University of Bergen

Faculty of social sciences

Lifting data to semantic formats

Petter Amundsen

Master thesis

2015

I. Abstract

The semantic web can offer the world of data new ways to use and see the information. However, the majority of data do not use semantic technologies. This thesis tries to explore and identify problems that occur during the transition of data to semantic technologies. I will lift three different cases: one using a set of public data from Norway; one using data from Vinmonopolet.no; and lastly one regarding the computer game League of Legends. The first two use several datasets while the last one only uses data from the game. When these data are in semantic technologies, I will show how to extract information from these cases. I will share the process of lifting, discuss what I did right and wrong in the cases and propose a method for lifting other data to these technologies.

II. Preface

This work represents the completion of a Master's degree in Information Science from the University of Bergen's Faculty of Social Sciences by the author Petter Amundsen. I would like to thank especially my supervisor Andreas Lothe Opdahl for his guidance, input and support throughout the year. I would also like to thank Espen Kvamme Rødland, Ane Amundsen and Sindre Tefre Eide for their help with the thesis. Lastly I would like to thank my family and friends for their support and for believing in me.

III. Table of contents

1 Introduction

With the emergence of semantic technologies, the possibilities of the Internet is greater than ever. Semantic technologies let us combine data from all around the web, enabling applications never before thought possible. Berners-Lee first envisioned the semantic web back in 2001. His vision was to create a web of information that is not only readable by humans, but by machines as well (Berners-Lee *et al.,* 2001). This could create an even more automated and more user friendly environment on the web. 14 years later the semantic web is still not as big a part of the web as it could be. Some have implemented the technologies, but far from all. There is one major obstacle standing in the way of these applications; the owners of the information has to make it available for semantic applications. This is holding many back because the cost of transitioning to the new technologies do not necessarily provide obvious profits, but are instead hidden until some outside source creates something new with the data.

The UK government has initiated a leap to publish data on their website http://data.gov.uk/data/. Some of these datasets are semantic but many are not. The Norwegian government has published some data on http://data.norge.no/, but there are as of this thesis not too much data available. Data about organizations from the Brønnøysund registries (BRREG, 02-05-2015), toll booths and some datasets regarding the economics of municipals are available, but there are much more that can be published. In this thesis the methods of creating semantic data will be put under the lens. I will go in depth on the problems that arise when lifting data to semantic formats. Lifting to the semantic web is not necessarily straightforward. There are many hinders along the way, increasingly so with the complexity of the data.

1.1 The evolution of the research question

The thesis was initially going to lift several public datasets: data from partifinansiering.no(PF), specific data from brreg.no and election data. The goal of this project was to use these lifted data to show how semantic technologies can improve the way we use data. The research question was in the previous version:

> *"How can the semantic web enhance public information to allow for better analysis and transparency"*

During the development it became apparent that the qualitative parts of the improvements were hard to locate. The information derived from these data can already be found on the web (KommuneProfilen, 2015), either directly or easily interpreted from different graphs or numbers. With more creativity and better knowledge of elections the outcome might have been different, but since the improvements were only time based, the research was changed to focus on the process of lifting in itself.

The research should now create a method for lifting data to the semantic web. During searches for such methods, very little was found. Ferdinand *et al.(*2004) present what can be seen as having the closest resemblance to this research. The article proposes a way to transform XML documents to RDF graphs while using the XML schemas to create an OWL ontology. Suárez-Figueroa *et al.*(2009) propose a way to write an Ontology Requirements Specification Document(ORSD). The ORSD is a document which is used "to state why the ontology is being built, what its intended uses are, who the end users are, and which requirements the ontology should fulfill"(Suárez-Figueroa *et al*. 2009). Studer *et al.(*2007) write about "Ontology development" on a very high level. In his book he includes the whole picture, including economics, management, post development and support activities. The work done in this thesis leans more towards the actual process of lifting data, including problems which can arise during this process, the evaluation of the used method and how and when to use existing vocabularies.

After the change, two research questions are attempted answered:

"Which problems arise when lifting data to semantic technologies?"

and

"Can a method that guides the lifting process be developed from these studies?".

The first question is looking for micro oriented and specific answers, while question two attempts to encapsulate some method for dealing with the variety of answers to the first question.

The research will be conducted as an empirical study where a set of different cases will be lifted and these experiences used to answer the research questions. As the thesis first started with lifting public datasets, these will constitute case 1. Two more cases are included; one about Vinmonopolets products and stores; and one lifting data from Riot Games API concerning League of Legends.

The League of Legends case is a previous project of mine. When the research changed direction, it became apparent that this project created experiences that could be relevant in the research. However, the modelling and lifting of this dataset is redone as part of this research.

In the first chapters I will discuss previously done work that are relevant and the method of the research. Then I will move on to the work done in the cases, before I propose a method for lifting data. Lastly I will end with an overview of how the research done might contribute to the field of semantic technologies and a section on future work.

2 Previous work

The field of lifting to semantic technologies is not a new field in any way, however, not many articles have been written on the subject of liftin. This chapter will look at the work that were found, as well as tools that can be used when lifting.

2.1 The Semantic web

2.1.1 The vision of The Semantic Web

Tim Berners-Lee *et al*. (2001) tells us the story about a future family. The family lets a computer automatically fetch all the information they need regarding medicines, scheduling appointments and other important work that at that point(and today) was considered manual work. His vision to introduce semantics would improve the web and allow it to be machine readable, letting us offload even more work to them (Berners-Lee *et al.,* 2001).

"The Semantic Web will allow machines to COMPREHEND semantic documents and data, not human speech and writings". With this statement he makes his intentions clear that the semantic web would require its own format for data: Resource Description Framework(RDF). RDF is a triple based data structure which allows the connection and meaning behind different data to be represented. He presents multiple ways to enhance these data using taxonomies, ontologies and inference rules. Taxonomies defines classes and what they are, ontologies represent what types of things exists and inference rules lets the data be interpreted, and find connections hidden in the structure (Berners-Lee *et al.,* 2001).

Several papers has been written on the subject of introducing the semantic web. As early as 2004, only 3 years after the article from Berners-Lee, Mills Davis, Dean Allemang and Robert Coyne published an article on how the semantic technologies was going to be more widely used in the decade. The article told tales of how the technologies were going to become more widely used on the web (Davis *et al.,* 2004). There are other papers that discuss how the semantic web actually were doing after 2010; the time when Davis said it was going to be big. These articles have been describing a rise in the usage, and how large companies like Google and Amazon have embraced it (Cardoso, 2009). Although it might not have been as big as Davis article argued that it would become, the papers agree that it has increased its tally, and if it continues will become a central part of the internet going forward. Berners-Lee *et al.* (2009) wrote an article on the problems that have emerged for semantic technologies and how the semantic web is going to help the web fulfil its potential if these are overcome (Bizer, 2009, Janev, 2010). The articles presented up to this point will be used to argument for the fact that using semantic technologies to represent the data in this application is a viable, and the best alternative.

The paper “The Semantic Web revisited”, Berners-Lee delves into the progress the semantic web, discussing technologies such as OWL, how organizations are necessary for the development of the semantic web and what he envisions the next steps to be (Berners-Lee, *et al.*, 2006).

The whole thesis is standing on these papers’ shoulders, trying to encapsulate their vision into a method for lifting data.

2.1.2 RDF/OWL

RDF is the core way information is represented on the semantic web. It is a technology used to create relationships between data. RDF represents its data in triples containing subject, predicate and object. This lets computers understand relationships between two different entities, as long as the predicates and URIs are the same (RDF Semantic Web Standards, 25-04-2015). For example, in Wikipedias semantic alternative DBpedia, a duplicate name of a page can be solved by having two different URIs, "http://en.dbpedia.org/resource/Ola_Normann(Musician)" and "http://en.dbpedia.org/resource/Ola_Normann(Writer)" are examples of such. While there could be more of each of those examples, one can build upon them to represent more differences.

OWL is an extension of RDF, and has as goal to allow for more rich and complex relationships. It lets us create assumptions and restrictions on the model, which might be required in more complex scenarios (Allemang, 2011). Horrocks *et al*. (2003) writes about the emergence of OWL as an ontology language. They discuss the positive sides, such as more diverse and complex expressive power than the predecessor RDF, and negative sides of the language, such as how it at a point of complexity will lack in expressive power. Moreover it discusses the syntactic problems, the semantic problems and how it functions.

These papers can be used to argument for the fact that the cases should use OWL to represent the data, such that it can make more complex statements and create restrictions, and to show that OWL is indeed sufficient enough to express these cases.

García-Castro (2010) tries to compare the different semantic web tools' interoperability. It presents the OWL interoperability benchmarking to do this comparison while using owl as the interchanging language. It uses a problem-focused approach to evaluate the technologies. It includes only a handful of the most used and known technologies, whereas the ones relevant to this paper's production is Jena, KAON2, SWI-Prolog, Protégé Frames, Protégé OWL, SemTalk, and WebODE. This study helps with identifying problems that can occur when combining datasets.

Allemang (2011) covers many topics within semantic technologies, such as RDF, OWL, SPARQL, good modelling practices and much more. It talks about the semantic web as a whole, displaying how it can be an improvement to the web, and the history behind it. This book will be used as a guide in areas where it is needed to brush up on knowledge of RDF, and as a quick reference for basic OWL and RDF language.

The World Wide Web Consortium is a community which develop standards within web technologies. It is led by Tim Berners-Lee, who is seen as the inventor of the Internet(WWW) who also envisioned the Semantic Web. Their website features detailed specifications on anything from RDF to OWL, and these pages will be used as a reference when modeling the graphs (Semantic Web W3C, 31-3-2015).

2.1.3 SPARQL optimization

Liu, C. *et al. (2010)* presents a RDF query engine that utilize an inverted index structure well as histograms to allow for more efficient queries. Stocker *et al.(2008)* attempts to "formalize the problem of Basic Graph Pattern(BGP) optimization for SPARQL queries and main memory graph implementation of RDF data". From this article I discovered the notion that lowering the amount of joins required by a query can improve the efficiency of it. Combined, these articles will be the backbone of the discussion of modelling with SPARQL optimization in mind.

2.2 Web scraping

Web scraping is the process of identifying, organizing and structuring data from web sites through their HTML. The HTML files are often the only way to find these data and considering that the HTML is not meant to structure the data for retrieval, but rather to make it human-readable, this process is often no minor task. Additionally, the web sites available on the web oftentimes has an incorrect structure, with either misuses of tags or other problems.

Zheng *et al.* (2007) has proposed a method for template detection and wrapper generation. The proposed method uses a DOM tree to cluster the web pages derived using a training set. This paper is relevant because it discusses and proposes a solution to identifying websites based on their template. Such templates are generated by scripts that decide which template to use, then inserts data into them. The vast number of webpages being analysed in case 1 make this type of grouping method useful. Considering that the proposed method uses a supervised algorithm, the method can inspire ways to scrape data from situations that resemble partifinansiering.no.

2.3 Evaluation

The SEQUAL framework is used as the evaluation for the proposed method. It is proposed by Krogstie *et al.* (1995). In this article two previous frameworks by Lindland *et al*. (1994) and Pohl (1994) are compared and their underlying traits analyzed. This framework is inspired by these two and is built as an extension of the framework by Lindland *et al.* (1994). The paper released by Jørgensen *et al.* (2001) discuss the use of this model in relation to active models. The article talks about what constitutes an active model, which in this regard are models that are available to the user during runtime, are open for change to fit the local needs and that otherwise can be changed after development. They go on to discuss the new requirements that these models would need and the thought behind these qualities.

2.4 Ontology building/Lifting

Ferdinand *et al*. (2004)'s article on lifting XML to OWL is the closest I have come to finding research that resembles this. It shows us how XML schemas can be transitioned to OWL format while using the XML documents to form the RDF graphs. This is on a lower level, which can help when attempting to find useful answers to the first research question. The fact that this article handles XML lifting means that this research does not need to focus as much on these technologies.

Suárez-figueroa *et al.* (2009) proposes a way to create a "Ontology specification document". This document is a way to define the goals of the ontology, who the end-users are and the overarching goals of the ontology. The document uses a set of competency questions(CQ) to define the requirements of the ontology, which inspired the use of these in the proposed method.

2.5 Other literature

Bharati and Chaudhury (2004) discusses how information quality and system quality affect the users in decision support systems. Decision support system are designed to help management make decisions on problems where the data are not particularly structured. As decision support systems can be imagined created using semantic technologies I will use this article to argue for the importance of modelling in such a way that SPARQL queries are not as time consuming.

Yang *et al.* (2004) writes on the subject of perceived online service quality. The article references a wide variety of other works' propositions for measuring these qualities. These qualities are discussed and compared, and in the end the authors propose 6 different dimensions to measure the perceived quality with. These include: reliability, responsiveness, competence, ease of use, security and product portfolio. The discussion behind the responsiveness quality is used to show how time consuming SPARQL queries should be avoided.

2.6 Tools

2.6.1 Protégé

Protégé is a platform for constructing semantic models. It has full support for OWL 2 which allows it to handle complex modes. Included are descriptive logic reasoners such as HermiT or Pellet. It allows for visual representations of the models, and supports plugins to expand on this and other features. This tool is used to create the models for the cases in this research. (Protégé, 30-04-2015)

2.6.2 Jena

Jena is a framework for Java that allows the developers to take advantage of semantic web technologies. It is compatible with every version of semantic technologies, including but not limited to OWL Full, OWL DL and OWL Lite. Jena offers reasoners for different variants of OWL. It has a wide variety of functions, allows for the storage of graphs in TDB stores as well as saving in standard OWL/RDF. The Jena library will be used as a core part of the lifting process in two of the cases(Apache Jena, 2015).

2.6.3 Apache commons

The Apache commons libraries are a collection of utility libraries for Java. The Apache commons-io is used within the lifting process of some of the cases to make reading datasets a more streamlined process(Apache Commons, 2015).

2.6.4 Gson

The Gson library is a Java library for interpreting, modifying and creating JSON data. This library is used especially in case 3: League of Legends considering all the data from the API is in JSON format(Gson, 30-04-2015).

2.6.5 Eclipse

This is the IDE I used when creating all the lifting code based in Java. It is a community driven open-source IDE which is rich in functions. It includes a debugger, allows for plug-ins to customize the IDE and can be used for many other languages. It has proven to be an important part of the development process in the Java based cases(Eclipse, 2015).

2.6.6 Rdflib

Rdflib is a library for Python that can handle semantic technologies. It is used in case 1 to lift the data, in contrast to the two other cases which use Java. It is not as rich as Eclipse, but it includes all you need to lift data(Hees, 2015).

2.7 Research method literature

Bryman(2012) discusses the different quantitative and qualitative social research methods. He discusses the difference between them, when each of them should be used, their criticisms, similarities and much more. The sections on validity and case studies in this thesis are very much influenced by this book. Bryman explains why research methods are important, the different methods in details and shows the reader the small borders between some of the methods.

Berg(2001) describe case studies in more detail and explains the different types of case studies that exists. He goes in depth on qualitative research methods, but also only these. He explains the different thoughts behind qualitative and quantitative methods, but does not explain quantitative methods in depth. He explains the techniques for “effectively collecting, organizing and making sense from qualitative data”(Berg, 2001:preface).

Stake(1994) writes on the different applications of case studies and how these can be categorized into *intrinsic*, *instrumental* and *collective case studies*. This is used to explain why this research use the “collective case study” approach considering the amount of cases and the units of analysis.

Jackson(2008) discuss data collected in research can use statistics in their analysis. This has been used to argue how the research is used differently from quantitative and statistical research.

Hevner *et al*.(2004) writes about design science, a research method which aims to create artifact which can improve some area in the desired field of research. The article provides a set of guidelines to follow during the research which allow the reader to have a good understanding of the research method.

3 Research Method

This chapter discuss the approach this thesis has taken to achieve its goal, why I chose to include the different cases and datasets, and how well I perceive the validity of the research to be.

3.1 Qualitative vs Quantitative research

In social research, there are two main methods: qualitative and quantitative. Quantitative research is the historically more dominant social research method. It has a more structured process that takes you from a theory to a conclusion. Its four main points is measurement, generalization, meaning that the data gathered often can be applied to people who was not in the study, and therefore correctly representing a group rather than some individuals; causality means the research often wants to not only conclude on how things are, but why they are that way; and replication, meaning that if the data was gathered again, the outcome should be similar. Qualitative research is more focused on using words to articulate a theory, rather than using numbers to justify or show relation(Bryman, 2012).

Qualitative research receives criticism on a few points: the research might be too subjective, meaning that the findings can be reliant on the researcher's view of what is important; the difficulty of replicating the research findings, also much based on the biased view the researcher might have; generalization problems, often formulated as how a few persons can represent a whole group.
This in mind, the quantitative method is still looking like the less relevant of the two methods. This is based on a few more points about qualitative research, one of which is that qualitative research is a more open-ended than quantitative. As quantitative requires a more structured process including collecting data with surveys in many cases, the qualitative method seems more effective for the question in this project. The participant observing as stated in Bryman's book
seems both more applicable and relevant(Bryman, 2012).

The result of this thesis is an analysis of the process of lifting these cases to use semantic technologies and from this analysis proposing a method that can be applied to new datasets. The unit of analysis is therefore not the datasets themselves, but rather the process of lifting them. This can be compared to research on the interviewing process of individuals, rather than the individuals themselves. This means that it will not produce any quantitative data, nor does it contain any predefined theory; but is instead an exploration of the process of lifting data to semantic formats. It is therefore evident that a qualitative method is most suitable for this research.

3.2 Design science

Design science is widely used within IS research. The purpose of research that follows such method is to create an artifact that solves a problem within the field of study.

Hevner *et al.* has proposed a set of guidelines to use when performing such research:

"

1. Design an artifact: Design science research must produce a viable artifact in the form of a construct, a model, a method or an instantiation.
2. Problem relevance: The objective of design-science research is to develop technology-based solutions to important and relevant business problems.
3. Design evaluation: The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.
4. Research contributions: Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies.
5. Research rigor: Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.
6. Design as a search process: The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.
7. Communication of research: Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.

