# Developing knowledge models of social media over World Wide Web

by

**Li Jinwu, B.Sc.(Hons)**

**Dissertation**

Presented to the

University of Dublin, Trinity College

in fulfillment

of the requirements

for the Degree of

**Master of Science in Computer Science**

**University of Dublin, Trinity College**

September 2013

# Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Li Jinwu

August 29, 2013

# Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Li Jinwu

August 29, 2013

# Acknowledgments

I want to say my sincere thanks to my supervisor, Melike Sah, who spend so much effort in teaching me how to research and giving me so many valuable suggestions. My thesis is based on her idea and her assistance. And my parents, my friends and everyone who support me during the course, I feel life in post-graduate is so meaning when you beside, me.

Li Jinwu

*University of Dublin, Trinity College*
*September 2013*

# Developing knowledge models of social media over World Wide Web

Li Jinwu, M.Sc.
University of Dublin, Trinity College, 2013

Supervisor: Sah Melike

User generated content from large Social Network is considered an important knowledge source in the future. The data is inherently unstructured or semi-structured.

In this project, we perform a case study on LinkedIn Ireland public profiles, to investigate how to develop a reusable knowledge model for it. Apart from the search engine in LinkedIn.com itself, there's no well known public available endpoint that allows users to query knowledge of interest from LinkedIn. We present a system that download and convert the raw web pages from LinkedIn to a machine-readable, interoperable format using Data Mining and Semantic Web technologies. One of the outcomes from this project is a dataset that contains facts about Irish industry and education in a structured manner.

The resulted dataset is publicly accessible SPARQL endpoint, everyone who interested in facts about Ireland can use HTTP request to access it.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Background

Social media like Facebook and Twitter are gradually changing the world, and becoming a new source of knowledge. Users generate the data with a feeling of rewarding, since they can get recognition and interaction from other users[1]; but for practitioners in Information Technology (IT) industry, User Generated Content (UGC) means a new, inexpensive and fast way to obtain data that is barely impossible in the past.

LinkedIn.com, the world's largest profession network[1], contains a large amount of hidden career and country base industry information, but yet waiting to be discovered. Unlike Facebook and Twitter, the study of LinkedIn.com is not getting as much attentions as it should be.

Semantic Web, as believed by many researchers, will be the "Next generation of knowledge representation and processing technology"[2]. It aims to extends the world of human readable web page and documents to a new world of machine understandable, interoperable metadata.

## 1.2 Motivation

The lack of study of LinkedIn.com provides an opportunity. This project use LinkedIn Ireland as the study subject, and aims to develop a reusable, queryable knowledge model for LinkedIn public profiles. The importance of the project can be described in several aspects:

Firstly, Getting fully insights about Ireland industry distribution, personal skills and professionals' education background are always important for a number of people. And

[1]According to their official website: http://www.linkedin.com/about-us

it can be easily to scale to LinkedIn worldwide.

Secondly, it's a good complement for government statistics about industries and professionals. The effectiveness and timeliness of UGC can always guarantee we are getting the first hand data.

Thirdly, at the end of this project, an online public dataset will be provided so that everyone who interest in Ireland facts can query the public endpoint to get information.

Finally, user interface can built on top of the knowledge model and the dataset to support complex queries and answer questions. Government, practitioners in Human Resources (HR), and job seekers are possible target audience.

## 1.3 Research Question

According to the background and motivations, this thesis address the following research questions:

1. Can we take advantages from Semantic Web technologies to build a knowledge model and generate user dataset from the public data provided by LinkedIn.com?

2. In order to achieve interoperability and common agreement, can we reuse existing ontologies and public vocabularies and integrate them into our model?

3. How useful and complete is the extracted data?

4. Is that possible to have a working user interface that make use of our dataset and demonstrate some useful use cases?

## 1.4 Contributions

### 1.4.1 Reusable knowledge model from LinkedIn public profiles

We present a queryable, extendible knowledge graph (Figure 3.1) that capture the data relationship of LinkedIn.com public profiles and company profiles. It can be used by LinkedIn internally or by other researchers who also interested in user generated content in LinkedIn.com.

### 1.4.2 Public online SPARQL endpoint for complex query about Irish industry

We publish our SPARQL endpoint at http://goo.gl/5HyziV. It's a standard SPARQL endpoint that powered by 4store[3]. Our endpoint accept HTTP POST requests and support a number of RDF format, such as XML, JSON, plain and turtle. Anyone who interested in discover Irish industry and college facts can use this service.

### 1.4.3 Scalable crawling strategy

The crawling strategy is scalable to any numbers of LinkedIn public profiles. If we consider a profile is a node, since in the profile, LinkedIn will suggest 6 to 8 similar profiles (nodes), the graph is expanding very quickly (exponential increase). Therefore, data mining practitioners (and other researchers) can make use of our strategy, as discussed in chapter 4, to download profiles in any subdomains of LinkedIn.

### 1.4.4 Mashup based city information extraction strategy

Based on our user study, the strategy is widely accepted by our survey participants, with average of 0.85 F-score and 4.089/5 user rating. The approach can be generalise to get city information by company names. Another approach is to use Google reverse Geocoding service[2]. However, even we don't have comparison result on hand, the result provided by reverse geocoding service seems to worse than using a country's yellow page database.

