engineering*, 63(2):503–527, 2007.

[7] Amir Ahmad and Lipika Dey. A method to compute distance between two categorical values of same attribute in unsupervised learning for categorical data set. *Pattern Recognition Letters*, 28(1):110–118, 2007.

[8] Hend Al Tair, Mohamed Jamal Zemerly, Mahmoud Al-Qutayri, and Marcello Leida. Pro-active multi-agent recommender system for travelers. In *Proceedings of the*

IEEE International Conference for Internet Technology and Secured Transactions (ICITST), pages 466–471. IEEE, 2011.

[9] Iman Avazpour, Teerat Pitakrat, Lars Grunske, and John Grundy. Dimensions and metrics for evaluating recommendation systems. In *Recommendation systems in software engineering*, pages 245–273. Springer, 2014.

[10] Franz Baader. Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. *Research Report RR-90-13*, 1990.

[11] Franz Baader. *The description logic handbook: Theory, implementation and applications.* Cambridge university press, 2003.

[12] Franz Baader, Ian Horrocks, and Ulrike Sattler. Chapter 3 Description Logics. In *Handbook of Knowledge Representation*, volume 3 of *Foundations of Artificial Intelligence*, pages 135 – 179. Elsevier, 2008.

[13] Liviu Badea and Shan-Hwei Nienhuys-Cheng. A refinement operator for description logics. In *Proceedings of the International Conference on Inductive Logic Programming*, pages 40–59. Springer, 2000.

[14] S. Balakrishnan and S. Chopra. Two of a kind or the ratings game? adaptive pairwise preferences and latent factor models. In *Proceedings of the 10th IEEE International Conference on Data Mining (ICDM)*, pages 725–730, Dec 2010.

[15] Florian Bauer and Martin Kaltenböck. *Linked open data: The essentials.* Edition mono/monochrom, Vienna, 2011.

[16] Majid Behzadian, R. B. Kazemzadeh, A. Albadvi, and M. Aghdasi. PROMETHEE: A comprehensive literature review on methodologies and applications. *European Journal of Operational Research*, 200(1):198 – 215, 2010.

[17] F. Bergadano and D. Gunetti. *Inductive Logic Programming: from machine learning to software engineering.* MIT Press, 1996.

[18] Mustafa Bilgic and Raymond J. Mooney. Explaining recommendations: Satisfaction vs. promotion. In *Proceedings of Beyond Personalization Workshop, International Conference on Intelligent User Interfaces (IUI)*, volume 5, page 153, 2005.

[19] Yolanda Blanco-Fernández, José J. Pazos-Arias, Alberto Gil-Solla, Manuel Ramos-Cabrer, Belén Barragáns-Martínez, and Martín López-Nores. A multi-agent open architecture for a tv recommender system: A case study using a bayesian strategy. In *Proceedings of the 6th IEEE International Symposium on Multimedia Software Engineering (ISMSE'04)*, pages 178–185. IEEE Computer Society, 2004.

[20] L. Blėdaitė. *Pairwise Preferences in Collaborative Filtering.* Master's thesis, Free University of Bozen - Bolzano, Italy, 2014.

[21] Hendrik Blockeel. Hypothesis language. In *Encyclopedia of Machine Learning*, pages 507–511. Springer, 2010.

[22] Marko Bohanec and Vladislav Rajkovic. Knowledge acquisition and explanation for multi-attribute decision making. In *Proceedings of the 8th International Workshop on Expert Systems and their Applications*, pages 59–78, 1988.

[23] Ralph Allan Bradley and Milton E. Terry. The rank analysis of incomplete block designs. I. The method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

[24] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Chapman and Hall, 1984.

[25] Lun-Chi Chen, Ping-Jen Kuo, and I-En Liao. Ontology-based library recommender system using mapreduce. *Cluster Computing*, 18(1):113–121, 2015.

[26] Sheng-Tzong Cheng, Chih-Lun Chou, and Gwo-Jiun Horng. The adaptive ontology-based personalized recommender system. *Wireless personal communications*, 72(4):1801–1826, 2013.

[27] William W. Cohen, Robert E. Schapire, and Yoram Singer. Learning to order things. In *Advances in Neural Information Processing Systems*, pages 451–457, 1998.