"(Hevner *et al*., 2004:83)

Considering that the end result of this thesis is a proposed method for lifting data, the method can be viewed as a sort of artifact. Design science is more used within IS research, but the fact that this research consists of lifting different cases(datasets) to semantic technologies to create experiences, and from these experiences create a method, the design science research method has a different approach than what is desired. The method is more iterative and heuristic than what is aimed for in this research.

3.3 Case studies

Case studies are generally used when conducting research in social sciences. It is used when researching groups of people, individuals or events and are not intended to gather data (Berg, 2001). As a qualitative method it focuses on analyzing the results from a single or set of in depth analyses. This is in contrast to quantitative research, which focuses mostly on large amounts of data and to generate statistical generalizations from them or tests hypotheses on these data (Jackson, 2008). The studies requires the researcher to get in depth information on the subject. The researcher can use a wide variety of information sources, such as being part of a group. (Berg, 2001).

While the fields that use case studies are often not related to computers and more towards social research, the idea of case studies can be a valid approach to problems such as these, where the goal of the research is to analyse a process and improve upon it. In this thesis I am going to attempt to use this methodology to assess the state of a set of different datasets and propose a universal method for lifting them.

Case studies can be divided into three groups, intrinsic, instrumental and collective case studies. Intrinsic studies are used when the research is related to the individual case, attempting to understand that case in particular. Instrumental case studies are studies that attempt to understand the theory behind a given case. The case itself is therefore not of significant importance, but has a "supportive role, a background against which the actual research will play out"(Berg, 2001). Collective studies are research that uses several instrumental cases to perform the research. Considering the state of the data which will be analyzed and lifted in this thesis, the collective studies are a fitting methodology for the research, considering the datasets will be picked based on their individual characteristics (Stake, 1994, Berg, 2001).

3.4 Cross sectional design

Bryman (2012) discusses in his book "Social research methods" the differences between cross-sectional design and case studies. He argues that the difference is that case studies are more interested in the case itself. He shows many examples where the "case" in a cross-sectional design study is a location and merely serves as a background for the samples. Many of these examples are cross-sectional design being presented as case studies and therefore I wish to show why this research use the case study method.

Bryman argues that the requirements for a cross-sectional design study are a few points: having more than one case, the data being sampled at a single point of time, and *quantitative and quantifiable data* which are used to find patterns between the cases. Considering the last point, *quantitative and quantifiable data,* this research would not fit into cross-section design, as the cases have very different results and the data are non-quantifiable (Bryman 2012).

In the examples of the discussion of the two methodologies, the cross-sectional design studies did not have the location as the unit of analysis. Instead, the location was the relationship between the cases

(Bryman, 2012). His point is that the research done in these locations does not represent the population as a whole, and as such I have to apply the same reasoning to this research. Case 1 does include a lot of the different data structures and show more problems that arise with lifting data, but more cases with different types of information has to be present to find a method that can hope to represent data-lifting as a whole.

Because of these reasons the research should not be able to use cross-sectional design nor design science, but instead find case studies to be the more suitable research method.

3.5 Cases

The following sections will discuss the cases which has laid the foundation of this thesis. As mentioned earlier. In addition to the public datasets, one concerning Vinmonopolets data are included as well as a previous project of mine. I will be discussing the thoughts behind choosing the different datasets, what can be achieved from the lifting and general information on the datasets. The inclusion of both the Vinmonopolet case and League of Legends case was somewhat opportunistic. However, I will show that the cases have been through a discussion on whether or not they should be included and why they eventually were.

3.5.1 Case 1: Vinmonopolet

Vinmonopolet is a government run chain that holds the only rights to sell alcoholic beverages containing more than 4.75% alcohol to the general public. These include anything from strong beer to wine and spirits(vinmonopolet.no). The information offered on the website describes all the different products that are sold either in their stores or online. The goal of the lifting process is to connect the data with more information from other sources, as well as making it possible to improve the search and filtering functionalities of their website.

The website’s search function can be used to find both articles and products, however I will only focus on the latter since I am not lifting the articles part of the website. The search function does not let the user specify any specific field to search for. Although this is doable with other technologies, it is simple when using semantic technologies.

The filter functionality is supposed to let the user maneuver towards a certain subset of items. It works by having a predefined set of fields with a set of values which can be chosen from. These do not include fields that are freetext or otherwise potentially unique, but rather fields like the product's characteristics, regions, type etc. An option for filtering the amount of alcohol an item contains is absent from the functionality, with the exception of above and below 22%. This is probably due to the wide variety of values, however, it could solved by using a range of values instead, such as setting minimum and maximum values. Additionally, not all of the beverage types can be filtered out. One example of this is the "øvrig svakvin" or "other table wine" which can be found with the search functionality.

Why it is included in the research

This case offers some new characteristics to the research. The data are well structured, easy to understand for humans, and were easy to obtain. When linking to other datasets, those datasets were already in semantic formats. This lifting process was also less hindered by the state of the data. This is in contrast to Case 3: Public datasets where weaknesses such as misspellings, aging of data and similar could be found. Additionally, it's one of the larger units of study as it contains over 14 000 products with many different fields. Lifting these data will provide experiences linking to outside datasets that are already on semantic form as well as using geographical data. It also contains the largest hierarchy of classes in the data.

The goals of the lifting process are as follows:

1. Be able to display the origin of products on a map(Find the coordinates of its origin)
2. Be able to display find products originating from a given region
3. Be able to display find the closest store that sells a given product
4. Be able to display show which products stores within X-km sells
5. Allow for improvement in the search and filter functionalities

3.5.2 Case 2: League of Legends

This dataset was originally a project I was working on in my spare time about half a year before starting this thesis. However, I realized during the work that it might be valuable to incorporate into the thesis, as it expands on the thesis quite well considering it not only is coming from a completely different field, but is also reliant on an API for its data. Due to the modelling being done a year before it was included here, both the API and data were different than today. I decided to redo the modelling considering I had gained a lot more experience with modelling at this stage.

League of Legends is a team based computer game which is played either 5v5 or 3v3. It has several maps which decide the game mode, number of players and objectives of the game. Each player controls a “champion”; an avatar. Each champion has 5 “abilities”(spells) in addition to two “summoner spells”. In the game, a player is referred to as a “summoner”, meaning these two spells can be used by all champions. Each team starts in either the bottom-left or top-right corner of the map, where they can buy items to make them stronger, regain health and come back to life after dying. The goal of the game is to destroy the enemy team’s base, located at the other side of the map. Each team has to destroy “towers” and kill “minions” which gives them better access to the enemy base and grants gold to grow in strength.

The project goal was to display information on each champions win-rate against each other. The reason I wanted to create this is because the games statistics are in many ways flawed and unhelpful, but that is not something I will go into in this thesis. Generally, what the program was designed to offer was a graphical representation of the win-rate of different champions. Even though that makes it different from many of the other statistical sites on the web, what really made this different was the customizable nature of the graph(technically a SPARQL query). The program was to offer options to show champion win-rate in specific individual matchups and team compositions, not just overall win-rate. Another goal was to let the user specify several champions, on either team, and get a graphical representation of how well another champion would do if placed in one of the vacant spots. Win-rates of champions when played by a specific player are also included, but are not a big priority considering the amount of games needed for statistical significance.

The model needs to incorporate any statistics which might be relevant and usable in a graph. The timeline data, which contains actual events taking place in a game, is too specific as they are mostly unique(there can be made distinctions but it would be hard to visualize in a graph, and the program is not intended to include these actions).

Why it is included in the research

The League of Legends case was considered a good addition considering it being reliant on an API, being ineligible for outside connections(unless it was datasets also using this data), having a different format(JSON) and having restrictions on availability. With “restrictions on availability” I here show to the fact that the LoL API requires a game account, API key and performing several queries to fetch data on one game with a limit to queries per 10 seconds(Riot Games API, 31-05-2015). The League of Legends case also introduces some new problems such as the same data being represented differently in several places. Furthermore, the fact that the data were supposed to be updated continuously shows a new angle of lifting data, which in itself can make this a valuable unit of study.

I used the goals already in place from the previous iteration, which means the program should be able to:

1. Show winrate of champions or combinations of champions
2. Show winrate of champions, or combinations of them, when in the same team as another champion
3. Show winrate of champions, or combinations of them, when in the opposite team of another champion
4. Show amount of games of champions or combinations of them
5. Show amount of games of champions, or combinations of them, when in the same team as another champion
6. Show amount of games of champions, or combinations of them, when in the opposite team of another champion
7. All of the above with a specific player playing a champion
8. All of the above in games from a specific time period
9. Filter by fields in the game(first blood for the champions team, not including timeline data)
10. Incorporate new data from the API regularly

As this thesis is about semantic technologies and the lifting process, the graphical aspect of this application has not been included in the goals.

3.5.3 Case 3: Public datasets

At the beginning of the research, the planned datasets were from partifinansiering.no(PF), the Brønnøysunds registry(BRREG) and valgresultat.no(election data). These were the core that allowed us to get information on how much money was donated to a party and how the election went, country-, county- and municipality-wide.

This case contains several units of study. The partifinansiering.no data, BRREG data and election data will all be treated as different units of study. The other small datasets, such as postal codes, municipal data etc. are relatively simple and will be treated as references and small additions that build on the other data.

The goals of this case are:

- Show what organizations contribute to which political parties over the years
- Show how much individuals contribute to political parties, divided by municipals, years and average income
- Show the change in election results over the years

- Show the change in election results in comparison to donations
- Show how the municipals largest industry code by employment affect their votes

The goals in the start of the research was to find qualitative improvements in this case. Because the research changed course the goals stated above is in a sense only what I required of the case. The attachments show larger variety of queries and solutions that goes beyond these goals.

3.5.3.1 Partifinansiering.no

This is a website that displays information regarding the income of political parties. It does not show the expenses or how the parties use the money, but shows the donations over a certain threshold that each party receives, as well as where their other income comes from. This is all based on an annual statement that each party has to submit. This data will be used by connecting the data to other datasets, such as BRREG to display information on who and what types of organizations donate to which parties.

Why is it included in the research

This data were one of the core datasets behind the idea to show information on political funding. To use it the lifter has to read Excel files as well as scrape data from HTML pages. It is a useful unit of study to build our lifting method on, because it offers a tough dataset to lift considering the data are hard to extract as well as having many weaknesses. These pages are not especially consistent in their structure and contain many weaknesses such as misspelling of organizations names etc. Additionally the data are old(2005-), which means that the names and addresses of the contributors might not be the same as today. Hence, it provides us with useful experiences with data washing. This data also has to be connected to the BRREG data before it can make any sense, and therefore has a different lifting process - one that involves external linking - than the majority of other datasets in this research.

3.5.3.2 BRREG

This is a registry of organizations in Norway. It allows the users to search for organizations and displays information on these to the user. It is available in both bulk, through an API and by scraping the website. The data will though be gathered from the API and through scraping. The decision behind this is discussed more in the work chapter of the case.

Why it is included in the research

This is included mostly to accommodate the partifinansierings data, and would not make any sense without it. It has no connection to other units of study than PF(it is connected to postal codes, industry codes etc.) but still plays a major role in this case. Considering that both scraping HTML pages(partifinansiering) and the use of APIs(League of Legends) it does not bring much in regards to availability or data washing, but it contributes in other ways. The connections made between partifinansiering.no and these data introduces many problems with lifting data. As the data from PF has many weaknesses, this process requires some creativity to connect the two. As it is part of this connection it is part of a unique way of lifting(in this research) that requires the data to be linked together before it is lifted. It also provides valuable experiences with data extraction as its data are extracted from two different endpoints and has to create a connection within itself.

3.5.3.3 Election data

The election data were originally going to be fetched from valgresultat.no, but it was discovered that there were differences between the numbers from the Norwegian Social Science Data Services(NSD) and from valgresultat.no. In a conversation on e-mail with the NSD, it became apparent that the valgresultat.no data(which now redirects to the official government website) only contained estimates. Therefore the election data are now gathered from the NSDs Municipal Database. This database is not open for anyone and one has to apply for access to it. The data here are gathered largely from Statistics Norway(SSB), but the data are not available from SSBs API. These data are connected to the partifinansiering data.

Why is it included in the research

The election data are mostly included to show how elections are affected by donations. It is available in CSV format, which is used in the Vinmonopolet case as well, but in contrast this datasets uses several files to represent the data and therefore needs to identify each file before lifting it. The difference between this dataset and the other smaller datasets is that this one requires a larger model.

During development the remainder of the datasets were added for different reasons. Some were needed to make sense of the data and some added more depth to the whole picture.

3.5.3.4 Additional datasets

In addition to those already mentioned, the following datasets are included, but not thought of as distinct units of study. These include:

- Selected data from SSB
    - Municipals land and water areal
    - People employed in the different industry codes in municipals and counties

    - Population count for counties and municipals
    - Average income for counties and municipals.
- Municipals from 2000-2013
- Postal codes
- Industry codes

The data from SSB are intended to allow the application to find statistics based on areal, industry codes, population, income and combinations of these, as this can be useful in election research for journalists etc.

This case will also include the lifting of postal codes. These are meant to accommodate the BRREG data to not only show valid postal codes, but which municipals these reside in as well as converting codes to postal places. This will be connected to the data from both partifinansiering.no and brreg.no. This data are gathered from (Bring, 31-05-2015).

The industry code data contains information on the hierarchy of industry codes as well as some descriptions of them. This data will be used with the data from BRREG and used to group together different types of organizations.

The municipal data from 2000-2013 is merely a overview of which municipals existed in which years. These will be used to identify municipalities that are merged or splitted into new ones and the names of these.

In addition to the ones listed in this section, other datasets were considered at some point of the planning. For instance http://www.bdb.no was considered since it contains information on the organizations economies which could be relevant when combined with other data from the BRREG registry. By using these data I could have shown how an organization’s economics corresponded with their contributions to political parties.

Overall, this case will provide experience using weak data; data that has to be washed, complex linking between datasets, a variety of different data structures, web scraping(BRREG and PF), APIs(BRREG) and lifting data in bulks(PF).

In this chapter I have discussed why the research is using case studies instead of other research methodologies such as design science and cross-sectional design, the thoughts behind picking the different cases and given the cases themselves an introduction.

4 Work

This chapter will present the lifting process of the cases. The work done here resulted in the experiences that are the background of the proposed process. The cases will be presented in sections of their own, presenting where and how the data were fetched, the decisions behind the data models, and the transformation of the data to semantic technologies.

4.1 Case 1: Vinmonopolet

This section will go into detail on the lifting process performed on the data from Vinmonopolet. This process was a lot more simple than case 3 as its data was more structured and less weaknesses.

4.1.1 Data background

This data are all available as a dump in CSV format. Furthermore, the data are divided into two files, one describing products offered in the stores or online, and one describing the stores and various data about them. The data are updated once every day.

The products are described using:

- Item number
- Item name
- Volume
- Price, price per litre
- Type(cognac, aquavit etc.)
- Product selection, shop category
- Characteristics(bitterness, sweetness, fullness etc.)
- Color, smell, taste
- Food that the drink fits to
- Country, district, sub district
- Vintage
- Raw-materials
- Method
- Alcohol, sugar, acid
- Ability to be stored
- Producer, wholesaler, distributor

- Packaging
- Cork type
- Item url

The regions are described in free text and are not necessarily “real” regions. For instance, cognac items can be described using the region “Cognac traditional”, which is not a politically recognized region but rather a term referring to certain regions where cognac is made.

The “store category” is a reference to the size\revenue of the stores. The higher the category is the larger the variety of products the store sells. The size is in this regard defined by the yearly revenue of the store (Vinmonopolet).

The store are described by the fields:

- Store name
- Physical address(street, postal number, postal place)
- Mailing address(street/postbox, postal number, postal place)
- Phone number
- Shop category
- Longitude
- Latitude
- This week number
- Opening hours each day this week
- Next week number
- Opening hours each day next week

The fields describing opening hours have been ignored because they do not offer anything towards the goals of the case. These data would become outdated rather quickly, as it only shows opening hours two weeks from the data are released. The data would therefore take up unnecessary space or have to be removed continuously.

Both the store and the items has an additional column which describes the time the data were last updated.

4.1.2 Modeling

The goal of the lifting is as written earlier to connect the data to geo data and to integrate the item and store data better. The model therefore contains two main classes, the item and the store. The rest of the data are mostly literals with the exception of regions, companies, product selection and shop category. These are represented using different URI’s and are therefore resources. This means that the structure is rather simple, however there are several hierarchies of classes.

Because of the amount of different types of beverages, the hierarchies are shown in the list below as opposed to a graph because of the sheer amount of classes.

The beverage hierarchy:

- Thing
    - Beverage
        - Beer
        - Distilled Beverage
            - Akvavit
            - Bitter
                - Bitter under 22
            - Fruit liquor
            - Genever
            - Gin
            - Grape liquor
            - Liqueur
                - Liqueur under 22
            - Rum
            - Vodka
            - Whisky
            - Other distilled beverage
                - Other distilled beverage under 22
        - Fruit wine
            - Sparkling fruit wine
        - Nonalcoholic
        - Wine
            - Fortified wine

            - Aromatized fortified wine
            - Madeira
            - Other fortified wine
            - Port wine
            - Sherry
            - Velmut
        - Red wine
        - Rosé
        - Sparkling wine
        - White wine
- Company(dbpedia.org)
- Food
    - Aperitif
    - Big game
    - Cattle
    - Cheese buffet
    - Dessert
    - Fish
    - Light meat
    - Sheep
    - Shellfish
    - Small game
    - Swine meat
    - Vegetables
- Selection
    - Base selection
    - Ordering selection
    - Portion Selection
    - Test selection
- Shop category
    - Independent assortment
    - Shop category 7
    - Shop category 6
    - Shop category 5

- Shop category 4
- Shop category 3
- Shop category 2
- Shop category 1

The beverages are divided into the types shown on the Vinmonopolet website.