## 1.5 Outline of the thesis

Chapter 2 discusses the state of the art in Data extraction and Knowledge modelling.

Chapter 3 covers system requirements, system architecture and design decisions.

Chapter 4 provides details about the implementations, including profile crawling strategy, data extraction, data normalisation and missing fields inference.

Chapter 5 evaluates the extracted data accuracy, LinkedIn profile completeness, data linkage and data fitness.

[2] https://developers.google.com/maps/documentation/javascript/geocoding?csw=1#ReverseGeocoding

Chapter 6 concludes the results and contributions, and future works will be discussed in this chapter as well.

# Chapter 2

# State of the Art

## 2.1 Introduction

Social media has become an important source of knowledge, user-generated content has a great potential useful data in terms of business opportunities and research data source. In this dissertation, the author performs a case study on Linked.com, a leading websites in Social Media, and build a knowledge model for the company and professional public profiles. The potential use of the dataset could be similar to [4], where applications will be built on top on the dataset to provide the user with customised data aggregation.

This project focuses on developing the knowledge model representations of user generated content in the context of Semantic Web. Semantic Web can be regarded as an revolution from Web of documents to Web of data and knowledge.[2] The key factor that differentiate it from traditional web is that, it guarantees machine-readable data that supports automatic reasoning. It increase the interoperability of the data by defining the semantic meanings. Normally, The Resource Description Framework (RDF) is used to describe the resources.

In the context of the Semantic Web, there's a movement that tries to encourage information holders to publish and link their data together; it's called Linked Data. More and more people contribute to the Linked Data Cloud[5], for example, government Linked Data has already been maintained by W3C.org, Ontologies and RDF are heavily used in Biomedical domain, the FOAF project has already attracted Social networks to use it to model the users, and the DBpedia, the Semantic version of the Wikipedia, has become the centre of the Web Ontologies[6]. So we decide to build our knowledge model using RDF, because it can take the advantages of Semantic Web, to support reasoning and machine auto-processing. Apart from that, SPARQL Protocol and RDF Query Language (SPARQL), can be used to infer the facts from RDF triples.

This project focuses on developing the knowledge model representations of user generated content in the context of Semantic Web. Ideally, the data model should be general enough so that new knowledge can be inferred from the extracted data. Because we are using RDF triples to represent data, SPARQL will be used as the query tool to answer questions.

In order to generate knowledge models from raw HyperText Markup Language (HTML) files of LinkedIn public profiles, a number of challenges are required to be addressed, such as Data Extraction, Knowledge Modelling, Content Integration and Evaluation of Extracted Data. In the next section, we discuss each challenge in detail.

## 2.2 Data extraction

### 2.2.1 Data extraction in general

[7] provides an up-to-date survey on web data extraction. In this paper, three common techniques for web data extraction is listed: 1. Tree-based approach: analysis on Document Object Model (DOM) trees. 2. Web wrapper: use procedures to seeks and finds data required. 3. Machine learning approach: using reasoning or other Artificial Intelligence (AI) techniques to find the data of interest. In addition, the paper provides a full list of famous applications that are being used in the real world. In our approach, as we can only access to the HTML files of LinkedIn public profiles, Web wrapper method will be used to extract data. Although the pages do not contain structure knowledge, the format are consistent and barely change. Even some profiles are incomplete, we can handle this in our Wrapper program.

[8] discusses four challenges or concerns that every research will encounter in the field of Semantic Web and Big Data.

1. Michael L. Brodie mainly focuses on data integration. He also provides a general form for it: 1. Define the concern. 2. Search for candidate data elements. 3. Extract, Transform and Load (ETL) the candidate data into appropriate formats. 4. Entity resolution to get unique, comprehensive data. 5. Answer the query/solve the problem.

2. Christian Bizer tries to motivate people to take the Billion Triple Challenge (http://challenge.semanticweb.org/). The challenge is about using pre-crawled data set to translate different vocabularies into uniform one, discover resources and fuse descriptions into an integrated representation. So the main challenges here are: 1. Large-scale RDF processing. 2. Data quality. 3. Data Integration.

3. Peter Boncz proposes the Linked Open Data Ripper, a web portal to combine open government data. The main challenges are the accessibility and the usability of the public government data. He is looking for robust, reliable user interface(s) (UI) that integrate Linked Data from multiple sources and allow users to query the data more easily.

4. Orri Erling believes systematic adoption of Database Management System (DBMS) technology into Semantic Web could be a potential opportunity, since efficient storage and query of DBMS has been researching for decades. A lot of optimisation mechanisms, performance tools have been developed to support the system. The challenges exist are: 1. we need to demonstrate the benefit of semantics. 2. smarter database is required for reasoning, but Web Ontology Language (OWL) is not enough. 3. we need to bring Linked data and RDF into the regular data-engineering stack.

These challenges are interesting topics that waiting to be addressed. Nevertheless, it provides a brief overview of the current status of Big Data stack.