[28] David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Machine learning*, 15(2):201–221, 1994.

[29] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, and Dasarathi Sampath. The youtube video recommendation system. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 293–296. ACM, 2010.

[30] Tommaso Di Noia, Thomas Lukasiewicz, Maria Vanina Martinez, Gerardo I. Simari, and Oana Tifrea-Marciuska. Ontological cp-nets. In *Lecture Notes in Computer Science*, volume 8816 of *Uncertainty Reasoning for the Semantic Web III*, pages 289–308. Springer, 2014.

[31] Pedro Domingos, Daniel Lowd, Stanley Kok, Aniruddh Nath, Hoifung Poon, Matthew Richardson, and Parag Singla. Unifying logical and statistical AI. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 1–11. ACM, 2016.

[32] Carmel Domshlak, Eyke Hüllermeier, Souhila Kaci, and Henri Prade. Preferences in AI: An overview. *Artificial Intelligence*, 175(7):1037 – 1052, 2011.

[33] Michael D. Ekstrand, John T. Riedl, and Joseph A. Konstan. Collaborative filtering recommender systems. *Foundations and Trends in Human–Computer Interaction*, 4(2):81–173, 2011.

[34] Yi Fang and Luo Si. A latent pairwise preference learning approach for recommendation from implicit feedback. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM ’12, pages 2567–2570. ACM, 2012.

[35] Nicola Fanizzi, Claudia d’Amato, and Floriana Esposito. DL-FOIL concept learning in description logics. In *Inductive Logic Programming*, pages 107–121. Springer, 2008.

[36] Alexander Felfernig, Michael Jeran, Gerald Ninaus, Florian Reinfrank, and Stefan Reiterer. Toward the next generation of recommender systems: applications and research challenges. In *Multimedia services in intelligent environments*, pages 81–98. Springer, 2013.

[37] Dieter Fensel. *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*. Springer, 2001.

[38] Peter Flach. *Machine Learning: The Art and Science of Algorithms that Make Sense of Data*. Cambridge University Press, 2012.

[39] Johannes Fürnkranz and Eyke Hüllermeier. Preference learning: An introduction. In *Preference learning*, pages 1–17. Springer, 2010.

[40] Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992.

[41] Michael Genesereth and Eric Kao. *Introduction to Logic*. Morgan and Claypool, 2013.

[42] Mustansar Ali Ghazanfar. *Robust, scalable, and practical algorithms for recommender systems*. PhD thesis, University of Southampton, 2012.

[43] Penny Gillespie, Gene Alvarez, Chris Fletcher, Praveen Sengar, and David Kohler. Predicts 2014: E-commerce becomes digital for those that can adapt fast. Report, Gartner, 2013.

[44] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–71, 1992.

[45] Thomas R Gruber. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220, 1993.

[46] Ramanathan V. Guha, Dan Brickley, and Steve Macbeth. Schema. org: Evolution of structured data on the web. *Communications of the ACM*, 59(2):44–51, 2016.

[47] Sumit Gulwani, José Hernández-Orallo, Emanuel Kitzelmann, Stephen H Muggleton, Ute Schmid, and Benjamin Zorn. Inductive programming meets the real world. *Communications of the ACM*, 58(11):90–99, 2015.

[48] Shengbo Guo, Scott Sanner, and Edwin V. Bonilla. Gaussian process preference elicitation. In *Advances in neural information processing systems*, pages 262–270, 2010.

[49] Martin Hepp. eClassOWL: A fully-fledged products and services ontology in OWL. *Poster Proceedings of ISWC2005. Galway*, 2005.

[50] Martin Hepp. GoodRelations: An ontology for describing products and services offers on the web. *Knowledge Engineering: Practice and Patterns*, pages 329–346, 2008.

[51] Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. Explaining collaborative filtering recommendations. In *Proceedings of the ACM conference on Computer supported cooperative work*, pages 241–250. ACM, 2000.

[52] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering. *ACM Transactions on Information Systems*, 22(1):5–53, 2004.

[53] Matthew Horridge and Sean Bechhofer. The owl api: A java api for owl ontologies. *Semantic Web*, 2(1):11–21, 2011.

[54] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From $\mathcal{SHIQ}$ and RDF to OWL: the making of a web ontology language. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(1):7 – 26, 2003.