The other fields are divided into either data properties or object properties. The data properties link to literals while object properties link to resources. The characteristics predicates(bitterness, sweetness, freshness, fullness and tannins) are all sub predicates of the "hasCharacteristic" predicate. This allows for simpler queries when asking for multiple different types of beverages.

The shop categories expand on each other. This relationship is not represented using hierarchies, as there are some semantic problems with such a representation. I discussed at length whether to use hierarchies or not and whether the categories actually are in a parent-child relationship. How this would be solved in SPARQL queries made it feel clearer and more intuitive to have this relationship represented using skos:broader and skos:narrower.

Each of the classes and predicates have a Norwegian and English label to allow for easy translation and to automatically display the information of a given product. When creating a representation of a product, these labels can be used to easily create a template for such product.

For outside vocabularies, the coordinates are represented using the standard http://www.w3.org/2003/01/geo/wgs84_pos#lat and http://www.w3.org/2003/01/geo/wgs84_pos#long predicates. The addresses are as in the other cases using the schema form. However, I did have to add an additional class "MailingAddress" predicate which represents the address that can be mailed to. The "schema:PostalAddress" predicate, however, represents the physical location of the store. These will in most cases point to the same resource, but there are cases where they differ(postboxes etc.). "foaf:name" and "dbowl:company" represents the name and class of companies respectively.

4.1.2.1 The WineOntology

Before starting any work with this dataset, outside vocabularies that could be applied to these datahad to be cataloged. The WineOntology, which is derived from http://ontolingua.stanford.edu/doc/chimaera/ontologies/wines.daml and available at http://www.w3.org/TR/owl-guide/wine.rdf, is the most prominent of these. It offers a lot of the concepts this case requires. It includes food that fit each drink, the grapes which make up each wine, regions etc. The reason it was decided against using this vocabulary and instead create an own is because:

- The characteristics is represented in a different way; being divided into 3 levels instead of 12.
- The regions are predefined and not matched up against any coordinates. This means any new regions we find must be created as a class rather than a link to another dataset. This is an unnecessary middle step.
- The ontology has defined different classes and range/domains on properties this would create inconsistencies in the ontology.
- Modifying this ontology to fit the demands of the dataset which is being lifted means that any outside user would have several different options of how the data were represented instead of one specific way.

This model is influenced by the WineOntology model, but is created and tailored for the use within Vinmonopolets data.

4.1.3 Lifting

Because the data was all received in CSV format, the lifting process can use much of the same methods as the public dataset case. It is worth mentioning that in contrast to Case 3, this case uses Java instead of Python as a programming language. This means I could take advantage of libraries such as Jena. Additionally, the semantic model was created in Protégé first and then included in the Jena model.

Each of the columns are linked to a predicate in the model using a HashMap. This is also the case for the beverage classes. This makes for a rather simple lifting process where it iterates through the lines and fields, and creates triples as told from the HashMap.

There were some problems when linking to outside datasets. For instance, the districts of the products were not necessarily politically recognized regions. These were not solvable using semantics, and had to be matched manually.

Even though the data are primarily in Norwegian, the districts seem to be represented in their own language. The exception being cases such as “Cognac tradisjonell”. GeoNames contains the name of each entry in a multitude of languages, any discrepancies in the language therefore seems to not be a problem.

Only the regions which use unorthodox names such as the “Cognac tradisjonell” has to be manually linked. This was done simply by finding the assumed region and then linking such pairs when iterated in the code.

On the contrary, the countries are all in their Norwegian names. Additionally, the countries might be represented with shorter versions of their name, such as “The Dominican Rep.” instead of “The Dominican Republic”. The geonames API did not seem to be able to handle these shorthand names. This means that I had to use some manual matching for these countries as well. The data was then downloaded in RDF form and included in the graph. Handily, this included a link to the Wikipedia article of each entry. Considering that the countries were in Norwegian, a middle point between the extraction and linking to DBpedia had to be used. The name of the countries was attempted matched against the overview of country names from (Oversikt av land, 2010), which contain information on countries, such as their Norwegian and English names. These names were then used to download RDF data from DBpedia.

The way the regions are were linked to GeoNames were by using the api.geonames.org to search for their names. For the regions, their respective countries were also provided to the API to narrow the search. Each search provided a larger set of results than just the ones that we wanted. These results were towns with the same names, smaller regions, etc. Because of this, the algorithm includes a set preferred types of entities. The one that matches this best is the one that is chosen. For regions this is preferred to be an administrative region instead of the seat of such region, the seat is preferred to a town etc. For districts, when they have been matched with an entity on GeoNames this entity was stored in a HashMap containing the name from Vinmonopolets data and the GeoNames entity. The names of the regions against the HashMap before looking it up at GeoNames to ensure the algorithm didn’t use query GeoNames when the data existed locally. Other datasets, such as LinkedGeoData and DBpedia were also considered. DBpedia was not chosen considering a weakness where the triples of a given entity are unpredictable; the regions won’t necessarily have coordinates associated with them. The LinkedGeoData dataset was not chosen because GeoNames had a SPARQL endpoint which offered some functionalities that could solve some of the CQs. The reason DBpedia was used instead of GeoNames for the countries was that DBpedia contain more information and seemed to contain coordinates for all of them.

4.1.4 SPARQL queries
The queries made for this case are fewer, shorter and more effective than the other cases. The difference here is that there are more math involved. This is discussed more in the discussion of case 1. The queries themselves are added in the attachments and include:

- An over/under query
    - This query finds every product that has a value between two numbers for a given predicate.
- Find beverages available close to a certain postal number
    - Creates a list of beverages available close to the given coordinates
    - Requires a way to find the actual distance in metres instead of the difference in coordinates, either post-query or an even longer mathematical query(which is difficult given the math provided in SPARQL).
    - When the distance is properly measured, this solves goal #4.
- Find the closest store that sells a given product
    - Proves that goal #3 is possible with some post-query work, similar to the previous query
- List nearby shops
- List amount of items from a certain region
- Average price in a region
- List of items from a certain region
    - Solves goal #2.
- Find coordinates of the region of a product
    - Solves goal #1 when combined with a map

The coordinates for the users are found by using the postal code of the user. This postal code is then found in the dataset from erikbolstad.no (Norske postnummer med koordinatar, 2014), where postal codes are listed with approximate coordinates. This is though not at all 100% correct, as it is community driven. It was however the best alternative I found.

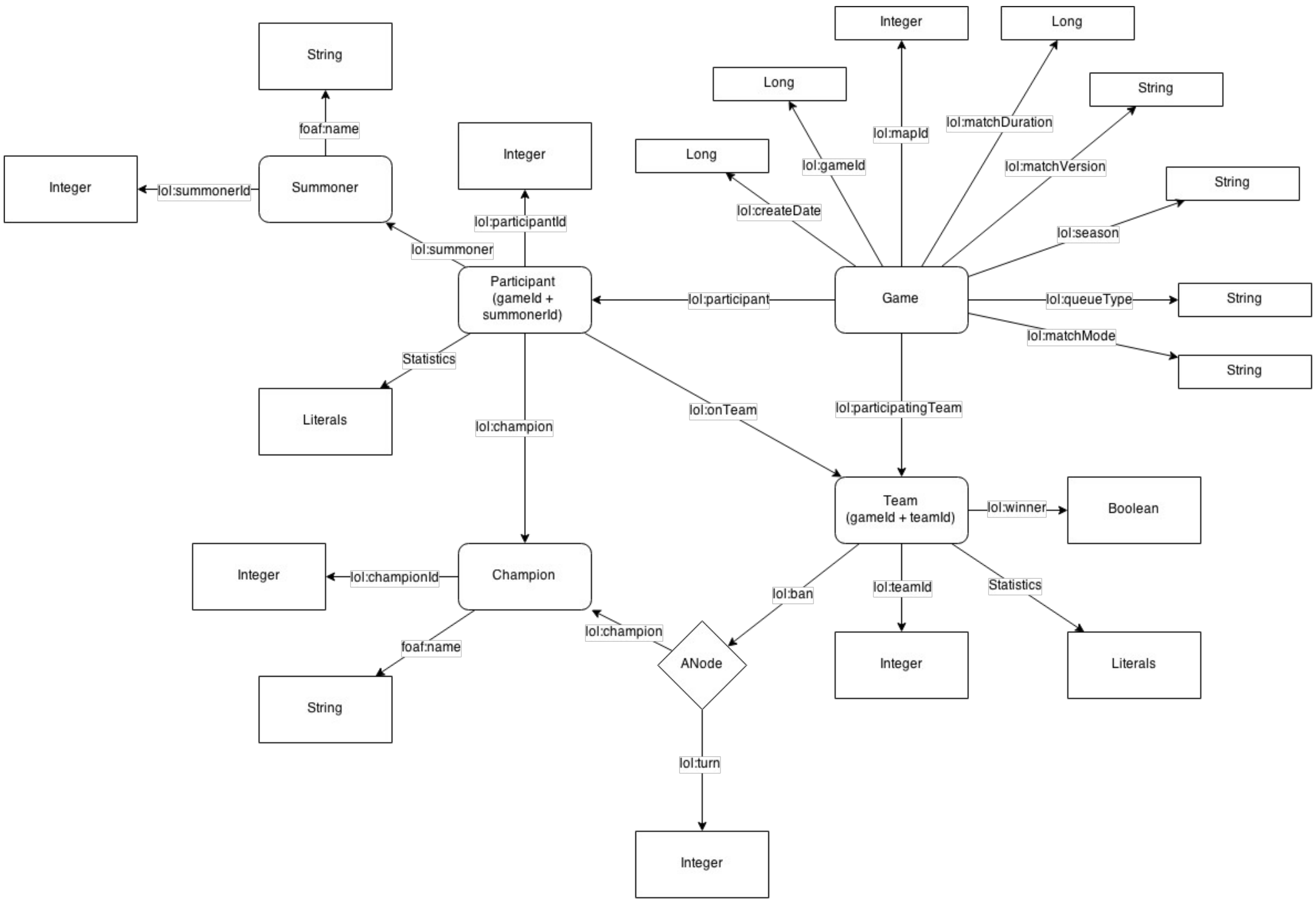
4.2 Case 2: League of Legends
This section will go into the work done on the League of Legends case. It will discuss how and where the data are fetched, the decisions behind the model, the problems and solutions of the lifting process and how the lifter continuously updated its data.

4.2.1 Data background

The data for this dataset is all gathered from the API(Riot Games API, 24-05-2015) provided by Riot Games, who is the creator of the game. The API is relatively new and is continually undergoing changes. Some of the objects available in the API are heavily based on the specific purposes they serve in the game’s client. It would for instance be logical to assume the “match” object represents one specific match. However, in reality it is a history of a player’s last 10 games. Additionally it only contains information on that specific players performance in the game. If you need detailed data about all the players in a game you have to call for a different object: “game”. The “game” object is the detailed representation of a specific game, expanding on the data from the “match” objects. These two objects holds many of the same data, but presents them under different names. Even the ID of the games are different. In the “match” object they are represented as “matchId” while in the “game” object they are “gameId”. The naming of these fields can create problems, and requires the lifter to be aware of these differences.

4.2.2 Modeling

It is evident that the core part of the model is the game object. The game object will contain some literals: map, the patch the game was played on, game mode type(queueType), competitive season, game duration, timestamp for the game and the id. These are all(except the id) information which can be used to further differentiate champions strength as it is always evolving during patches and competitive seasons. The model has adopted some of the ways the API represents the data. The game objects links to all players in the game as participants, which represents a player's participation in a game, instead of directly to a player object. The reasoning behind this is that the statistical data, which comes with each player in a game, needs to differentiate which game it is connected to. In contrast to the API’s structure, each team is represented as an own object instead of an integer connected to each player and has a “winner” predicate which was the case in the previous iteration. The team object now includes statistical literals, a winner flag and bans(champions which were not allowed to be played).



Graph 1: The League of legends model. Rounded rectangles represents resources, square rectangles literals and rhombi anonymous nodes. The statistics are not included in this image because of the sheer amount of literals(>130 for participants, ~10 for teams(both depending on map). The parentheses shows what is used to identify the objects where it is not its own ID.

The vocabulary used here is all created specifically for this model and are in the "http://www.uib.no/pam019/schema/lol" vocabulary. The only outside part is the foaf:name predicate used to describe champions and summoners.

The "queueType" predicate describes which queue the game used. With queue, the game means several things. It decides the map it is played on, the amount of players and whether or not the game was a ranked game. This means that there are a wide variety of valid objects for the queueType predicate. I considered making a hierarchy of classes to allow for easier queries, but when imagining what a query would look like, I realized it would not reduce the graph complexity enough to account for the fact that new queues are always introduced, removed and changed. When querying for specific queues, one will in most cases only need one line anyway, especially considering most maps or game modes which are useful for this application are a single queue(The competitive games are on the “Summoners Rift” map using a 5v5 format).

The “Game” class is hierarchical. It has three subclasses; “Matched_Game”, “Custom_Game” and “Tutorial_Game”. The tutorial game will be mostly non-existent as mostly new players play it. This is considering that the application will only download data from specific summoners or the top rated summoners. The class is therefore mostly in for clarity and to avoid errors when lifting. The Custom game class is originally another match type object, but it has very different applications from the Matched Game class. Custom games are often used for players who plays with a full set of friends and play either using normal rules or applying rules of their own. It is therefore completely unpredictable and not usable in any statistical way. The Matched Game is where players team up to play against an unknown team in a predefined ruleset and therefore the type of game we will want to use for statistical purposes. These classes are gathered from the “matchType” field and could have been modelled as a literal. It was remodelled to be the class of the games as it is the most high-level representation of the games, above matchMode and queueType, in that order. Being hierarchical it might shorten the queries in some cases, as I recommend selecting the game classes using the class, even though it could be identified by its predicates.

4.2.3 Lifting

The data needed for the competency questions to be met is in reality only champions, games and summoners. All of these data can be found from three objects: champions, game and match. But we also need to find the default set of players to follow, which is the Challenger tier(top 200) players in every region. This means that there will be approximately 1000 players followed, and the amount of games will quickly become large. To do this the league/challenger object has to be used as well. The API has as mentioned a limit on the amount of requests per 10 seconds. This had to be handled during the lifting, but did in the end not affect the dataset directly. It did, however, affect the amount of games that could be downloaded. This is the reason that only the top players are downloaded instead of every game that is played(Riot Games API, 31-05-2015).

The evolving nature of the API is evident from its documentation. The game and match objects(from now on talk about match as detailed game and game as simple game), which represents the same object, has a lot of inconsistencies. For instance, the “queueType” from detailed game is equivalent to “subType “in simple game, “matchMode” and “gameMode” are equivalent objects, etc. This, in addition to the frequent downtimes the API goes through, shows some of the of weaknesses which have to be taken into account.

The reason the simple game object was needed was not only to fetch the ID of games. The detailed games, that were fetched using these IDs, only contained information that could identify summoners if the game was ranked. The simple game however, contained identification information on all summoners. The simple game object’s list of participating summoners contained both champion and team, which meant it could be used to match with the detailed games participant data. The only time this would not be unique would be if the match mode were “one for all”, which is a game mode that at the moment does not exist. For this reason the code had to filter out the process of matching participant and player in this mode.

When lifting data from JSON to RDF, the process used a set of try and catch statements. This was needed because each map contained different fields in their statistics. Had there been clear differences in the fields on each map, then this could have been done with different lifters for each map. However, the map’s data are intertwined, meaning that some data exists in several maps, but each map also have some unique features. The use of try and catch to divide the maps was therefore found as the best solution.

Considering that there were no outside linking, this process is rather simple. It merely fetches the data and transitions it to RDF.

4.2.4 Continuous updates

The lifting process in this case is a little different from the other cases. The other cases lift the data once and then leave them. This case on the other hand needs to be continuously updated with data to stay relevant. While this is true for the other cases, it has not been prioritized considering that the lifting process would be more or less the same(new donations in case 3 could be added afterwards, but the rest of the data would still be gathered in bulk just at a different time). In this case however, since there is an API that allows us to see which data are new, this process is a little different. The data-extraction and the data-lifting are divided more heavily in this case. The program has a list of players which it asks the API for the match history of. From this match history it continues to ask for the detailed data of the games that don’t already exist in the lifted data. The detailed game data are then lifted separately, and then stored in a temporary dataset. The temporary dataset exists to minimize the time the main dataset is busy with adding new matches. This makes the program usable by the user while the new data are gathered(considering it is done on a different thread). When the program is started and at a set interval, this process is repeated to ensure that all games are lifted. Since the match history of each player only contains the last 10 games, this process should be done around every 4-5 hours to be sure to fetch all games.

4.2.5 SPARQL queries

The queries in this case are somewhat different than the others. There are really only one query for the whole case. This query finds both win rate and amount of games for each champion. The query has to allow for the insertion of a wide variety of options. If the user wants to only see games that are from a certain queue, or that contains several different champions, the query needs to be able to handle this. This is done by creating the query in the Java code. The function that creates the query is passed all the options that are chosen by the user, and inserts these into the query along with the other statements required for it to work. This is by far the hardest query of the cases, as there are very many different options that needs to be accommodated for. Even though a new model was created for this research, the options in the old application are the only ones included because of time limitations and the fact that they are enough to reach the goals of the research.

4.3 Case 3: Public datasets

This section presents the work done in Case 3: Public datasets. In contrast to the other cases, this section has more individual datasets that are combined. The partifinansiering.no-, brreg.no-, election- and municipal-data will have their own subsections, while some of the smaller datasets are combined in the larger ones. How well done the connection of the data was is discussed in section 5.2.