[9] gives a relative short introduction of several ways to mine data from LinkedIn.com, typically, LinkedIn Search, raw data processing, and third-party tools. Among these approaches and tools, the Natural Language Toolkit (NTLK) and [10] are two resources that worth to study.

### 2.2.2 Approaches

[11] approaches the problem of web table data extraction by using two-dimensional visual box model. This paper introduces extracting information from a high level of visual features. It uses the representation of web browser rendering, and save the practitioner from parsing low level Cascading Style Sheets (CSS), JavaScript, HTML tags. The key difference is that, the traditional approach uses tree-based representation of web pages, such as HTML or Extensible Markup Language (XML), so the whole information extraction is processed in low level, using HTML/XML parses. As far as the author can tell, this approach only works for tables and lists, so it cannot be applied to arbitrary elements on web pages.

[12] discuss about automatically extracting concepts from semi-structured data, specifically, they use PowerPoint slides as the knowledge source. They combine ontology learning and natural language processing techniques to produce the knowledge representation. The process as follows: 1. normalising the text contents by splitting statements, replacing non-alphanumeric symbols, expanding abbreviations, etc. 2. creating parse tree for sen-

tences. 3. defining a set of weighting models. 4. Extracting text features (e.g. topic, title, bullet, sub bullet) for each term and applying "link-distance algorithm" to determine to correct concepts. What can be learned from the paper is that they effectively use Natural Language Processing to tag each term and then define weighting models to hierarchically extract concepts using text features. But the problem still exists, that is, the 42% of overall performance (F-measure) is not enough to apply this techniques into real world E-learning application. Apart from that, in their future work, they plan to introduce multi-media feature extraction into the their paper. The author believes the high values of F-measure is very important for real use of this technique, which is the thing that this paper cannot handle.

[13] presents a framework that exploits the Web documents using a "Tree Alignment Algorithm", in which they build trees iteratively and try to find record boundary and repeating patterns. Then they build "conceptual graphs" to represent domain knowledge. Finally they map the conceptual structure to the extracted data items. Because the conceptual graph is directly mapped to a database schema, this approach can reduce the time of converting the extracted content to database records. The approach proposed here could be very useful in this project, which also trying to extract data of interest from semi-structure LinkedIn profile files. However, as far as the author can tell, the approach might be not scalable, as manually creating a "conceptual graph" is required, which makes the approach no better than using pure "Regular Expression" approach. Nevertheless, we can learn from the "mapping" process and adopt it. In our case, Levenshtein distance (Edit distance) or Cosine similarity (Vector space model) could be used to classify vocabularies and correct typos.

[14] describes a method to populate Wikipedia info-boxes from Wikipedia article. It trains "value extractors" from training data using structural analysis. Structure discovery algorithm is used to overcome the shortcomings of regular expression, in which it tries to merge important patterns from a frequent pattern list. One thing is not clear in this paper is how to choose correct attribute value among a list of potential attribute values. It does mention using "Conditional Random Fields" (CRF) to learn label tokens based on features. "Combining regular expressions" provides better results, it worth further investigation.

[15] talks about metadata extraction from enterprise content. It performs a case study on documents that described by Docbook DTD, which is used widely by many organisations. The motivation of the paper is to provide a novel framework for personalised information retrieval (IR) system. It also generate an Ontology for user modelling. This approach is deeply couple with the Docbook content, similar approach might be used in this project as our data are deeply couple with LinkedIn html structure.

## 2.3 Knowledge modelling

We are living in the era of Web 2.0, which means that large scale of the user-generated content are available on the Internet in a loose or semi-structure format. Traditionally, Data Mining is performed on relational databases or data warehouse, in a way that practitioners look for unaware patterns internally. But gathering data from blooming Social Media websites cannot be fully addressed in the traditional approach, as most of the websites are producing HTML or XML files. A mapping between raw data format and relational database table is required but hard to generalise to other data consumers.

That's why we need Semantic Web. Semantic Web aims to replace the Web of documents to the Web of machine processable, automatic reasoning web services or web databases. It provides interoperability to data by strictly narrow down the data into triples and allow each piece reference others using unique resource identifier. The potential of Semantic Web is difficult to estimate, it might totally change the development paradigm[16]: data drive possible applications instead of what we do today, applications determine data format.

### 2.3.1 Semantic Web technologies

Semantic Web technology stack can be thought as a layered graph as shown in Figure 2.1. We are going to introduce some of the technologies in this section.

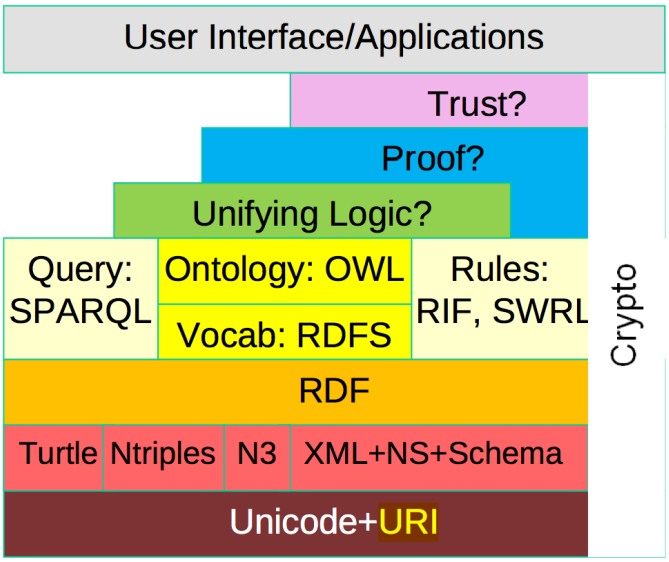


Figure 2.1: Semantic Web Stack[1]

[1]Copy from Dr. Rob Brennan's lecture notes.