[55] Ian Horrocks, Peter F Patel-Schneider, and Frank Van Harmelen. From shiq and rdf to owl: The making of a web ontology language. *Web semantics: science, services and agents on the World Wide Web*, 1(1):7–26, 2003.

[56] Ian R. Horrocks. *Optimising Tableaux Decision Procedures for Description Logics.* PhD thesis, University of Manchester, UK, 1997.

[57] Ian R. Horrocks. Using an expressive description logic: FaCT or fiction? In *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 636–647. Morgan Kaufmann Publishers Inc., 1998.

[58] Caden Howell. Application of ILP to a breast cancer recurrence data set with aleph, 2009. Department of Computer Sciences, University of Wisconsin-Madison.

[59] Rong Hu and Pearl Pu. Helping users perceive recommendation diversity. In *Proceedings of workshop on novelty and diversity in recommender systems (DiveRS)*, pages 43–50, 2011.

[60] Z. Huang, W. Chung, T. H. Ong, and H. Chen. A graph-based recommender system for digital library. In *Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*, pages 65–73, 2002.

[61] Eyke Hüllermeier, Johannes Fürnkranz, Weiwei Cheng, and Klaus Brinker. Label ranking by learning pairwise preferences. *Artificial Intelligence*, 172(16):1897 – 1916, 2008.

[62] Luigi Iannone, Ignazio Palmisano, and Nicola Fanizzi. An algorithm based on counterfactuals for concept learning in the semantic web. *Applied Intelligence*, 26(2):139–159, 2007.

[63] Bjørn Sand Jensen, Javier Saez Gallego, and Jan Larsen. A predictive model of music preference using pairwise comparisons. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1977–1980. IEEE, 2012.

[64] Shaowei Jiang, Xiaojie Wang, Caixia Yuan, and Wenfeng Li. Mining user preferences for recommendation: A competition perspective. In *Proceedings of the Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data: 12th China National Conference, CCL 2013 and First International Symposium, NLP-NABD 2013*, pages 179–189. Springer, 2013.

[65] Toshihiro Kamishima. Nantonac collaborative filtering: recommendation based on order responses. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 583–588. ACM, 2003.

[66] Jörg Uwe Kietz. Learnability of description logic programs. In *Proceedings of the 12th International Conference on Inductive Logic Programming*, pages 117–132. Springer, 2003.

[67] Stasinos Konstantopoulos and Angelos Charalambidis. Formulating description logic learning as an inductive logic programming task. In *Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ)*, pages 1–7. IEEE, 2010.

[68] Amy N. Langville and Carl D. Meyer. *Who's# 1?: the science of rating and ranking.* Princeton University Press, 2012.

[69] Nada Lavrač and Sašo Džeroski. *Inductive Logic Programming: Techniques and Applications.* Ellis Horwood, 1994.

[70] Jens Lehmann. DL-Learner: learning concepts in description logics. *The Journal of Machine Learning Research*, 10:2639–2642, 2009.

[71] Jens Lehmann. *Learning OWL Class Expressions*. PhD thesis, University of Leipzig, 2010.

[72] Hector J Levesque and Ronald J Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational intelligence*, 3(1):78–93, 1987.

[73] David Lewis and Jason Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 148–156. Morgan Kaufmann, 1994.

[74] Shu-Chuan Liao, Kuo-Fong Kao, I-En Liao, Hui-Lin Chen, and Shu-O Huang. PORE: a personal ontology recommender system for digital libraries. *The Electronic Library*, 27(3):496–508, 2009.

[75] Mingfeng Lin, Henry C Lucas Jr, and Galit Shmueli. Research commentary—too big to fail: large samples and the p-value problem. *Information Systems Research*, 24(4):906–917, 2013.

[76] G. D. Linden, J. A. Jacobi, and E. A. Benson. Collaborative recommendations using item-to-item similarity mappings, July 24 2001. US Patent 6,266,649.

[77] Francesca A Lisi and Umberto Straccia. A FOIL-like method for learning under incompleteness and vagueness. In *International Conference on Inductive Logic Programming*, pages 123–139. Springer, 2013.

[78] John W. Lloyd. *Foundations of logic programming*. Springer, Jan 1987.

[79] John W. Lloyd, editor. *Computational Logic*. Springer, 1990.