This section will cover the lifting process of:

- Partifinansiering.no
- The Brønnøysund registries
- Election data
- Municipals from 2000-2013
- Regional data from Statistics Norway

4.3.1 Partifinansiering.no

This was the core idea for the thesis itself, as such it was prioritized to be lifted first along with the BRREG and valgresultat datasets. These data were in a previous class project scraped from the website's HTML even though large portions were available in yearly Excel files. This iteration uses these Excel files instead of scraping. This is because the Excel files are a more structured, only contains the data we want, and are easy to extract the data from. The data concerning the actual donations on the other hand are only available through the site's HTML and therefore both approaches had to be used.

These datahas a lot of weaknesses: The archive is up to 9 years old and unrevised, meaning many addresses could have changed during this time as well as names of companies.

4.3.1.1 Data background

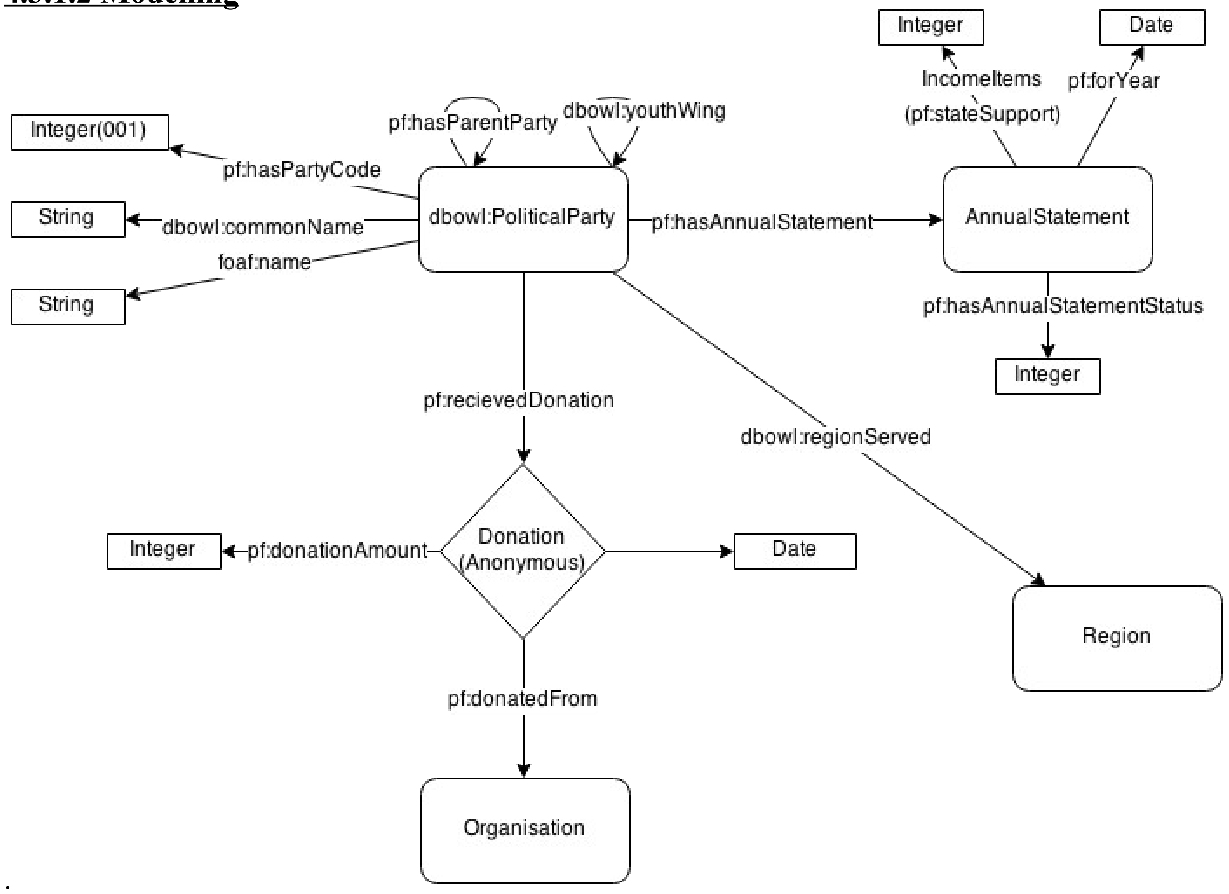
In Norway, each registered political party has to file an individual report stating its income once a year. This report segregates income into these types:

- State support
- Municipal/County support
- Other government support
- Subscription fees
- Income from fundraising
- Capital income
- Business income
- Donations from private individuals
- Donations from commercial organizations
- Donations from organizations in labor market
- Donations from other organizations
- Other donations
- Transfers from other party units
- Other income

(Kommunal- og moderniseringsdepartementet, 2015)
The parties also have to report donations that give over 35 000 to their main organizations, over 23 000 to county organizations and over 12 000 to municipal organizations as of 2013. These numbers has been lower in earlier years (Lov om visse forhold, 2014). The report consists of several more points, but these are not present in the data from partifinansiering.no and as such that data are not included in this process.

The law states that a contributor is to be identified with name and postal address(non-individuals have municipality instead of postal address). This is the only identification that is required in the report. As these are handled manually by the political parties themselves, this means that many of the organizations listed as contributors have incorrect spelling, their names being an abbreviation of the full name, or are listed using different standards(AS, A/S or A.S etc.). In addition there are many inconsistencies with uppercase and lowercase.

<u>4.3.1.2 Modelling</u>



.

Graph 2: The model of the partifinansierngs data

The model is as shown in graph 2. I have chosen to show each of the dataset's models individually instead of the combined version. This is because the combined version is very complex, but the structure is shown well enough in the individual ones

PoliticalParty objects are identified using their identification number from partifinansiering.no. This is because it can be derived directly from the data, without linking to other datasets. There exists different identifiers in other public datasets, but I have elected not to use these as the ones from partifinansiering are both intuitive, easy and unique. The youth wings of the parties use the same notation, only they start with a 1 rather than a 0. For instance, KrF is identified as 001 and its youth wing as 101. The URIs for this model can be found in the attachments.

Annual statements build on their parties' identifiers by adding a year to the end of the URI. This is unique as the statements are only sent in annually.

The income posts predicates are declared specifically for this project, as there exists no fitting vocabularies for them.

The donations are represented as anonymous nodes. They are identified by the year they were made in, the party they were given to, and organization or person that gave them. At the time this was lifted, this was unique, but this might change as more donations are made. however, it does give us the clue that the donations are being summed if there are more than one from a contributor. The decision to use anonymous nodes is based on the fact that not all the contributors could be identified, and therefore the use of organization number could not be counted on for identifying the contributor. For the private individuals that donate, they have to be identified by name. The Norwegian laws state that any collection of personal information has to be reported to the NSD. The NSD has laid down some rules saying that these data has to be made anonymous after this thesis is finished(only if work continues). This would create a wide variety of numbers representing private persons, names of businesses as well as the same businesses misspelled or abbreviated. The form of the URI would not be easy to understand for other people using the model, and therefore I let the donations be identified by its triples.

After looking through a set of already existing vocabularies which describe regions, I have decided to create my own to describe municipalities and counties as the existing ones does not have a proper representation for counties or the ones that do are not hierarchical, such as the ones from http://vocab.org/places/schema.html.

4.3.1.3 Lifting

The process of lifting this dataset consists of two independent stages, one for lifting the annual statement data and a separate process for the donation data. The statement data are available in Excel files, while the donations data has to be scraped from the HTML files. Because of how intertwined the donation data and BRREG data are, it is in practice the same lifting process. The donation data are scraped and structured first. Then the process attempts to link it to BRREG data before it lastly lifts all the data simultaneously. In this section I will talk about the lifting of statement data and the data gathering of the donations data. The BRREG section will discuss the linking of the two datasets.

I used the xlrd library for Python when lifting the statement data. The library makes reading the Excel files significantly easier. The Excel files contain data such as the party-, county- and municipal-identifier, statement status(delivered, not delivered, below threshold etc.) and all the income posts in addition to the name of the party. The name of the party is on the lowest level, meaning that the main organization of the KrF party would be represented as “Kristelig Folkeparti Hovedorganisasjon”. Therefore there is another sheet that contain the names of main parties and their respective party identifier. Since each main party can be represented as either their normal name or including the string “hovedorganisasjon” etc., the names in the different sheet will be used as the common name of a party. This means that, even though a main organization's name is featured in the foaf:name triple, the common name of a party is declared in the “dbowl:commonName” triple, which is shared by all the children and main organization of a party in addition to the foaf:name which represents the local name.

The process of lifting the Excel files were rather straightforward. Each party only had to be connected to their main party and the regions. These have been represented with the URI format of "http://www.uib.no/pam019/regioner/" plus the county and municipal concatenated. For instance, the municipal of Bergen would be represented as: "http://www.uib.no/pam019/regioner/1201" where “12” represents the county and “01” the municipal. This is the standard way of representing municipalities and they will always appear with the county id first. The county Hordaland is represented as "http://www.uib.no/pam019/regioner/12".

The main political parties are identified "http://www.uib.no/pam019/politiske_partier/001" where “001” is the party identifier. Linking a child party to its main party is done by adding the municipal and/or county identifiers at the end of the main party’s URI, as in partifinansierings identifiers. The identifier of the main organ of a political party is as mentioned stated on each line of the Excel files. The sub organizations are represented as for instance:"http://www.uib.no/pam019/politiske_partier/0011201", which would represent Kristelig Folkeparties(001) organization in the municipal of Bergen(01) of the Hordaland county(12).

The income posts and statement status were only a matter of matching a column in the sheet with a given predicate as declared in the model. This process can be made more streamlined by using the triplify method described in section 5.1.3.4.

The donations were more difficult as they had to be scraped from HTML files. Attempts were made to get the data in a more convenient format, considering that the statement data were available in Excel files, but the SSB(which holds the data from partifinansiering.no) were unable to help because of § 2-6 of the Norwegian law states they can not give out information on individuals.

With 428 municipals, 19 counties and the political parties being able to have local parties in each of the regions as well as one on a national level combined with the data being divided into yearly files (with 9 years) means that there are more than 29000 pages to scrape. Because of this it was decided to download every page so that I could do the scraping locally. The data from previous years will not change, so this will make the process a lot faster and relieve the server of a lot of traffic.

It is apparent that the archives have gone through a series of changes to the structure of the HTML files throughout the years. There are numerous differences between the structure of early years and the current years. Differences depending on which types of donations have been made also exists. Files which represent parties that have gotten donation from both private individuals and organizations have a different structure than those with one or the other.

A file without individual donations does not include the structure for such donations. Additionally, the tables that hold contributions from organizations are differently structured depending on whether or not the party also received donations from individuals and vice versa. This is the case even though the two types of donations are held in separate tables. The only way to create a correct overview of all the different structure variants, I would have to manually look through all the files HTML code. Considering the massive amount of files in the dataset this would though be impossible within the timeframe set for this stage of the process. However I have sampled all the different structure in the files that I could find to create the best possible scraper.

In the HTML files any donation from an organization is grouped together with other non-registered organizations. With non-registered organizations I here mean groups that donate but that are not registered in the Brønnøysund registries. This can include organizations from other countries, bands(though some are registered in the registry) and fundraisers. This means that not all the donations can be linked to an organization and therefore the % which is shown in section 5.2 is more accurate than what it seems. For these donations the contributor is represented using their name in the URI.

4.3.2 Brønnøysund registries

This is the largest dataset in this case by a substantial margin. The data from partifinansiering might have a lot of files, but the data they contain are dwarfed compared to this dataset. It contains not just all of the organizations in Norway, but organizations that have succumbed to bankruptcy or otherwise shut down. The data are available in three different ways: the BRREG website, where its users can search for organizations with similar names; an API with a few different methods of retrieval, but without the ability to search for similar names; and in bulk files. Both the first two methods were chosen for this process. This was because it gave an advantage considering the data are always up to date when you fetch it as well as a way of searching for organizations which are misspelled or otherwise not correct from the other datasets. This does have a downside since the lifting process needs to include a way of correctly reading data from the HTML on the websites; which introduces much more problems than reading from a CSV file. The fact that reading and matching from a CSV file to another well structured source is done in case 1 means that the method I will propose is based on a wider set of experiences. It should be mentioned that there are flaws in this process, as discussed in the discussion of this case later on.

I have included the model and lifting process concerning industry codes in this section. This decision was made considering the industry codes are only relevant in combination with BRREG data. These data are gathered from SSB (Statistics Norway, 2014) and are an overview of the different industry codes and the hierarchical structure of them. It contains information in the form of strings which describe what is included, excluded and titles; all stored in Excel format. Additionally, the postal code data are also lifted in this case. The postal code data represent what type of location each code is, such as postbox, street address etc. Additionally it identifies the municipal of each postal code. While postal codes are used in donation data as well, the PF data already include the home municipals of individuals, and as the organizations data are linked to the BRREG data, these data are only required here.

4.3.2.1 Data background

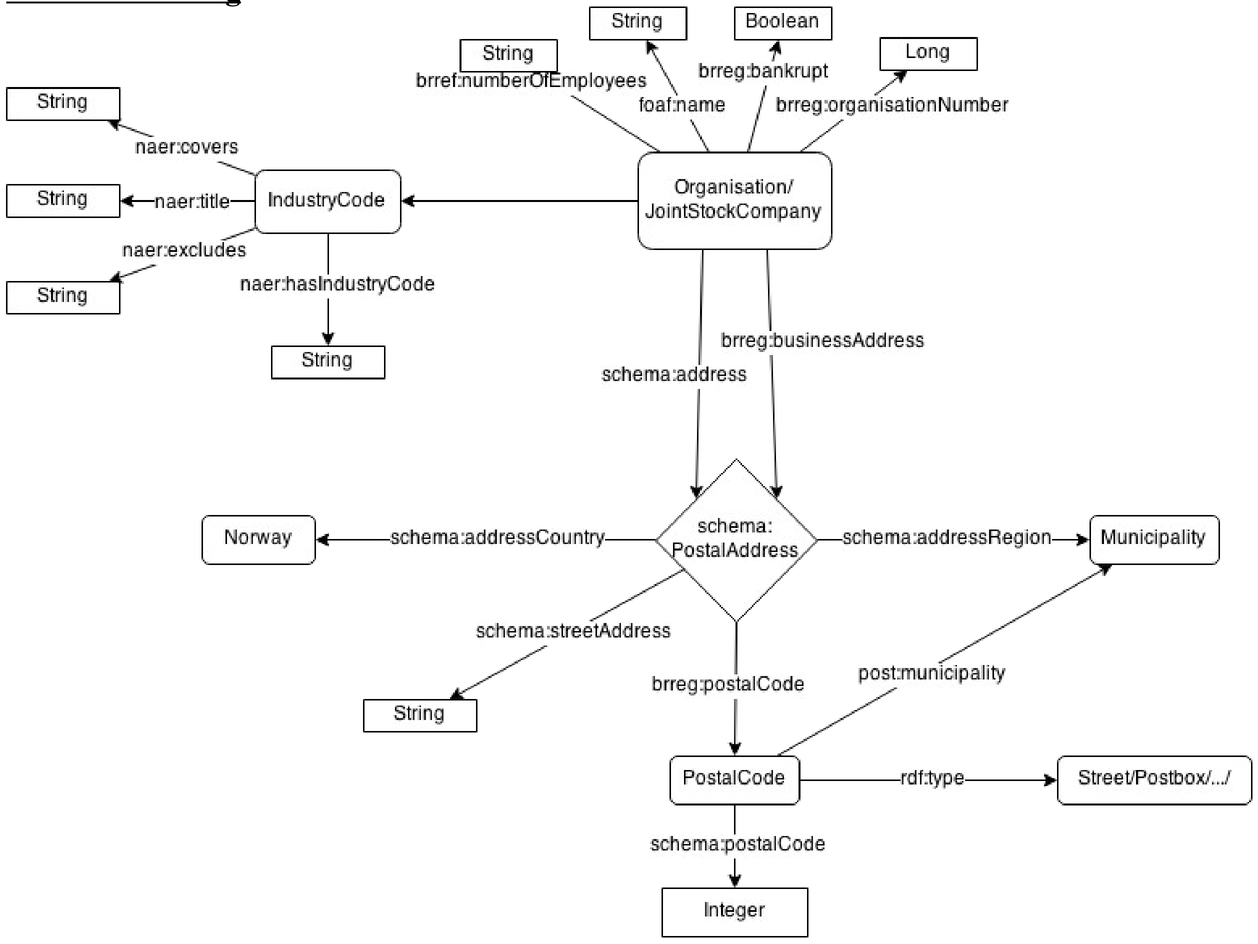
The Norwegian law states that the government has to create a registry that has the responsibility of gathering information about: the state, counties and municipals; companies, foundations and associations; and sole proprietorships. (Lov om Enhetsregisteret, 2011) The registry is required to contain information about organizations such as:

- Name
- Address
- Organisational form
- Industry
- Date of establishment
- CEO

The result of this law is Brønnøysundsregisteret.

In addition to the fields listed above, it contains information on the organization's employment count, signature, board members and more. These will only be present in the data if known for that given organization. This means we can only be sure that the first 6 types mentioned is present in the data and therefore it must be taken into account in any SPARQL queries used on the dataset.

4.3.2.2 Modelling



Graph 3: The model for the BRREG data

The organizations have the class “Organisation” or a subclass of this class, such as “JointStockCompany”, “PublicLimitedCompany” and “SoleProprietorship” among others. The literals connected to the organization are not all the potential ones but instead the ones that are included in most organizations(except numberOfEmployees). The complete list of information that can be included can be found at "https://confluence.brreg.no/display/DBNPUB/Informasjonsmodell+for+Enhetsregisteret+og+Foretaksregisteret".

The "schema:address" and "brreg:businessAddress" predicates represents two different addresses, but both need not be present in the data about an organization. They are only separated to reflect the difference depicted in the data from BRREG and therefore share the same structure. However, both addresses are of the schema:PostalAddress class.