**Uniform resource identifier (URI)** : A string that can be used to uniquely identify a web resource. According to [17], URI is considered as the standard resource identifier to represent any HTML or RDF object or concept. The reason behind that is others can easily access the resources using Hypertext Transfer Protocol (HTTP) requests.

**Resource Description Framework (RDF)** : RDF is a graph-based data model that used in Semantic Web. It represents knowledge using a triple structure. An expression in RDF is a "subject-predicate-object" triple. It can be represented by a graph where the subject and the object is the start node and end node and the predicate is the link. Nodes can be a URI or a literal. It has a variety of notations such as N3, Turtle, and XML, but they are all interchangeable. Notice that it just a data model that allows us to describe things that in a specific syntax, but has no assumption.

**Resource Description Framework Schema (RDFS)** : It intends to provide vocabularies to standardise the structure of RDF resources. It is a set of classes and properties that use the RDF language to provide basic ontologies. The reason we need these vocabularies is that RDF Triple itself is not informative enough. Different datasets need a standard (just like protocol) to communicate, otherwise, no one will understand the "semantic" of other datasets. So RDFS specifies a basic vocabulary such as: subClassOf, DataType, domain, range, etc. to structure the RDF resources.

**OWL** : OWL aims to add more constraints on RDFS to describe resources in details. For example, owl defines disjointWith, complementOf, equivalentClass, and cardinality on top of RDFS. It makes the triple expression become more expressive and specific.

**Link Open Data Movement** : It's a community effort starting in 2006 to unlock hidden semantics in a way that making RDF publicly available using open standards and protocols. Many open datasets were published by these efforts from many domains such as geographic locations (e.g. Geonames), general knowledge (e.g. DBpedia), broadcasting data (e.g. BBC), bioscience, etc. The best way to feel the impact of the Cloud is to visualise the graph (Figure 2.2):

[18] demonstrates how to collect, analyse FOAF documents. According to the paper, FOAF is one of the most popular ontology that being used at the moment. One of the main produces of FOAF documents is blog website. It's easy to use FOAF specific tags to identify the documents, and looking for patterns. Apart from above, the reader

[2]Attribution: "Linking Open Data cloud diagram, by Richard Cyganiak and Anja Jentzsch. http://lod-cloud.net/"

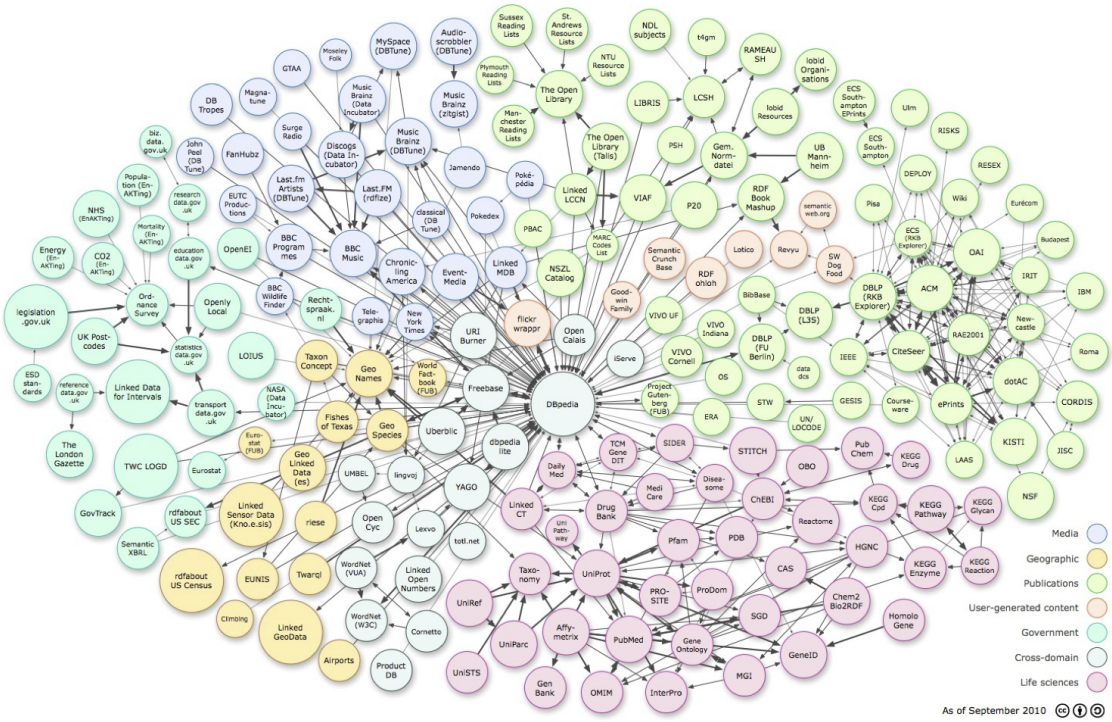


Figure 2.2: Link Data Cloud[2]

known from it that LinkedIn.com also use the FOAF ontology, but they protect the FOAF documents from public access. This paper implies that we can use FOAF Ontology to describe LinkedIn public profiles and extend it if necessary.