[80] Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. *Content-based Recommender Systems: State of the Art and Trends*, book section 3. Springer Science and Business Media, 2011.

[81] Fabiana Lorenzi, Fabio Arreguy Camargo Correa, Ana L. C. Bazzan, Mara Abel, and Francesco Ricci. A multiagent recommender system with task-based agent specialization. In *Agent-Mediated Electronic Commerce and Trading Agent Design and Analysis*, pages 103–116. Springer, 2010.

[82] Vukosi N. Marivate, George Ssali, and Tshilidzi Marwala. An intelligent multi-agent recommender system for human capacity building. In *Proceedings of the 14th IEEE Mediterranean Electrotechnical Conference (MELECON)*, pages 909–915. IEEE, 2008.

[83] Ryszard S. Michalski, Ivan Bratko, and Avan Bratko. *Machine learning and data mining; methods and applications.* John Wiley & Sons, Inc., 1998.

[84] S. E. Middleton, D. D. Roure, and N. R. Shadbolt. *Ontology-based recommender systems*, pages 779–796. Springer, 2009.

[85] Stuart E. Middleton. *Capturing knowledge of user preferences with recommender systems.* PhD thesis, University of Southampton, 2003.

[86] Tom M. Mitchell. Generalization as search. *Artificial Intelligence*, 18(2):203 – 226, 1982.

[87] Tom M. Mitchell. *Machine Learning.* McGraw-Hill, 1997.

[88] Hla Hla Moe and Win Thanda Aung. Building ontologies for cross-domain recommendation on facial skin problem and related cosmetics. *International Journal of Information Technology and Computer Science (IJITCS)*, 6(6):33, 2014.

[89] Miquel Montaner Rigall. *Collaborative recommender agents based on case-based reasoning and trust.* PhD thesis, Universitat de Girona, 2003.

[90] Stephen Muggleton. Inductive logic programming. *New generation computing*, 8(4):295–318, 1991.

[91] Stephen Muggleton. Inductive logic programming: derivations, successes and short-comings. *ACM SIGART Bulletin*, 5(1):5–11, 1994.

[92] Stephen Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13(3-4):245–286, 1995.

[93] Stephen Muggleton and Luc De Raedt. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19:629–679, 1994.

[94] Stephen Muggleton, Luc De Raedt, David Poole, Ivan Bratko, Peter Flach, Katsumi Inoue, and Ashwin Srinivasan. ILP turns 20. *Machine Learning*, 86(1):3–23, 2012.

[95] Stephen Muggleton and Cao Feng. *Efficient induction of logic programs.* Turing Institute, 1990.

[96] Ion Muslea, Steven Minton, and Craig A Knoblock. Active learning with multiple views. *Journal of Artificial Intelligence Research*, 27:203–233, 2006.

[97] Vito Claudio Ostuni, Tommaso Di Noia, Eugenio Di Sciascio, and Roberto Mirizzi. Top-N recommendations from implicit feedback leveraging linked open data. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 85–92. ACM, 2013.

[98] Vito Claudio Ostuni, Giosia Gentile, Tommaso Di Noia, Roberto Mirizzi, Davide Romito, and Eugenio Di Sciascio. Mobile movie recommendations with linked data. In *International Conference on Availability, Reliability, and Security*, pages 400–415. Springer, 2013.

[99] Weike Pan, Li Chen, and Zhong Ming. Personalized recommendation with implicit feedback via learning pairwise preferences over item-sets. *Knowledge and Information Systems*, 2018.

[100] Deuk Hee Park, Hyea Kyeong Kim, Il Young Choi, and Jae Kyeong Kim. A literature review and classification of recommender systems research. *Expert Systems with Applications*, 39:10059–10072, 2012.

[101] Gordon D. Plotkin. A note on inductive generalization. *Machine Intelligence*, 5(1):153–163, 1969.

[102] Pearl Pu, Li Chen, and Rong Hu. A user-centric evaluation framework for recommender systems. In *Proceedings of the fifth ACM conference on Recommender Systems*, pages 157–164. ACM, 2011.

[103] Li Qian, Jinyang Gao, and H. V. Jagadish. Learning user preferences by adaptive pairwise comparison. *Proceedings of the VLDB Endowment*, 8(11):1322–1333, 2015.