4.3.2.3 Lifting

This section covers the lifting process of the BRREG data. It is assumed the data from partifinansierng.no is already scraped.

There were several ways to fetch data about the organizations on BRREG. Using the data dump available from data.brreg.no, the API on data.brreg.no or scraping the search function could all be considered. The data dump's pros are that we receive a lot more data in a very easily converted format. its cons are that if we were to continually update the data, it would take a lot longer to lift and use more bandwidth. However, the reasons I decided to use both the API and scrape the website, was because doing so means I do not have to create such a search functionality, minimizing the work while creating a wider set of experiences as data dumps are explored in both the SSB data, NSDMD data, and in case 1. In a real world application the data dump might be a better option as it provides more data that could be used in this application.

The API has a set of commands that can be used to find data. The full documentation can be found on "https://confluence.brreg.no/display/DBNPUB/API". The API has two different endpoints. One for main entities of an organization and one for sub entities(child organizations). Since we do not know which type the contributor is, the algorithm first queries the main entity endpoint. If no match is found, the algorithm then queries the sub entity endpoint.

The contributors from the partifinansierings data are as mentioned only presented using their name. It could be an abbreviation, the full and correct name or a combination of the two in addition to the address of the organization at the time of the donation or financial statement.

The API does not support regex functions but it does support filtering by using keywords such as "startsWith", "ge"(greater than) etc. This is not sufficient since there exists abbreviations in the dataset in addition to a large set of misspellings, especially in names, and therefore the API calls only search for names which are exactly alike. This is the reason the search function from the website will be scraped as well since it can search for partial names. The process of fetching data on contributors is as shown in the following pseudo code:

```
for each organization in donations
    if organization is in list of known name mistakes
        look up organization in API using organization number from list

    if no data found OR organization not in list of known name mistakes
        look up organization in main entity API using name
        if no data found
            look up organization in sub entity API using name

        if no data found
            scrape the web search function using name and address
```

The “known name mistakes list” is a file created manually in the beginning to match abbreviated organizations with their full name counterparts. It makes for a bit more manual work, but is required if a high amount of matches are to be done. The abbreviations are usually for organizations such as LO(Landsorganisasjonen) with different sub organizations listed as for instance LO Vaksdal. There appear to be little consistency between sub organizations names on brreg.no and partifinansiering.no. The API and search function are not optimal when handling abbreviations. For instance the organization “Kommunesektorens organisasjon” is in the registry listed as “KS”. However, any searches for this abbreviation, or the original name returns no results (the organization number was only found from its website) and therefore the manual matching is even more required.

After searching for data on each individual organization, the process will attempt to group together the different donations that have not been linked to BRREG. This means that if two donations have been made by the same contributor, these will be grouped together. This matching is done by checking whether or not the addresses match and using a levenshtein ratio to find the likeness of the two names. It is shown in pseudocode below.

```
for each donation in donations as donation1
  for each donation in donations as donation2
    if donation1 is not donation2
      if data from BRREG exists for either donation1 or donation2 but not both
        use addresses from the API data for the donation that has data from the API
      else
        use addresses from the pf.no data

     if either the address or business address is equal for both donations
      and if the levenshtein ratio is higher than 0.7 for contributor names
        group donations on contributor with data from BRREG

      else if the addresses is not equal but the levenshtein ratio is higher than 0.9
        group donations on contributor with data from BRREG
```

Only one of the donations is allowed to have data from BRREG when grouped together. This is because donations that are alike and have data from BRREG will be modeled into the same RDF resource anyway. This is a tiny step in removing false positives. The addresses used in the grouping algorithm are the ones fetched from the API where such exists and from partifinansiering where they do not. This ensures that the addresses are correct as of the time when the API were queried. I ran the matching algorithm with varying numbers in regards to name similarity to find the best one. The best one is the one that has the least false-positives while having as many true-positives as possible. When checking the positives from the algorithm, the only false-positive from this variation has been on two organizations whose name differed by only one letter.

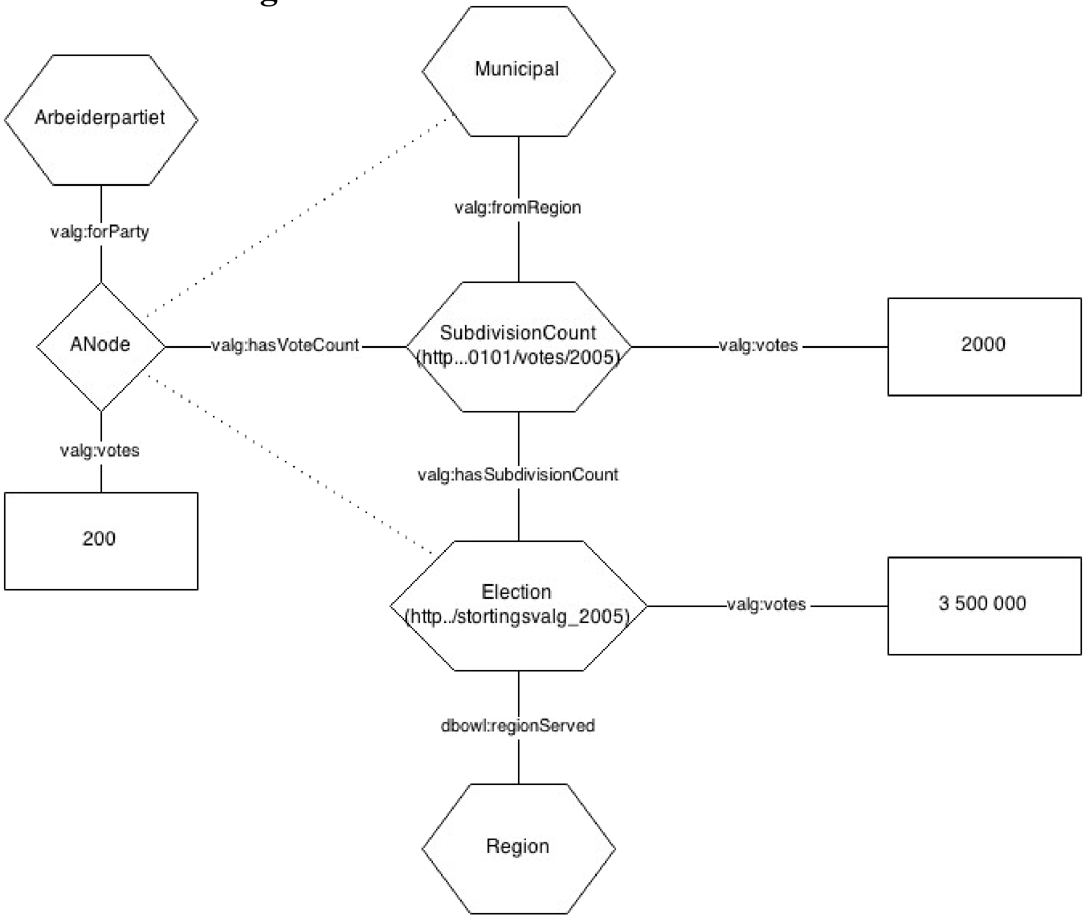
Various other issues existed in the lifting process. One of these stems from the addresses of the partifinansierings data being differently formatted. They could include the owner of the address, be postboxes or street addresses and contain any of these data in any order. While being a solvable problem, it needed to be analyzed thoroughly so that we can match addresses as well as possible.

Even though the API divides its search into sub entities and main entities, some of the main entities are also sub entities. To clarify, the hierarchical structure is only one level deep, therefore the levels below(but also including some in) counties are not listed as sub entities, but rather main entities with no ties to the main organizations. Because of this I created a way to match organizations with their parent organizations. This process consists of matching a set of fields on certain criteria. These criteria includes checking if the name starts with a certain string, the industry code and on the sector code. These were specifically made for LO but can be adapted to other uses. However, I have not placed too much effort into this as it did not help towards any of the goals of the research.

4.3.3 Election data

Originally, the election data were to be fetched from valgresultat.no, but as access to the NSDMD was gained, those data were no longer needed. In fact, the data from the governments websites are only estimates while the data from NSDMD are taken from SSB's database and are the actual numbers. In addition there were data from more elections than from the government website(in this context government website refers to "regjeringen.no", as both are government run public websites). The data gathered is only from elections of 2005 and later. The reasoning behind this is that the earliest data on partifinansiering.no are from 2005.

4.3.3.1 Modelling



Graph 4: The model for election data

The dashed lines are from an alternate version which I considered using to avoid the “subdivisionCount” object. The reason the subdivision count exists is to allow the model to have a literal for total votes in a municipal. This is partly to accommodate the fact that it exists in the original data and to avoid queries having to use math, which would become an entire subquery. To clarify, in the dashed lines version both the subdivision count and the valg:votes triple from it would be removed. The “regionServed” predicate is only in for reuse and clarity, since the elections lifted are all on a nationwide scale. This is considering that every “regionServed” predicate points to the “Norway” resource.

The “Election” class is the parent of the class “GeneralElection”. This is the only type used in the data, but it is shown in the graph as “Election” for the sake of clarity considering it is designed to be used for all elections. With the case of other elections, such as elections in municipals, the “subdivisionCount” object is indeed unnecessary. The alternate version discussed in the previous paragraph can be used in these election without subdivisions.

4.3.3.2 Lifting

The data are as said from the NSD municipal database. The data are gathered in CSV format, which makes the lifting process relatively simple as there are no connection other than to the political parties. The table columns are simply associated with a given predicate which is placed in a triple with the election resource and its respective cell. Each election have several files, one for each political party, which contains the set of votes from a given municipal to this party.

4.3.4 Municipals from 2000-2013

The dumps containing information about municipals show data for a given year. The dumps for all the years in 2000-2014 have been fetched and compared to one another and used to find municipals that are established or removed in this period. The point of this is to include a way to filter out municipals which no longer exists in the timeframe of a query. This can be necessary as (especially) the industry codes queries are very time and resource consuming. Displaying the information using tools such as SGVizler was also part of the paper at some point, this added to the need of these data as SGVizler only accepts SPARQL queries without any post processing.

The way this is done is by looking at two and two years at a time and if a municipal exists in one but not the other, a triple using the predicate “dbowl:discontinued” or “dbowl:established” is added to the model.

4.3.5 Regional data from Statistics Norway

These include:

- Municipality area
- People employed in the different industry codes in municipals and counties
- Population count for counties and municipals
- Average income for counties and municipals

The data are all in CSV format since the structure is rather simple and one line can be directly linked to a municipal. The only data that needs to be linked to other datasets are from the industry codes dataset where we link the ones listed in SSB’s data to the ones lifted earlier in correspondence with BRREG data. This did though provide problems since the two datasets grouped codes differently. For instance the grouping from the BRREG data included the codes 35 and 36-39 separately while SSB grouped 35-39 as one.

4.3.6 SPARQL Queries

This case produced many SPARQL queries to solve its goals. Some of these were long, complex and resource consuming. Because of the amount and size of the queries, these are included in the attachments of the paper. The queries finds:

- The sum of donations and amount of donations from organizations within a certain industry code to a political party.
- The sum of donations and amount of donations from organizations within a certain industry code to a political party(includes local party’s donations). Includes average donations.
- Sum of donations + amount of donations from organizations within a certain sectorcode to a political party(includes local party’s donations). Includes average donations.
- Donations from private individuals based on where the person lives. Grouped by party, year and municipal of the contributor.
- Donations from private individuals. Includes the population percentage that donates, as well as a how much is donated per person. Grouped by municipals.
- Industry codes with the most employees per municipal.
- The 3 largest industry codes in municipals(very resource consuming).
- The municipals average income and political parties with the most votes in these municipals.
- List the political parties and the average income in the municipalities where the party had the most votes. Lists results from general elections.

- Shows the average income of the municipalities grouped by industry code.
- Which parties win in the most municipals with different largest industry codes.
- Percentage division of parties in municipals grouped by the largest industry code.

5 Discussion

This chapter opens with the proposed lifting method, then goes on to discuss the individual cases, their contribution to the process, any improvements that could be made, and what was of adequate quality in them. After the individual cases some ideas that could benefit all cases are presented and at the end the research method will be evaluated. The chapter will discuss the work done, what could have been done better, as well as the statistics of entities successfully linked to outside datasets.

5.1 Proposed lifting method

In this section I present the method for lifting data to the semantic web. This is based on my experience in lifting the cases in this thesis. Considering the differences between the cases, I believe it as a method can be used on a broad variety of data and will discuss the differences needed to be made with different data structures.

The process consists of 4 stages: goal specification, model construction, data-lifting and evaluation. It should be perceived as an iterative method; the average process will probably be involve 2-3 iterations. The goal stage *can* be skipped after the first iteration. The process is finished when all the stakeholders agree that the goals have been met to a satisfactory extent.

5.1.1 Goal specification

The first stage specifies the goals of lifting the data. This stage aims to end with a set of well-defined goals which have to be fulfilled at the end of the lifting process. To capture the goals of a given dataset one has to examine how the data are used now, what the meaning behind the data are and what the data represent. This is an important part of the process as the other stages are dependent on the goals to be able to lift the data correctly. At the end of this stage we are left with a set of objects that represents the most important aspects of the model and a set of competency questions(CQ).

Stakeholders often have their own visions of what is to be achieved by the system after the lifting. This is usually at a high level and therefore not detailed enough to be transformed into an effective model without more detailed analysis. These visions can therefore function as an overarching goal and it is important to have these in mind in all stages of the process(Wieringa, 1996).

5.1.1.1 Existing functions

When performing a lifting process on a set of data, the data are in most cases not new, but rather a part of an already existing system. It is therefore important to make sure the existing functions of the system are kept intact and not changed unless absolutely necessary or decided by a stakeholder.

Stage one begins by figuring out how the data are used in the application/dataset. This includes simply displaying pages for entities, searching, which parts are visible to the users and how they are presented etc. Some data might be displayed assuming the user has knowledge about the field and some might be simplified.

As an example, the list below shows potential functionalities already existing in Case 1: Vinmonopolet:

- The website should inform the users of which products are available
- The website should informs the users on where these products can be found
- The website should show information on each product(rawmaterials, region, method, produced etc.)
- The website should let the user choose a preferred store and see which products this store sells
- The website should let the user search for products using the fields(rawmaterials, region, name etc.)
- The website should let the user filter the products using a set of predefined fields
- The website should show the location of every store
- The website should inform the user of which food each product complements
- The website should show which products are only available online or by order

5.1.1.2 Competency questions

The next step is to create the actual competency questions. The CQs should include the functionalities found in the previous section. Additionally, the questions should include any new functions. For Vinmonopolet the new functionalities that were wanted from the lifting would now be:

1. Be able to display the origin of products on a map(Find the coordinates of its origin)
2. Be able to display find products originating from a given region
3. Be able to display find the closest store that sells a given product
4. Be able to display show which products stores within X-km sells
5. Allow for improvement in the search and filter functionalities
6. Retain all the old functionalities

The CQs are often the goals set by the stakeholders. Sometimes these can be included directly but often these goals need to be broken down to more specific functions.

These points should always be in the back of our minds when the model is created. This can help with creating models that allow for more effective SPARQL queries. If the model requires very complex queries this might make for very long execution times. This is especially important in systems that are meant for consumers. According to Bharati and Chaudhury(2004) the loading time and response times are important for how the user perceive the system quality; however, information quality is more important than the system quality.

5.1.2 Vocabulary/Ontology defining

The next stage is the process of creating and identifying different vocabularies to form the model. This section will use what I have learned to describe a proposed process for this phase.

The phase has been divided into a set of stages:

1. Understanding the data
2. Identify the main concepts
3. Linking data
4. Evaluation of existing vocabularies
5. Building the model

5.1.2.1 Understanding the data

When creating a model, it is not enough to understand what the data are, one has to understand what information it tries to convey and how it attempts to do so. For most of the data this is relatively easy, however, in some cases it might be more difficult. One instance of this is in Case 1: Vinmonopolet. The stores are here divided into categories from 1-7. These categories tell what size the store is, as well as what products the store has available. Each product is linked to one of these categories to show where it can be bought. These categories inherit from each other, meaning that category 7 stores include all products from category 6, category 6 includes all products from category 5 and so forth.

This might sound like a simple example, but it is something to think about when creating the model. Should the model represent these categories as classes that inherit from one another? Use literals to state the category? If one were to use classes that inherit, one would end up with a semantic problem. If one were to ask for every product that a category 7 shop sells, one needs to include the ones from category 6 as well. In a SPARQL query on a model using inheritance, the query would include something like this:

```
?beverage vin:shopcategory ?category.
?category rdfs:subClassOf vin:Shop_Category_7.
```

This means that category 6 would have to be a subclass of category 7, which semantically speaking would be wrong, as category 6 isn’t really a subclass of category 7, rather, category 7 builds upon category 6. In this instance the categories are represented as classes that are linked together using the skos predicates skos:broader and skos:narrower.

This example introduced some problems to think about when creating the model. I can only assume that in more complex datasets more problems like this would occur. For these reasons it is important to take note of the whats and whys of the data being lifted.

This stage of the lifting process should also include going through the data to find any anomalies in it. With anomalies, I here mean data that in some way or form are not what is expected. For instance some of the donations in case 3 was made from outside of Norway, which was not expected. As these can not be expected to be linked to the Brønnøysund registries, if these cases were not found, the linking to BRREG would seem to be less effective than it actually was. Another example is to see through the different regions that are listed in case 1. Seeing many of the Cognac products listed as being from “Cognac tradisjonell” instead of a recognized region, it could be assumed that the area are Cognac, France was all listed as this region. This was a mistake and some of the sub-districts there did exist in the data.