[5] provides a comprehensive state of the art on the Linked Open Data (LOD). It introduces "Linked data principles", how to publishing Linked Data, publishing tools, existing applications. Also, related developments and research challenges are given to guide the later researchers.

### 2.3.2 Linked Open Data

As mention in the previous section, Linked Open Data is a movement that data providers start to publish and link their data to each other in RDF format. It enables[19]: sophisticated data processing, connecting distributed data and change the world from Data Islands to a Global Data Space. The bar for publishing the Linked data is not restrict, as it only needs to conform following basic principles[17]:

1. Use URIs to name the things and use HTTP URIs to guarantee accessibility.

2. Provide standard information when users access it (RDF*, SPARQL).

3. Include links to other URIs.

As more and more important websites join in this movement, LOD is becoming a huge knowledge graph as shown in Figure 2.2. DBpedia, a RDF version of Wikipedia, now become the centre of Open Data Cloud. A lot of tools have been built for LOD[5], for instance, Linked Data search engines, allow you quickly look for the documents you are interested in; Publishing tools, allow you quickly publish the data in RDF even though the origin files in display in other format. Our project aims to contribute to the LOD in a way that we provide queryable web service that allow people to discover important facts and statistics.

[6] provides a brief introduction about the DBpedia. It firsts talk about the extraction of structure information from Wikipedia, which is followed by a list of datasets. Finally, it talks about how to access, query the dataset online (using HTTP, SPARQL endpoint and RDF dump), how DBpedia interlink with other open datasets, and how to search DBpedia.org using built-in user interface. Through the paper, the authors try to convey a fact that DBpedia is the nucleus of the Web of Data, which is a reasonable claim.

[20] talks about a live extraction framework that can consume Wikipedia updates and reflect on the DBpedia triple store. The key process is as follows: 1. Use different extractors to deal with different types of content. 2. Assign states to extractors, namely, an extractor could be in either "updated", "not modified", or "remove" state. 3. Apply heuristic method (by comparing current Axiom to previous one) to minimise the number of triples that need to be updated. To increase the effectiveness of "mapping" between Wikipedia and DBpedia, templates are introduced to infer the correct attribute names and correct values. Keeping DBpedia content up-to-date has several benefits, such as enhancing the integration with Wikipedia, increase the use of DBpedia in live scenarios. So later if the project want to keep the Ontology and triples up-to-date and reflect the instant change in LinkedIn.com, using the approach mention in this paper could be a potential solution.

[21] gives a detailed introduction of DBpedia Spotlight – a Web Service to detect DBpedia resources in text. The key improvement of the disambiguation process is: instead of using traditional "TF-IDF" to weight the words, it uses "TF-ICF" (term frequency-Inverse Candidate Frequency). Moreover, to maximise the annotation result, the authors suggest use customised configuration when annotating. This web service could be very useful when later the reader tries to annotation the data fields in LinkedIn public profiles.

### 2.3.3 Building ontologies

We can think of it as a collection of terms that defines the concepts and relationships of an area[3]. It is the cornerstone of the Semantic Web; by publishing ontologies and combining them together, the web of knowledge will finally be constructed.

[22] mainly focus on the strategy of building simple Ontologies for social networks. A tripartite model is suggested in this paper, specifically, an Actor-Concept-Instance model. The paper demonstrates the applicability of the model using two examples. The paper also shows how the ontology is emerged based on the model and how it is extended to support Ontology Extraction from Web Pages. However, this approach mainly about Community Ontology Construction, as LinkedIn public profiles has no or very limit connection information. In our approach, we will try to enhance linkage/mapping to other datasets, like DBpedia (for general information), Academic Institution Internal Structure Ontology (AIISO) (for academic skills, courses), Freebase (for general subjects), etc.

[23] focuses on extracting information from Artificial Intelligence related conference and workshop and building an Ontology for AI. Again, it constructs domain concept knowledge from nested tags. for example, in HTML, <h1>means a more general term than <h2>, so an instance of <h2>is a subclass of an instance of <h1>. Then in the optimisation process, it performs "ontology pruning and union" to handle concept duplication. However, this strategy might result in wrong classification. To summarise, this approach is very useful provided the user knows the contents in the web pages is valid for hierarchical classification. It could not be generalised for other loose structured websites.