[104] J. Ross Quinlan. Learning logical definitions from relations. *Machine learning*, 5(3):239–266, 1990.

[105] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings*

of the ACM conference on Computer supported cooperative work, pages 175–186. ACM, 1994.

[106] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In *Recommender Systems Handbook*, pages 1–35. Springer, 2011.

[107] Elaine Rich. User modeling via stereotypes. *Cognitive science*, 3(4):329–354, 1979.

[108] Fabrizio Riguzzi, Elena Bellodi, Evelina Lamma, Riccardo Zese, and Giuseppe Cota. *Learning Probabilistic Description Logics*, pages 63–78. Springer, 2014.

[109] Lior Rokach and Slava Kisilevich. Initial profile generation in recommender systems using pairwise comparison. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1854–1859, 2012.

[110] Nicholas Roy and Andrew McCallum. Toward optimal active learning through monte carlo estimation of error reduction. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 441–448, 2001.

[111] Neil Rubens, Dain Kaplan, and Masashi Sugiyama. *Active Learning in Recommender Systems*, book section 23, pages 735 – 764. Springer Science and Business Media, 2011.

[112] Sebastian Rudolph. Foundations of description logics. In *Reasoning Web. Semantic Technologies for the Web of Data*, pages 76–136. Springer, 2011.

[113] Ulrike Sattler. A concept language extended with different kinds of transitive roles. In *Proceedings of the 20th Annual German Conference on Artificial Intelligence: Advances in Artificial Intelligence*, KI '96, pages 333–345. Springer-Verlag, 1996.

[114] J. B. Schafer, J. Konstan, and J. Riedl. Recommender systems in e-commerce. In *Proceedings of the 1st ACM conference on Electronic commerce*, pages 158–166. ACM, 1999.

[115] Vincent Schickel-Zuber. *Ontology Filtering Inferring Missing User's preferences in eCommerce Recommender Systems.* PhD thesis, École Polytechnique Fédérale de Lausanne, 2007.

[116] Manfred Schmidt-Schauß and Gert Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[117] Burr Settles. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114, 2012.

[118] H. Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Proceedings of the 5th annual workshop on Computational learning theory*, pages 287–294. ACM, 1992.

[119] Guy Shani and Asela Gunawardana. Evaluating recommendation systems. *Recommender systems handbook*, pages 257–297, 2011.

[120] Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 1948.

[121] Ehud Y. Shapiro. *Algorithmic program debugging.* MIT press, 1983.

[122] Sajid Siraj. *Preference elicitation from pairwise comparisons in multi-criteria decision making.* PhD thesis, School of Computer Science, University of Manchester, UK, 2011.

[123] H. Small. Co-citation in the scientific literature: A new measure of the relationship between two documents. *Journal of the American Society for Information Science*, 24:265–269, 1973.

[124] John F. Sowa. Semantic networks. In *Encyclopedia of Artificial Intelligence.* John Wiley & Sons, Inc., 2nd edition, 1992.

[125] Ashwin Srinivasan. The Aleph manual. Technical report, Computing Laboratory, Oxford University, 2000.

[126] Rudi Studer, V. Richard Benjamins, and Dieter Fensel. Knowledge engineering: principles and methods. *Data and knowledge engineering*, 25(1):161–197, 1998.

[127] Mona Taghavi, Kaveh Bakhtiyari, and Edgar Scavino. Agent-based computational investing recommender system. In *Proceedings of the 7th ACM conference on recommender systems*, pages 455–458. ACM, 2013.

[128] R Development Core Team. *R: A language and environment for statistical computing.* R Foundation for Statistical Computing, Vienna, Austria, 2012.

[129] Louis L. Thurstone. A law of comparative judgment. *Psychological review*, 34(4):273, 1927.

[130] Nava Tintarev and Judith Masthoff. Designing and evaluating explanations for recommender systems. In *Recommender Systems Handbook*, pages 479–510. Springer, 2011.

[131] Ignacio Fernández Tobías. *A semantic-based framework for building cross-domain networks: application to item recommendation.* Master's thesis, Universidad Autónoma de Madrid, 2013.

[132] Heather L. Turner and David Firth. Bradley-Terry models in R: the BradleyTerry2 package. *Journal of Statistical Software*, 48(9), 2012.