5.1.2.2 Identifying main concepts

This stage of the construction has the end goal of understanding which concepts are the most central ones in the data as this can be an important part in creating a correct model for the data. As discussed in Suarez-Figueroa *et al*.(2009) and Studer *et al*.(2007) you might not be able to model everything optimally and therefore instead should focus on modelling the core concepts correctly. This is of course dependent on the complexity of the data/model. Not all of the *concepts* in this regard is obvious, some are hard to spot. An example of this is in the League of Legends case, where a game links to a player and a champion. Linking these directly would have created problems considering that the statistics are linked to a player's participation in a game. Because of this there had to be introduced an anonymous node to represent the given player's presence in each game. This is something that might not be obvious before all the data present for the game, players and champions have been examined.

5.1.2.3 Linking data

The requirements developed in phase one are in this stage used to identify the linking points between datasets. The term “linking points” will in this context be used to describe objects that exists in two datasets. These objects can be anything from strings or id numbers to objects described using semantic technologies. Identifying the linking points means finding and describing the fields or data which contains the information that is to be linked, as well as finding the *most suitable* dataset to link with. For instance the regions of Vinmonopolet can be linked to several other datasets such as: LinkedGeoData, DBpedia and GeoNames. The decision of which one(s) to use depends on what you want from the linking. In this instance the data that was sought to be included were ultimately coordinates and a Wikipedia description. The decision eventually fell on GeoNames since it contains coordinates for all of the entries as well as a link to the Wikipedia article and DBpedia page where present. DBpedia itself was not chosen because it contains some discrepancy regarding properties in its entities; there is no guarantee that the resource contains coordinates. In addition it contains resources describing anything, not just geographical data, which means that the probability of false linking is a bit higher. Combined with the fact that there are differences on properties and even classes, the GeoNames dataset was perceived as a better choice for linking. I would from this example recommend as a general rule that the most specific dataset is used when linking data.

In this stage one should focus on identifying potential new datasets and the resources that are sought within it rather than the linking itself. The linking process is discussed and done later on in the process.

This stage is important when considering the interoperability of the data. When linking the data to outside datasets, the interchangeability of the data are important. When the data are being lifted in the later stages of the process, the process has to know how the data are represented in comparison to other datasets. If the previous stages of the method have not been fully completed, one can expect to find some syntactic interoperability problems(García-Castro *et al*, 2010),

5.1.2.4 Evaluation of existing vocabularies

The vocabulary describing the model should be standard or predefined if possible. This is mostly decided based on the semantics of the vocabularies and the classes and properties they define. If the vocabularies define properties which have the domain of a class you could use, but the range of entities which are not applicable to the data you should use, you may want to consider other vocabularies. For instance the

WineOntology defined by the W3C (WineOntology) defines the regional property as having a range of predefined areas. Considering that the vinmonopol dataset includes many districts and subdistricts not present in the predefined list, in addition to having an issue where the characteristics are not represented as the integers 1-12, it was decided not to use it and rather connect them using owl:sameAs and rdfs:seeAlso.

5.1.2.5 Building the model

When the stages leading up to this is done, most of the work creating the model is already done. What remains is defining your own properties and classes then combining it into a model.

When building a semantic model the use of visual representations of the model can help give an overview of the model. In my experiences using visual representations in this stage can be helpful as a starting sketch. After the graph is created in concept/visually, I used Protègè a to create the actual model. Protègè offers visualization of the model as well, but in my experiences it has not been implemented well and therefore I have either visualised by hand or using draw.io diagrams. Since Protègè is intended for OWL models the user has to be aware of if this is what the model uses.

5.1.3 Lifting

5.1.3.1 Tools

Semantic technologies require us to use some tools to handle and create data. There are several technologies which allows us to edit, create and explore the data.

Protégé, an application created by Stanford University, is a "A free, open-source ontology editor and framework for building intelligent systems" (Protégé, 30-04-2015) . It allows for creating ontologies in the full OWL 2 language, which makes it very powerful. (Protégé, 30-04-2015) Protégé has for instance been used to create the models of each of the cases in this research.

Apache Jena is a open-source Java based framework. It is probably the most advanced and used framework for handling semantic data in Java. It can handle OWL ontologies, which makes it powerful in itself. Jena also allows for querying the data and has built in support for an end-point available over HTTP, in addition to a TDB triple store. In short, it has a lot of features(Apache Jena, 2015).

Rdflib is a Python library for working with RDF. It has no native support for OWL, but packages such as the OWL-RL library can expand rdflib to include OWL-RL support. This package also includes RDFClosure, which can expand the graph using transitivity etc. In my experiences the closure is too time consuming compared to alternatives such as Jena, and as such I would only recommend the usage of rdflib if you are using Python (Hees, 2015). The library is more lightweight and easier to use than Jena, but if you know Java, I would argue that Jena is the better option.

For Python use there are other libraries such as RDFAlchemy(http://www.openvest.com/trac/wiki/RDFAlchemy), Fuxi(http://code.google.com/p/fuxi/), which uses rdflib, ORDF(http://ordf.org), Django-RDF(http://code.google.com/p/django-rdf/), Djubby(http://code.google.com/p/djubby/), Redland(http://librdf.org/) and SuRF(http://packages.Python.org/SuRF/), but these have not been used during the development of this paper.

5.1.3.2 Weaknesses

Almost every dataset has weaknesses. A weakness in this context is defined as a variable or field that create problems during the lifting. In many cases these problems occur when linking to other datasets, which especially happens when the data have problems such as: consistent spelling mistakes, are old, encoding differences, limitations on data gatherings(e.g. amount of calls allowed in a time period to an API), oversimplification of data, and other types of problems you may encounter. Some of these weaknesses are hard to spot before the lifting process and are sometimes only spotted in the resulting graph. For a streamlined process it is important to identify as many weaknesses as possible before starting the coding.

When lifting data, the data are likely to already have a defined use. This can range from informing customers, creating statistics or other goals. Data that has a goal of informing customers will in many cases be simplified, making it easier for the public to understand the data without prior knowledge of the field. This can create weaknesses during the lifting since the data might be converted into less specific but more humanly understandable versions. As an example, the district from the Vinmonopol case can show some of the problems introduced with this simplification. In many of their Cognac products the district listed is “Cognac traditional”. This district is not an official region in France nor a specific Cognac district(Fins bois, etc) but rather a generic term meant to encapsulate all the Cognac districts. This means that when the district is attempted to be linked to GeoNames, the district won’t exist in the GeoNames dataset and therefore be much harder to link. Even though the process of linking data should always use rdf:type and other categorisations to identify its target, it is ever more important in these instances.

If the data being lifted is old, the problems this creates has to be identified. If the data for instance has at several points in time changed its structure, the lifter will be required to solve this, sometimes using several version. An example is the donations to parties on case 3. Here, the organizations might have changed name or address and the number of organizations that can be identified in BRREG has to be found. If there exists old versions of the other data as well, this might be used to solve these issues.

Since weaknesses are often discovered late in the development process, they are one of the main reasons the process performs better when used iteratively.

Outside issues can also be counted as weaknesses of sorts. Web services can introduce new problems to the lifting process as well. The League of Legends case has a good example of this as the maximum number of allowed API calls are on average 1 per second(the allowed amount increases if there has been a period without calls). This puts a cap on the amount of data that can be gathered and/or matched. This can therefore create problems when handling big datasets. Another example is the GeoNames SPARQL endpoint, which I expected to use in the Vinmonopol case. The endpoint however only gave access to a certain subset of the GeoNames data. The data I wanted to match with the endpoint was not the same and therefore it took a lot more math in the SPARQL queries to find the information that I wanted.

5.1.3.3 Data gathering

Data gathering is the process of finding, gaining access to, and sorting the data which are to be used in the lifting process. The datasets are in most cases found before this stage in the process; however, the lifted data might be required to be linked to other new data. If the data originate from APIs, this requires some way of gaining the right sets of data before the semantics can be applied.

5.1.3.3.1 Scraping data

In many cases the data being lifted will be in a structured format, but in some cases the job gets more dirty. For instance in the public dataset case the lifting process had to scrape data in several datasets. The lifter therefore needs to implement a code which reads the unstructured data, in these cases HTML. These tend to be more specific than the lifters for more structured data. The archive on partifinansiering.no's

HTML files has for instance changed its structure over the years, which makes it even more specific as you might have to create two or more different codes depending on how different the structure is.

There are tools available that make this work less complicated. While scraping data from partifinansiering.no in case 3, I used a tool called Beatiful Soup. This tool allows the code to pass it HTML code. Then it will create a structured hierarchy of the HTML and allow the code to search for specific elements and iterate through the different levels of the HTML.

5.1.3.3.2 Structured data

In many cases the data being lifted will be in a structured form. More often than not, data will be available as either JSON or CSV among others. In this context when talking about structured data I mean data that use a format whose main purpose is to store the data, and only the data, in a structured way. HTML will therefore not count as structured in this context as it not only contains the data, but also elements that are there to present the data.

For these sets of data the work consists mostly of iterating through the data and creating rules for converting to semantic formats. These datasets are often straightforward in terms of the coding. Difficulties are mostly met when the data calls for connections between lines or fields as there needs to be some way of connecting even after the previous data has been interpreted. Difficulties might be an exaggeration as it's not particularly hard, but rather just needs extra collections to remember which resources in the graph the data should be connected to. This is mostly necessary when the model calls for anonymous nodes.

For CSV files, tools are unnecessary considering how simple the structure is. Other formats such as JSON or Excel can be handled by using tools such as:

- Excel: xlrd(Python), Apache POI(Java)
- JSON(java): gson, Jackson

Some formats, especially Excel, can have inconsistencies with regard to data formats. Excel files have in case 3 shown that it is prone to misrepresent the data. Some of the files represented numbers as string, integers as floats etc. These mistakes are usually not hard to correct, but in large datasets they are often not found before the algorithm is run.

When lifting XML data, the data are as in JSON data structured in a way that allows you to directly link elements to a semantic terms. As Ferdinand *et al.*(2004) proposes, this can be handled by mapping the different structures of the XML schema to OWL and then using the XML document to create the triples in the RDF graph.

It could be argued that Excel is not necessarily a structured format as defined above as it also contain presentation information(color, borders, scripts etc.). But I have decided to include it as a structured method as the tools that are used make it easy to ignore these layers.

5.1.3.3.3 Fetching data from APIs

When using data gathered from APIs, new situations appear. Many APIs have limits on the number of requests in a given time frame. In the event where a lot of data has to be fetched, some APIs might offer ways to download data in large bulks while others requires the data to be downloaded individually(for instance the games in Case 2). These data are always structured, oftentimes using JSON format, but they might also allow the developer to choose from a set of different data structures.

With APIs, the flow of the lifting process is much more streamlined, as when new data are fetched, it does not require the rest of the data to be gathered as well. If the data were available as bulk data, whenever new data was to be added, the old data would need to be both downloaded and sifted through.

Problems with APIs come when the API is updated or with downtimes. How much an API is down varies from case to case, but it can be a problem if the data are used in consumer related products, since the product might rely on having up-to-date data. Changes to the API can also happen and in certain cases this can create a need for modifications of the process or an entirely new lifting process.

5.1.3.4 Triplification

When extracting the data from its source, the best course of action is not always to directly apply their semantic meanings. In some cases it might be preferable to have a stage called "triplification" which is basically transferring the data into a semantic structure, bar any properties and classes. This might be preferable since it can make the lifting process more streamlined. Rather than meddling with the specificity of the semantics, the data can simply be represented in triples using, for example, the column names in an Excel document. When all the data are extracted, the semantics of the model is applied to the triples. Having a triplification step in the process can also benefit a team which is working in parallel, as the modelling team does not have to be finished before the extraction of data can begin. This is even more relevant in a process which relies on the scraping of data from a structure that is not well defined, as this takes significantly more time than a well defined structure.

5.1.3.5 Linking to other datasets

When linking, one usually has a native dataset and a foreign dataset. The native dataset is the one being lifted, while the foreign one is an outside dataset that you want to link some data to. The data being linked from the native dataset will always be the correct data, the data being linked to in the foreign dataset, however, is not always as easy to identify as the same object. There can be many entities called "Champagne"; beverages, regions and cities can all be found when searching for it. Some form of equivalence test must therefore be present in the linking process to ensure the right entity is linked to. In the instance of case 3, this is done by having an completely equal name(as that is enough considering no two organizations can have the same name) or a very low difference in names in combination with an equal address. In case 1, the regions are matched using an equal name and then, using a priority based list of region types, selects the entity which matches the name and highest priority region type.

Using the linking points identified in the model construction stage, we first attempt to find the fields and relationships which best can represent a unique entity or type of entity. To do this, a sample entity of each of the different types of linking points from the native dataset has to be mapped with each of its counterparts in the foreign dataset. The defining fields and relationships must then be identified in the foreign dataset. This will in most cases be either classes, the more widely used the better(see YAGO etc.). It might also be specific relationships, such as in case 3 where addresses became determinant for the linking.

An effort has to be made here to make sure the objects you want to link to are the only ones that can be identified with these variables. In case 3 the laws regarding both naming of organizations and for the statements that report donations to the government had to be read to be certain of how unique each name could be considered. The small differences matter here, for instance, the difference between two organizations names are only required to be 1 letter different. Either it being in the name or in the declaration of the type of firm(AS vs ANS) does not matter.

When using data that is not structured in the sense where each entity is in its own field/column, one might have to use some Named Entity Recognition technology to find each object. An example here is case 3 if the donations, contributors and addresses were all contained in the same table element(they are not, but if they were the different entities would have to be identified).

Sometimes the solution is linking data manually. This can be required where the entities are represented with synonyms or other names that makes matching with other datasets hard. In the event of this the solution I implemented is to create a list of known mistakes which are mapped to either their resource in the foreign dataset or to their real name. This can be very time consuming depending on the amount of data. Case 3 required some manual matching between pf.no and brreg.no. This was because some of the donations are only identified by abbreviations that were not listed in the BRREG dataset. Some of these were semi-manual in the sense that I created a script that matches the items based on known traits of these organizations and then links them to a predefined organization.

When linking to outside datasets, one has to take into account the technologies that are used by the respective datasets. Problems emerge with the interoperability of the technologies when combining data from two different technologies. García-Castro *et al*. (2010) talks about the transitions between different semantic technologies, in this case tools. They show how the different technologies affect each other in the different tools they use. Some transitions are problematic and it is therefore important to be aware of this problem when connecting data. As the amount and complexity of data you include, the more types of interchangeability can be found(Garcia-Castro *et al.,* 2010).

5.1.3.6 Continuous updates

Many datasets will be most relevant when there are new data added continuously. New donations, updated data on organizations and new matches all add more value to the data. Although not all datasets need to be updated continuously to be relevant, those that do requires the lifting process to be built around this. In Case 2: League of Legends I talked some about how that case solved this issue. As far as I can see these issues can be divided further into three cases:

- When new data are added while being cut off from the previous data, as in the League of Legends case.
- If new data are added without it being differentiated from the old one
- If the old data are changed either alone or in addition to either of the two other cases.

This problems in the public dataset case can be solved the same way the League of Legends case does it, by dividing the process thoroughly and then fetching it individually, in both the other two situations, the whole dataset has to be lifted again. In any of these cases there has to be a way to identify when new data are added/changed.

The third type is present in Case 3: Public datasets. The data available on partifinansiering.no are updated through the year. If the data were part of a program which was used and meant to be updated as new data were introduced, it would need to apply a combination of the ways mentioned above. The donations for previous years could be ignored, as it does not change, but it would have to look for new data in the current year and lift the whole year if there are new data. This means that portions of the data have to be lifted again, but a large part can be left open.

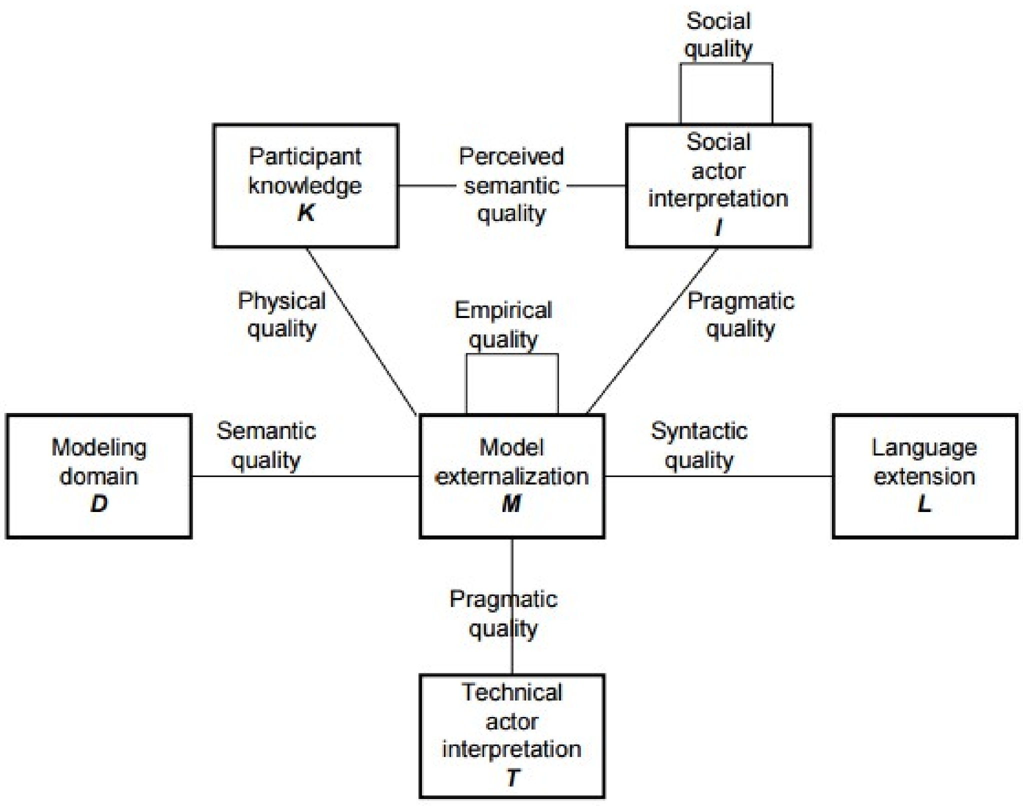
5.1.3.7 Lifting to OWL

When building an OWL model one has to make sure that whichever library that is used to create the semantic model can handle OWL. In my experience, using reasoners can be a time consuming matter. In case 3, where I used Python, an independent reasoner built upon the rdflib library that was used was implemented. However, the reasoner used extremely long time to applying closure to the entirety of the model. Although there was a lot of data in this case, the reasoner was allowed to run for more than 8 hours, still producing no results. In the other cases where I used Java and Jena, the reasoners included in Jena proved to be much more reliable and were later implemented in the public dataset case. This might be because of the different types of reasoners provided in Jena. One could choose between OWL-FULL reasoning, a reasoner that only handled the transitivity of rdfs:subclass and other smaller OWL and RDFS reasoners. One does have to be wary of the different types of reasoners though, as they are not all for the same purpose. Doing research on which fits your needs are mandatory.