In this project, our goal is to build an Ontology for LinkedIn public profiles using automated process. The reasons for doing that are, firstly, Linkedin.com is one of the main knowledge sources for professional information. People publish their education, skills, experiences on the site and we expect these kinds of information can answer a lot questions. For example, decision maker may want to track the trending of an industry by looking at the number of employees and the number of new startups in the specific area. Secondly, we choose Semantic Web because we want to link the knowledge into the Web of knowledge (LOD) to maximise the usability of our data. The interoperability feature provided by RDF can lead to flexible use of triples (Again, in this case, data can drive the application developments)

[3]http://www.w3.org/standards/semanticweb/ontology

## 2.4 Content integration and classification

One of the major problems in Information Extraction (IE), especially in social media information extraction is the variety of the similar words. For example, in LinkedIn.com, a user can claim himself as "Graduated from Trinity College Dublin", meanwhile, another user will say she is studding at "TCD". When we build an ontology and try to link our data to LOD, we really have to be very careful about declaiming a term more than once. A false positive result is also not acceptable, in a way that we might misclassify address "Dublin" in "Dublin Core" as the capital of Ireland. So finding ways to clean up the data and classify them correctly are considered two complex tasks in IE.

[24] gives a very comprehensive introduction about machine learning in text categorisation. document indexing and dimensionality reduction are common techniques to increase the effectiveness of accessing data. Probability classifiers, decision tree classifiers, on-line methods, neural networks, etc. At last, measures of effectiveness was discussed. At the moment, we will not try to parse the "Summary" section in public profiles (In LinkedIn profiles, the summary section is where users write "abstract" about themselves). But if we need more detail knowledge for the Ontology, we might use the approach listed in this paper.

[25] proposes an approach to build re-usable dictionary repositories for text mining. The key idea is to build a new dictionary by using synonyms from existing dictionary. They only use synonym relations, which cannot be enough to represent more complex semantic relations. And if the practitioner choose an inappropriate dictionary to start with, he will end up getting nothing back since the similarity value is too low. Apart from that, according to the authors, the idea of generating text corpus for the existing dictionaries can save about 50%-60% of time.

[26] talks about how to integrate government data from different data sources. The integration flow is as follows: 1. Mapping and Scrubbing. They maps attributes to a simple global schema, and cleansing on data value level. 2. Data Transformation, in which they transforms the source data structure to the global schema and separates data of different types. 3. Deduplication. A tool called Duplicate Detection Toolkit was used to match across data sources. 4. Entity Fusion. They fused the matched entities to obtain a single representation. "Dempster-Shafer-Theory" is used to induce weights for attributes. We can investigate the mapping process since we will require map person to other linked dataset instances.

## 2.5 Evaluation of the extracted data

As everyone can publish their data on the Internet, the evaluation of the data quality becomes a very important aspect in Ontology building. People cannot or hard to reuse the data with bad quality, so publishing the data without quality assurance will significantly reduce the value and the reusability of the data. Therefore, we evaluated some metrics and a data assessment framework:

[27] lists quality metrics for metadata. This project can use some of these metrics directly to evaluate the quality of the result of the data mining. Data Completeness is achieved by comparing extracted data with the "ideal" representation. Data Accuracy can be achieved by the degree of correctness. In this project, it's possible to compare manually collected data with the automatically extracted data. We can use user studies, by introduce volunteers to extract the data. Then by investigating the manually collected results to machine auto-generated ones, we will have some confidence about our data correctness. Conformance to expectations is a way to test whether the schema meets the requirement of use cases, and supports arbitrary complex queries. Because our dataset will be used by another project: "Leveraging Power of Social Media and Data Visualisation", we can evaluate the dataset by looking at whether the data is complied with the user and visualisation requirements. So, the metrics list in this paper can evaluate the quality of the data.

[28] presents a Linked data quality assessment and fusion framework that can be used to measure, express the quality of data. It's a part of the Linked Data Integration Framework (LDIF). The integration process works as follows: 1. access web data, 2, map the vocabulary from different schema using R2R framework. 3. LDIF also resolute multiple identifiers for the same entity by using "Silk-Link Specification Language". 4. the data quality assessment module contains a set of scoring functions, and it also support user-extend scoring function and customisation. 5. finally, the data fusion module includes conflict ignoring, avoiding and resolution strategies to "sieve" the data and generate a cleaner representation. Since this paper focus on both quality measurement and data fusion, what we can use from this paper is the Data Quality Assessment module. It's possible to use the built-in scoring functions directly or implement new methods.

## 2.6 Chapter Summary

In this chapter, we discussed and evaluated state of the art in Data extraction, Knowledge modelling, Content integration and Evaluation method. We looked at different approaches in Data extraction, and we decided to use the parser approach in our development. We

introduced the fundamentals in Semantic Web technologies, which will be applied in the next chapters. We discussed the methods in content integration, and the possibility of using them in our project. We also talked about the evaluation methods that are useful in our system evaluation. Notice that the approaches discussed in this Chapter by no means must be used during our system development, but understanding their pros and cons can help us make decisions.

# Chapter 3

# Design

In this chapter we discuss the requirements of the system, both functional and non-functional, the knowledge model we designed and also the software architecture. In the architecture, we use modular programming paradigm for flexibility purpose. We also talk about decisions we made and also provide reasons to justify our decisions.