5.1.4 Evaluation

The evaluation of the process consists of determining whether or not the lifted data answers the competency questions produced in the Goal stage. All the stakeholders have to agree on the competency questions being adequately answered before the process can be considered done.
It is important to be able to identify when the model can be improved as a result of a new iteration. If the stakeholders do not agree, some of the competency questions are unanswered or new ones emerge, a new iteration of the process should be started. A new iteration might not require a full run of the goal stage, depending on what improvements are envisioned.

Several different frameworks are available to evaluate the quality of the model. Even though these evaluations should be up to the persons conducting the lifting, I will quickly highlight one as an example: the SEQUAL framework that was proposed by Krogstie *et al.* (1995) and later improved upon to handle active models in Jørgensen *et al.* (2001). This framework can be explained as shown in *graph 5*.



Graph 5: The framework as in Jørgensen et al. (2001).

This framework shows different aspects that can affect a model and its quality. The lines connecting aspects represent quality types. These quality types are explained in more detail below. The italic lines are my interpretation of how each different quality can be used in the evaluation of a lifting process.

- Physical quality
  - Externalisation: the knowledge of each participant group is to be included in the model. This includes users as well as developers.
  - Internalisability: the model has to be available so that each of the participants can understand it.

  - *This quality can be used to interpret whether or not the model shows every aspect of the data correctly.*
- Empirical quality shows to the mistakes that are made when the model is interpreted by people or tools.
  - *If the model has poor choices in predicate names, classes or hierarchies, this can be reflected in this quality.*
- Syntactic quality: corresponds to the use of the language of the model.
  - *In a lifting process it can for instance be relevant to the correct usage of OWL. If this is of low quality then there will emerge problems in many aspects of the process.*
- Semantic quality shows how well the model is representing the domain. It can be divided into two sub qualities:
  - Validity, which judges if the data included in the model are correct and related to the goals.
  - Completeness, that all the data that are relevant are included.
  - *This is an important aspect of the evaluation. If the semantic quality is low, it is likely that the goals of the process will not met.*
- "Perceived semantic quality is the match between the participants' interpretation of a model and his or her current explicit knowledge. As the domain D cannot be completely known, semantic quality can only be tested indirectly through the participants' knowledge." (Jørgensen, Krogstie, 2001:5)
  - *Considering that the domain of the lifting process will in most cases be a predefined set of data, unless the outside datasets are seen as the "unknown" part of the domain, this will not be relevant without additions to the method.*
- Pragmatic quality shows how the end-users interpret the model.
- "Social quality: The goal defined for social quality is agreement among different participants' interpretations. Social quality affects communication among participants about the contents of the model." (Jørgensen, Krogstie, 2001:5)

(Jørgensen, Krogstie, 2001:4-5)

Using this framework and its quality types to show how well a lifting process has done in different aspects can be valuable and contribute to the decision on whether or not the process has reached its goals while being of adequate quality. If not, one can perform another iteration of the process to improve the quality where needed. It should, however, be said that not all lifting processes require all the above mentioned qualities, as not all would meet the requirements for being considered "active models". Which framework is used should be a decided by the participants of the process, with regards to the end-use of the product and whether or not it can be considered “active”.

The framework was revised by Krogstie *et al.* (2006) where they proposed an improved version.

5.2 Discussion of Case 1: Vinmonopolet

This case went as well as can be expected during the lifting. The requirements were met, however, the filter and search improvement goal cannot be seen as completely met. The reason this can not be treated as met is because it is required to change the source code of the search itself, as such a method has not been developed in this work. It is especially hard to determine whether or not the filter functionality could be improved, considering the functions fields are based on product information, which have many recurring values throughout the dataset. Some fields, such as alcohol percentage, could be included with above/below(>40%, <60%) options, but whether or not this is an improvement on the previous database is not certain. Considering that the new database allows us to at least retain the same functionalities, it could be argued that the requirements were met, but I chose to leave this last goal open.

The GeoNames dataset was used because of its SPARQL endpoint and functionalities therein (allowed for searching for nearby places). Later in the development, when the SPARQL queries were to be tested, I discovered that the endpoint did not use their entire dataset, and the data I needed was not included. This meant that the endpoint could not be used to find nearby stores, and instead had to be done manually. The algorithm for this is as follows:

“

$a = \sin^2(\Delta\varphi/2) + \cos \varphi 1 \cdot \cos \varphi 2 \cdot \sin^2(\Delta\lambda/2)$

$c = 2 \cdot \mathrm{atan2}(\sqrt{a}, \sqrt{(1-a)})$

$d = R \cdot c$

Where φ is latitude, λ is longitude, R is earth’s radius(mean radius = 6,371km)
note that angles need to be in radians to pass to trig functions!”

(Movable-type, 2014)

This means that the distance between two points can not be properly determined in a SPARQL query given SPARQL’s mathematical functions, it can only be presented as the distance without accounting for things such as the curvature of the earth. To transform it to either kilometre or miles the calculations has to be done post-query. Although this is not a major problem, I was hoping this would be solvable in the GeoNames endpoint as part of a query.

The linking to GeoNames was also the most difficult part of lifting this case. Identifying the regions in the GeoNames data was done using a prioritized list of entity types. If there were several resources in GeoNames with the exact same name in the same country, the resource with the highest priority would be picked. Basically, the larger resource would be prioritized and regions were prioritized over populated areas. Although this worked for the purposes of this case, I am quite sure there are better ways to solve this, and if done again this would probably be one of the things that should be changed as the prioritized list does not necessarily create optimal solutions for every region.

In addition to the districts being represented wrongly, the countries have some problems. They, as mentioned in the work section, use their Norwegian names in addition to abbreviations such as "Bosnia-Herceg.", "Brit. Jomfruøy." etc. This resulted in the use of some translation before linking to DBpedia by using the dataset from erikbolstad.no (Oversikt over land, 2010). This could, however, also have been solved by searching on GeoNames, as the samples I have looked at after the development seem to contain names for the countries in most languages. Additionally, this could be done using the information provided from the companies creating them, which I assume is in English or its native language, if the lifting was handled internally.

Contribution to the proposed method

The case contributed especially to the idea of interpreting the data and what it tries to convey. In the other cases the data were straightforward and handpicked for a specific purpose, but in this case some of the data were classes and categorizations that were not straightforward. The case also required more difficult decisions regarding existing vocabularies. In case 2 and 3 there were no vocabularies that could explain the core concepts of the data, while this case represents the same concepts that WineOntology does. The thoughts behind this decision can prove to be important if outside people are to use the datasets.

5.3 Discussion of Case 2: League of Legends

The CQs of case 2 were all solved by the end of the lifting. Although this might have been the lifting process that went smoothest, a lot can be learned from it.

The lifting of this dataset was hindered somewhat by the rate-limit of the API. To receive an API-key, the developer has to have a registered account in the game. This does not cost anything, and as such does not put any restrictions on the process. However, the initial API-key is only so-called "developer key";

meaning that the key has the relatively low rate-limit of 10 requests per 10 seconds. To get a higher rate-limit, one has to get the application “approved”, meaning that it has to be available on the net as well as finished. Considering that the goals of this application was merely to lift it to semantic formats and use these to show statistics, not present it in any way on the Internet, getting a higher rate-limit was not possible. However, the program does not suffer a lot from this, as the lifting itself would not be different. The main difference would be the number of games that could be included, and more games would means we have a better chance at achieving statistical significance.

In the description of the case I wrote that the reason I started the project was because of the poor state of the statistics offered by the game developer. Even though they have an API that allows developers to fetch data, they have not given the statistical side of their game any updates in a long time(it is to be said that there is a new client in the works and it can be speculated that it will include better statistics than the previous version) with the exception of timeline data, though this timeline data has not been included in the data I lifted. Having these data in semantic formats allows us to create ways to visualize it using graphs as shown in the attachments. Another goal was to allow the users to create their own graphs, rather than just a set of predefined ones. The use of graphs which take SPARQL queries and an endpoint as input to output a graph allows us to create a wide set of options that users take advantage of. The code that allows this is rather simple. It provides a graphical representation of a set of options, at this time player name, champions on each side, game mode, queue type etc. It then builds a SPARQL query with the users’ data and displays this in a graph. This was though using the old model, which did not include more than a handful of statistical literals, and therefore the graph can allow for even more modification by the users, such as higher and lower values for game time, certain stats for a champion etc. with the new model. The data do not have to use semantic technologies to accomplish this, but it is none the less made easier by the application.

The second use I see from lifting the data are closely related to the first one in which it is both statistical and not specific to semantic technologies only. With the data in semantic formats we can use machine learning to create clusters of champions and even players to see which ones work well together. This can be used by both players to explore team compositions, and by developers to help balancing the game so that no single champion or set of champions have an advantage without being able to be countered, and thereby creating an atmosphere in which the game can flourish.

From the data owner's perspective

The case could especially have been improved in regards to both the time consumed and the quality of the content had the API been in a better state. It is understood that it is an ongoing process, and even as this thesis is written, new versions and structures of the API is being deployed. Not all of the callable objects are intuitive and some data seem to be redundant. If the end result requires significant amounts of games, the fact that each game has to be queried individually, in addition to only being able to see the 10 last games of each player, creates boundaries when combined with the APIs cap on calls per time period.

Contribution to the proposed method

This case was mostly different from the others because of its use of JSON data and being completely reliant on an API. This is reflected in its contributions to the method, as its lessons can be found mostly in the subsections on *structured data* and *fetching data from APIs* in the *data gathering* section. It also contributes in the section on *identifying main concepts*, because of the modelling of a player's participation in a match. Lastly, this case is the main contributors contributes to the section on *continuous updates* and is the main contributor to this section. The other cases' contribution to this section is merely afterthoughts and not based on experience.

5.4 Discussion of Case 3: Public datasets

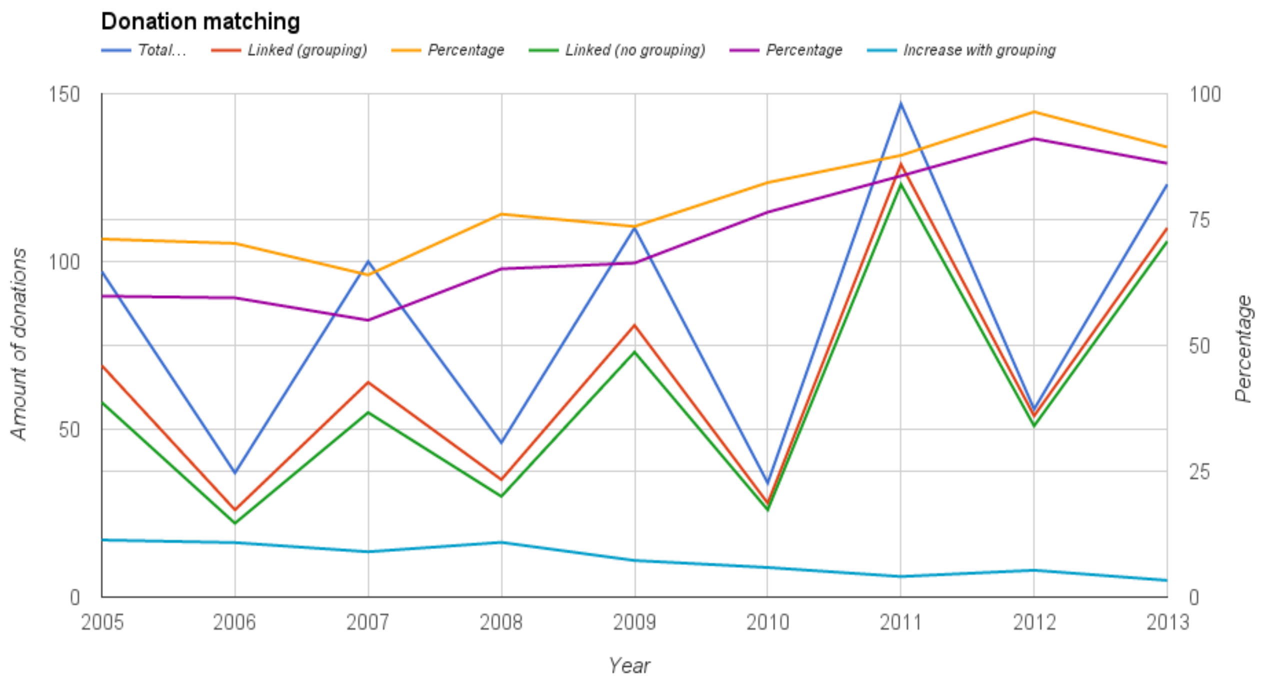
The cases were all lifted successfully, although with varying degrees of work required. This case was significantly more work heavy than the other two, mostly because of the raw state of the data, but also because there were more datasets included in it.

These data were as mentioned earlier lifted using Python. However, this proved to slightly hinder the use of OWL. The reasoners I found and tried in this case were very time consuming, and in the end the Jena reasoners had to be used post development. This can lead us to believe that Python might not be the best environment to lift data to OWL in the very least. If better reasoners that I have overlooked are available, it can be as viable as Java.

Linking BRREG and pf.no data

When the data are in a poor state, a robust linking algorithm should be a priority of the lifting process. This is prevalent in the linking of donation data and BRREG data in case 3.

A prerequisite to linking contributors in donations to BRREG was that their names were unique. This was especially important because of how old the data were, resulting in potentially both outdated names and addresses. The name of a registered organization is unique(Altinn, 2014), however, to differentiate two organizations only 1 letter needs to be different. Therefore the process of linking the data required the names to be exactly alike in the donation data and BRREG. As shown in the lifting section of the Brønnøysund registries, any donations where the contributor is not identifiable with name, they are grouped together with other contributors whose names are equal enough(>0.7 on the levenshtein ratio) *and* have an equal address. Before this grouping algorithm there were 750 donations(not counting donations from private individuals) and after there were 340, which shows how many donations were made by similar organizations. These numbers are though from grouping when none have been matched with the Brønnøysund registries. Considering that there might be faults in the algorithm, the script instead ran this algorithm after the linking to BRREG was finished. The algorithm is the same, but only the donations where the contributors have not been found on BRREG will be eligible to be grouped with others. This means that there will be more request to the API but less(if any) mistakes made from the grouping algorithm. This is because the organizations that have data from BRREG will be transformed into the same semantic resources during the transformation process. When changed, this resulted in only 67 groupings, but it still significantly increased the percentage of donations with contributors identified on in the registry.



Graph 6: Shows the relationship between years, donations and percentage of orgs matched from PF to BRREG.

The “Total donations” displays the total number of donations from a given year, not counting donations from individual people. The red line represents the number of donations which were donated by an organization which has been identified on BRREG using the grouping algorithm. The green line shows the same as the red line only without the grouping algorithm. Blue is the total number of donations in a given year, yellow is the percentage of linked with grouping while the purple one is the percentage without grouping. The light blue line shows the percentage increase of organizations link to BRREG when using grouping over non-grouping(essentially the gap between yellow and purple).

From the graph it is evident that the election years contain much more donations than the other years. Year 2007 and 2011 are both on the same level as the election years. For 2011 this is in large parts because of the Utøya incident, which caused a lot of donations to Arbeiderpartiets(Labour party) youth organization. Many of these organizations are either fundraisers or not from Norway. This makes these donations not eligible to be linked to BRREG and reduces the linking percentage. When starting the project I mistakenly assumed that the donations would be from Norwegian organizations. This proves that one should attempt to sift through the data before lifting it unless the size of the data makes that difficult. This part especially has contributed to the process concerning the need to understand and go through the data.

The graph also shows that any donation from the earlier years has a significantly lower chance of being linked to an organization in the Brønnøysund registries. The chance of being matched increases gradually with the years. This is probably due to the state of the data. Since the data are old the organizations might have undergone name or address changes, liquidations etc. What all this can show, is that linking data to other sources is heavily influenced by the state of the data. When lifting older datasets, linking data not only to the foreign datasets, but to itself, can provide a significant boost to the state of the data. In cases where each individual case is highly relevant, such as projects where the end goal revolves around statistics this can be a detrimental factor for the end result.

As this is all done by an outsider, one can argue that especially the PF and BRREG datasets could benefit a lot from an internal lifting process in contrast to the outside process shown here. The data could prove to have a significantly higher percentage of contributors linked to Brreg if the owners of these two datasets cooperated in creating an ontology. The taxonomies would probably reach the closest solution to real life if done by the ones owning and gathering the data. Certain improvements on this could also be made in regards to the collection of data, as mentioned earlier. This notion can be carried over to the other cases as well. If the data-owners themselves transformed the data, it could also be assumed that this would be a continuous effort and as such the inconsistencies of the data would not be as prevalent(at least if there were organizational numbers involved).

Election data

The dataset concerning elections did not prove to be too valuable to the research. It did not appear to give any new insights on the process of lifting data, and as such is only discuss briefly. As it used CSV data it was a relatively simple process. The only work except of the direct transformation to RDF, was identifying the individual files and the manual matching of party names to the party identificators from PF.