## 3.1 Requirements

In order to answer the research questions and produce public dataset at the end, the system should be capable to:

1. Have a knowledge model describing how the data should be stored (in RDF format).
2. Download LinkedIn personal public profiles and company public profiles, the data should be in HTML format.
3. Extract data from the raw HTML files.
4. Normalise the metadata to provide consistent and structure output, which means the system should be able to correct dirty data. This is important for upper layer, such as UI that powered by SPARQL query, as minimum efforts are required if we have normalised data.
5. Convert the data into RDF triples, and store it in a public accessible triplestore.

Additionally, the system also have a number of non-functional requirements that need to be fulfilled:

**Crawling performance and parsing performance**

The system should be able to download and parse enough profiles in a small number

of days. Because timeliness is the nature of user generated content, if it takes too long to do this, we lost the chance of getting first hand information.

**Query performance**

The system should be able to respond to the query from user interface quickly enough to make sure the UI usability.

**The choice of programming language(s)**

The ideal programming language(s) for this project should be dynamic typing, widely used, cross platform scripting language. The reason behind that is we want quickly iterate the program. As parsing and converting user generated content are hard, we cannot make assumptions about anything. Therefore, statically typed, compiled programming language (such as C++ and Java) is incapable for this task. Perl, Python are two possible options. But since Python is easy to learn and have rich communities support in terms of Semantic Web and Natural Language processing, we finally decide to use Python to implement our parse and RDF converter.

## 3.2 Knowledge model

As discussed in the previous chapter, we decide to use Semantic Web technologies for knowledge modelling. The first thing we need to do is the come out with an Ontology that can reflect the actual state of LinkedIn personal profile and company profile. After investigate with samples and discuss with the upper layer UI designer, we come out with Figure 3.1

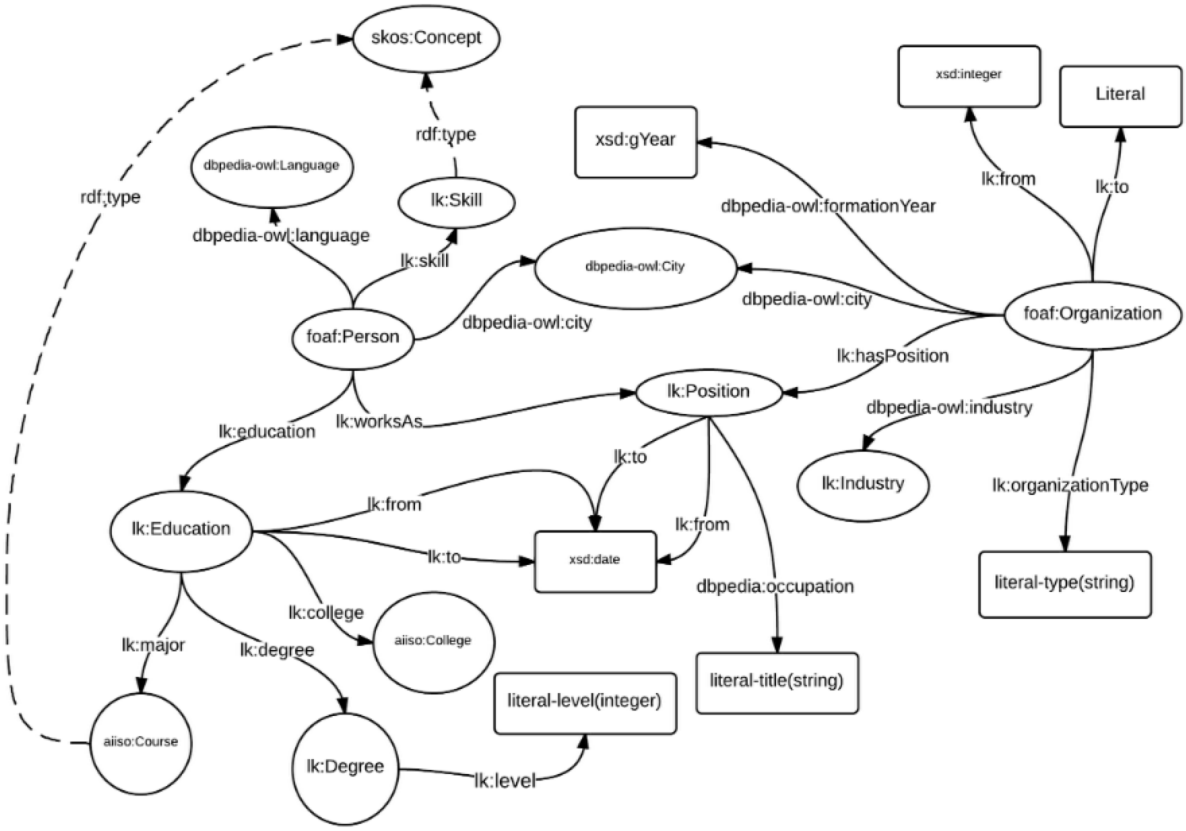


Figure 3.1: Knowledge Model for Linked public profile and company profile

As shown on the figure, the model can be divided into three cores: Person, Education and Organisation. In LinkedIn personal profiles, a person might have current living city, skills, work position, job title, start date and end date of the position. Besides, a person will have education background, such as college name, major, degree and start year and end year of the college.