Technical improvements

The case would benefit from a review after all the cases were done as there are some weaknesses in the case. The first thing that could have been improved is to include the bulk BRREG dataset instead of fetching from the API and website. This would open some new possibilities, for instance, there could be made a custom function for looking up organizations to link to contributors. Considering the problems with the API not finding similar results as the websearch, this in combination with easier access to addresses could allow for a more robust linking process. The reason the bulk was not used for now is, as stated earlier, due to the fact that the data are always up to date on the website, and that there are search functions available. I have though realized what should have been realized earlier; the data will be outdated just as fast as the one from the bulk data, as the data we lift are from the newest versions anyway and are not updated in our data because of their origins. New liquidations, bankruptcies or changes to the names of addresses would appear in neither one without steadily updating the data and this can be done with both ways.

This case has almost unlimited potential to include more data. From both SSB and NSD(though often the same data), bdb.no and many more have data that can be relevant to include in the case. This would allow for the data to show how big companies are before they donate, how their economy is affected by these donations(if the donation is to the party that wins the election, how is the company's economy after?). If the official sources were to give out these data in semantic versions, there might be many applications for such data when combined with different ones. If the BRREG data were semantic, the donation data could instead of being identified with name and addresses be by organization number, automatically extracting the information from BRREG. Since the contributors only have to give the name of the organizations, it sometimes is not clear which branch of an organization is the contributor. If the donation data had a more robust system for the statements in this regard, it could make the data more transparent and ease the work for those using it without creating (noticeably) more work when reporting it.

One of the first goals of the case was to graphically represent the data. This was originally planned to be achieved by using the SGVizler library. SGVizler lets a website show charts using Google Charts simply by including a JavaScript file and putting an attribute with the SPARQL query and SPARQL endpoint in the div where the charts is to be shown. This allows for information to be implemented in the website easily, as no post-processing is required. The problem with this is when columns are generated in the query the charts don't seem to understand how to represent them. An example of this is the query which found the 3 industry codes with the most workers in each municipal. This meant that each number for the industry codes are in one cell of the resulting table while the industry code itself is in another. This would require a custom graph interpreter or post processing which SGVizler seemed unable to handle.

The partifinansierings data consists of more than 51000 HTML pages. This is a very large number considering the structure of the HTML varies in them. The way the lifting works in this case was by having a single script for all the different structures. This can appear messy if someone who have not written the code is to modify it(without extensive commenting at least). This can be improved by using machine learning to cluster the different structures in the files and using these clusters to decide which script is used to lift any given page. Considering this is a supervised learning method, the coder would still need to have a overview of the different types of structures out there, but this is also something that can be solved using machine learning.

Contribution to the proposed method

This case provided many ideas for the proposed process. Especially the parts in the process where one is to explore and identify the weaknesses of the data are inspired by this case. The case consisted of very unstructured data(in some datasets) with a lot of mistakes. The linking sections of the process also draws upon this case, as it contains many datasets that are all linked together, from several data sources. This is also a part of the decision to recommend iterating through the process as iterations improved this case a lot.

5.5 SPARQL Queries and optimization

It has become evident from the work in this thesis that SPARQL queries are not straightforward, especially when taking efficiency into consideration. Both the League of Legends graph queries and the queries about which industry codes have most employees per municipal proved to be more time consuming than anticipated. The field of query optimization has been extensively researched through the years(Liu, C, *et al*. 2010) and can be considered an important part of making semantic web more mainstream. From the work of Liu, C *et al*.(2010) and Stocker, M *et al*.(2008) it can be understood that the efficiency of a query is heavily reliant on the number of joins a query creates. To solve this, histograms can be used to find the optimal solution, in this regard the solution with least joins. In the queries that find the amount of employees in the top 3 industry codes of municipals(case 3), the amount of joins in combination with the large amount of data meant that the query became time consuming.

If this was a commercial product meant for consumers, the time taken to perform certain queries could reflect poorly on the perceived service quality of the product (Yang *et al*., 2004). Therefore it might be worth it to create semantic models which require as few number of joins as possible to decrease the workload for the query processors. I will not spend any time to research the best possible ways to do this as this thesis is mainly about lifting data, not optimization, but it is relevant for the method considering if done, it has to be done during the lifting process. For the partifinansiering case the solution might be as simple as calculating the top codes during the lifting process and introducing a predicate for them. This is though only possible when we know the queries that will be used on the data and is really a band aid solution.

5.6 Validity

When Bryman(2012) talks about the criticisms of qualitative research, one of the criticisms are that results are very influenced by the points of interest of the researcher. One researcher can focus on different points in the data than others and as such the outcome is too reliant on the person conducting the research. As case studies are mostly used in social research and not computer science, I will argue that that criticism is somewhat weakened by the fact that the subject in this instance is static. The data in the cases would be the same for everyone in the sense that the data's structure is from a finite set of variants, most often XML, JSON, Excel, CSV etc. This makes for a certain limitation on the variants of end-product of this research.

I would argue that the point of interest of the researcher is more likely to have an effect on the models created in the cases. Considering that the end-product of the research is the proposed method of lifting, this should cause too much variance.

Research where people are the subject contains a lot more variables and diversity than research conducted on lifting of datasets. The data in the datasets have infinitely possible combinations, but the structure of them are not. Bryman also mentions that one can not assume that research conducted on a limited set of a population represents the entire population. He writes "A case study is not a sample of one drawn from a known population.”(Bryman, 2012: 406). The three cases included in this thesis are so few that it cannot be assumed that it represents all of the datasets that can be lifted. The notion that qualitative research, instead of attempting to generalize the entire population, generalizes the theory of the field(Bryman, 2012:406) is what makes this research viable.

The cases included in this study varies enough in content to give a general overview of the requirements for lifting data successfully. I would argue that the research is relatively dependable in the near future. If standards were to change, or some new data structure was to become frequent, this would not necessarily create changes to the method, but rather require additions to the method. The one big thing that could change is if the world implemented semantic technologies more frequently. In such case the method might change to spend more time evaluating existing vocabularies and less on creating your own in addition to having more potential datasets to combine the data with.

5.7 Evaluation of the research method

The research method worked as expected. The cases provided useful experience to propose a method. However, several improvements can be made if one are to perform this research again or continue it.

5.7.1 Stakeholders

Considering the intended end-result of Case 3: Public datasets the researcher could have had more communication with election researchers or journalists. These people could possibly have ideas on which data that were required to create a qualitatively better environment than what the case provides. This could even after the directional change of the research have proved useful as the process could have included different datasets or the graphs could have been modelled more towards visual representations(such that it is compatible with for instance SGVizler). The competency questions for Case 3: Public datasets could have been drastically different if outside people were consulted. This could also be improved in the other cases, as one major weakness of all of the cases are the lack of any stakeholders beside myself. This can make the development of the cases lack key CQs, although it would probably only cause differences in the model or cause the data to be linked to more/different datasets, the evaluation of the cases might be drastically different. It should though be noted that the Vinmonopolet case is rather straightforward in terms of already existing data and the League of Legends case is very specific in its goals.

If the research included stakeholders for its cases, it might be that method had been different in certain areas. Lifting data will in most cases be about fulfilling some stakeholders visions and requirements and the method as proposed does not handle any issues concerning communication or conflicts in regards to goals.

5.7.2 Variation

Qualitative research is often differentiated from quantitative with the notion that it focuses on depth instead of breadth.(Bryman, 2012:392) However, it can be argued that the perfect research would include a healthy dose of each. Datasets can be categorized based on a set of characteristics: data-structures, availability, the information it relays, how often it is updated, and how long it stays relevant to name a few. The cases in this research have some variation of these characteristics, but considering that there are only 3 cases(although more than 3 datasets), there is no doubt that the research could benefit from more cases and variation within these. Berg(2001) argues that collective case studies should involve several cases to investigate broader parts of the given field. Even though it can be argued that these cases are enough to propose a lifting process, some more characteristics could be helpful. Datasets that require Natural Language Processing(NLP) to extract the data would for instance give a very different and more complex lifting process compared to the cases in this research. Other characteristics, such as datasets using XML can, even though they are not as complex as NLP, also possibly help with the broadness of the method. The other characteristics listed are somewhat covered, but more cases with each one can still benefit the research by making it more diverse.

This chapter has discussed what was well done and what was problematic with the work, as well as presented a method for lifting data. I have discussed improvements that can be made with regards to SPARQL optimization in the model, which can certainly be a challenge in the lifting process. How well each case has performed is discussed, and one could from this argue that cases 1 and 2 were easier and better lifted than case 3, mostly because of the difficulties found in the raw data in case 3.

6 Conclusion

This research has explored the world of data-lifting, specifically to semantic technologies. I have shown some previous works that are relevant to this field, lifted a set of cases to semantic formats using a variety of technologies and shown what problems arise when doing so. The work has investigated different data-structures, availability, complexity and structure. The thesis contains a method for lifting data that builds upon the experiences from the cases. I have discussed what can be done better, which types of data structures are left out of the cases and how models can be optimized. The steps laid forward in the method are inspired by the problems and solutions in the cases. Even though there were some problems(mainly mathematical), I would argue that the cases were adequately lifted to meet the demands set beforehand. As the method envelops the problems and solutions of the lifting process, this can be interpreted as an answer to both the research questions.

The first research question reads: *"Which problems arise when lifting data to semantic technologies?"*. This has been answered in the *work* and *discussion* chapters, where problems like the data being old, inconsistencies, misspellings making trouble when linking to other datasets, optimization, using math in the SPARQL queries, the data being simplified, and more are introduced. The problems presented here are of course not the only ones that can emerge during lifting. It does, however, show a variety of types of problems and some solutions to these.

The second research question; "*Can a method that guides the lifting process be developed from these studies?"* is attempted answered by the proposed lifting method. Again, it is not possible to say for certain, but the method proposed would work on any of the cases I lifted, which has a broad specter of variety.

Considering that the existing work in this field is rather slim, and the previous work I have shown to are at a higher level of ontology defining, the method presented here can be considered an addition to the field of semantic technologies. It can help people and organizations in their efforts to lift data to semantic technologies, either to create new ways of using data, or by increasing its value to outside users. Either way, it can help make the change to semantic technologies easier for data-owners.

Future work

The method presented here has not been tested after it was proposed. It is based on the experiences I had when lifting the cases and draw upon what I thought worked during my work of lifting the cases as well as what I felt could have been done better. The method can benefit from a third party using it to lift some dataset. Additionally the work done here can as discussed in earlier sections be improved by lifting more cases using different data-structures or ways to extract the data from its source, such as NLP. The work on the League of Legends case will continue after the research is finished, attempting to incorporate larger parts of the model into the available customization by the user.

7 References

1. Allemang, D., & Hendler, J. (2011). Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL. Morgan Kaufmann Publishers Inc.
2. Altinn. (2014). Hvordan beskytter jeg navnet? Retrieved May 05, 2015, from https://www.altinn.no/no/Starte-og-drive-bedrift/Forberede/Krav-til-navnet/Hvordan-beskytter-jeg-foretaksnavnet/
3. Apache Commons - Apache Commons. (2015). Retrieved February 04, 2015, from https://commons.apache.org/
4. Apache Jena. (2015). Retrieved April 30, 2015, from https://jena.apache.org/
5. Berg, B. L., & Lune, H. (2001). Qualitative Research Methods for the Social Sciences (8th ed.).
6. Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web. Scientific American. doi:10.1038/scientificamerican0501-34
7. Bharati, P., & Chaudhury, A. (2004). An empirical investigation of decision-making satisfaction in web-based decision support systems. Decision Support Systems, 37(2), 187–197. doi:10.1016/S0167-9236(03)00006-X
8. Bizer, C., Tom, H., & Berners-Lee, T. (2009). Linked Data - The Story So Far.
9. Bring. (n.d.). Postnummerregister med endringer. Retrieved May 31, 2015, from http://www.bring.no/hele-bring/produkter-og-tjenester/brev-og-postreklame/andre-tjenester/postnummertabeller
10. Bryman, A. (2012). Social Research Methods.
11. Brønnøysundregisteret. (n.d.). Retrieved May 20, 2015, from http://www.brreg.no
12. Cardoso, J. (2009, September). The Semantic Web Vision: Where are We? IEEE Intelligent Systems, 22–26.
13. Davis, M., Allemang, D., & Coyne, R. (2004). Evaluation and Market Report.
14. Ferdinand, M., Zirpins, C., & Trastour, D. (2004). Lifting XML Schema to OWL. In N. Koch, P. Fraternali, & M. Wirsing (Eds.), Web Engineering (Vol. 3140, pp. 354–358). Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/b99180
15. García-Castro, R., & Gómez-Pérez, A. (2010). Interoperability results for Semantic Web technologies using OWL as the interchange language. Web Semantics: Science, Services and Agents on the World Wide Web, 8(4), 278–291. doi:10.1016/j.websem.2010.08.008
16. Google Gson. (n.d.). Retrieved April 30, 2015, from https://code.google.com/p/google-gson/
17. Hees, J. (2015). Rdflib. Retrieved May 24, 2015, from https://github.com/RDFLib
18. Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. MIS Quarterly, 28(1), 75–105.
19. Horrocks, I., Patel-Schneider, P. F., & van Harmelen, F. (2003). 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. doi:10.1016/j.websem.2003.07.001
20. Jackson, S. L. (2014). Research Methods and Statistics: A Critical Thinking Approach (4th ed.).
21. Janev, V., & Vraneš, S. (2011). Applicability assessment of Semantic Web technologies. Information Processing & Management, 47(4), 507–517.
22. Jørgensen, H., & Krogstie, J. (2001). Active models for cooperative information systems. Proceedings of NIK'01.

23. Kommunal- og moderniseringsdepartementet. (2015). Partifinansiering. Retrieved from http://www.partifinansiering.no
24. KommuneProfilen. (2015). KommuneProfilen. Statistikk og nøkkeltall for kommuner og regioner, fylker og landsdeler - basert på offisiell statistikk fra SSB. Retrieved May 05, 2015, from http://www.kommuneprofilen.no
25. Krogstie, J., Lindland, O. I., & Sindre, G. (1995). Towards a deeper understanding of quality in requirements engineering. In J. Iivari, K. Lyytinen, & M. Rossi (Eds.), Advanced Information Systems Engineering (Vol. 932, pp. 1 – 11). Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/3-540-59498-1
26. Krogstie, J., Sindre, G., & Jørgensen, H. (2006). Process models representing knowledge for action: a revised quality framework. European Journal of Information Systems, 15, 91–102.
27. Lindland, O. I., Sindre, G., & Solvberg, A. (1994). Understanding quality in conceptual modeling. IEEE Software, 11(2), 42–49. doi:10.1109/52.268955
28. Liu, C., Wang, H., Yu, Y., & Xu, L. (2010). Towards efficient SPARQL query processing on RDF data. Tsinghua Science & Technology.
29. Moveable-type. (2014). Calculate distance and bearing between two latitude/longitude points using haversine formula in JavaScript. Retrieved from http://www.movable-type.co.uk/scripts/latlong.html
30. Norske postnummer med koordinatar. (2014). Retrieved May 05, 2015, from http://www.erikbolstad.no/postnummer/
31. Norwegian Social Sciences Data Services. (n.d.). NSD Municipal database. Retrieved from https://trygg.nsd.uib.no/kdbbin/kdb_start.exe
32. Oversikt av land. (2010). Retrieved May 24, 2015, from http://www.erikbolstad.no/geo/
33. Pohl, K. (1994). The three dimensions of requirements engineering: A framework and its applications. Information Systems, 19(3), 243–258. doi:10.1016/0306-4379(94)90044-2
34. Riot Games API. (n.d.). Retrieved May 24, 2014, from https://developer.riotgames.com/
35. Shadbolt, N., Berners-Lee, T., & Hall, W. (2006). The Semantic Web Revisited. IEEE Intelligent Systems, 21(3), 96–101. doi:10.1109/MIS.2006.62
36. SSB. (n.d.). Standard for næringsgruppering(SN2007). Retrieved April 30, 2015, from http://stabas.ssb.no/ItemsFrames.asp?ID=8118001&Language=nb
37. Stanford. (n.d.). Protègè. Retrieved April 30, 2015, from protege.stanford.edu
38. Statistics Norway. (n.d.).
39. Stocker, M., Seaborne, A., & Bernstein, A. (2008). SPARQL basic graph pattern optimization using selectivity estimation. Proceedings of the 17th …, 595–604.
40. Studer, R., Grimm, S., & Abecker, A. (2007). Semantic Web Services. (R. Studer, S. Grimm, & A. Abecker, Eds.) (pp. 107–135). Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/3-540-70894-4
41. Suárez-Figueroa, M. C., Gómez-Pérez, A., & Villazón-Terrazas, B. (2009). How to write and use the Ontology Requirements Specification Document. On the Move to Meaningful Internet Systems: OTM 2009.
42. The Eclipse Foundation. (2015). Eclipse. Retrieved February 04, 2015, from https://eclipse.org/
43. Vinmonopolet. (n.d.). Retrieved May 20, 2015, from http://www.vinmonopolet.no
44. W3. (n.d.). RDF Semantic Web Standards. Retrieved March 06, 2014, from http://www.w3.org/RDF/

45. W3C. (n.d.). Semantic Web - W3C. Retrieved March 31, 2014, from http://www.w3.org/standards/semanticweb/
46. Wieringa, R. J. (1996). Requirements Engineering: Frameworks for Understanding.
47. Yang, Z., Jun, M., & Peterson, R. T. (2004). Measuring customer perceived online service quality. International Journal of Operations & Production Management, 24(11), 1149–1174. doi:10.1108/01443570410563278
48. Zheng, S., Song, R., Wen, J.-R., & Wu, D. (2007). Joint optimization of wrapper generation and template detection. In Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '07 (p. 894). New York, New York, USA: ACM Press. doi:10.1145/1281192.1281287