For a LinkedIn company profile, it might have, headquarters, company type (public, private own, etc.) and industry type (e.g. IT) and company size.

Our knowledge model links the personal profile with company profile using "position". The whole graph is linked so that we can perform arbitrary query. For example, we can discover the relations between education background and organisation through person and position.

One key thing to note is Semantic Web is built around the idea of triples, which means an expression has the structure of "subject", "predicate" and "object". In the graph, the names in circle are "Class", the link between two "Classes" is call "Property" (predicate), it is used to link an instance of one Class to an instance of another Class.

### 3.2.1 Reusing existing ontologies

Ontology reuse is an important concept in Ontology Engineering. According to [29], it increases the quality of the application, achieves interoperability, improves cost in ontology development and helps applications agree on the domain concept. One mission of Semantic Web is to achieve data interoperability. If two applications cannot understand the meaning or the semantic of data from the other side, then these two applications cannot communicate. Therefore, we need to try our best to reuse existing, well known Ontologies so that other Semantic Web application can at least partially understand our domain, hence reduce the needs of Ontology mapping.

In Figure 3.1, we partially reuse following vocabularies or ontologies: Simple Knowledge Organization System (SKOS), AIISO, DBpedia, Friend of A Friend (FOAF). However, the reasons for reusing them are varies:

SKOS

Just as its name suggest, SKOS is used for "knowledge organisation", such as "structure vocabularies" and "classification schemes"[30]. We can use this language to define academic disciplines (major). For example, one LinkedIn user study "Computer Science", another one study "Artificial Intelligence", if we define, in a SKOS manner, say "Computer Science" is a "broader" term of "Artificial Intelligence", then our system can handle the discipline hierarchy. It also good for our endpoint users. Since hierarchy meanings both general and specific, so users who write SPARQL queries to get knowledge from our system can have more simple but detailed control of what should be returned.

AIISO

We make use AIISO class definition to define our education part. As the Ontology is used to describe the internal state of an Academic Institution, it has definition about college, course (we also called major or discipline), and degree. So we don't need to create our own concept (since no one know anything about our ontology) about education.

DBpedia

In the previous chapter, we already introduce the importance of DBpedia. The more terms we can reuse from DBpedia, the more interoperability we obtain. We use both class definition and property definition from DBpedia, they are: City (Class), city (property), industry (property), language (property), Language (Class). With FOAF and AIISO, these three ontologies form a backbone of my knowledge model.

FOAF

FOAF is another widely used ontology, it is used to describe and link the data in Social network[31]. We use foaf:Person to define our LinkedIn user, and use foaf:Organization to define the company in LinkedIn work experience.

Even though we couldn't find and reuse enough existing properties in our knowledge model, but my major Class are all from well known ontologies, which means there is a chance to allow other applications to discover our contents.

## 3.3 System architecture

According the requirements, we design the system as Figure 3.2 shows:

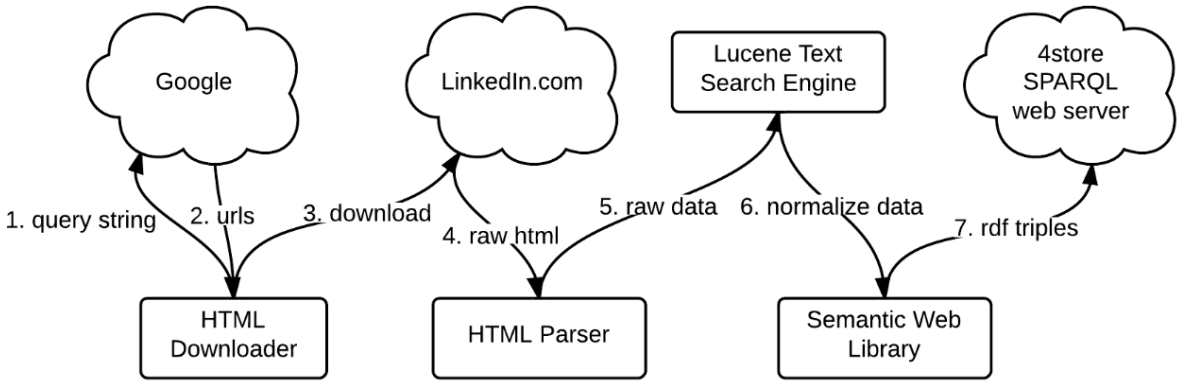


Figure 3.2: System Architecture

As LinkedIn.com will not provide Application programming interface (API) for downloading their public profiles, we need to do it using Google search result.

So the system flow will be as follows, this description is a brief overview of the system process, more details will be given in the next chapter:

1. Use Google search engine to query for profiles in LinkedIn Ireland.
2. Get the response from Google, and download the html files base on the Uniform resource locator (URL) in response content.
3. Call the parser to parse the HTML files and get the fields that will be used, as shown in the Figure 3.1.
4. Then the data extracted will be sent to a data normalisation module (Lucene search engine in this case), where the data will be cleaned up and normalised.

5. Finally we can build the RDF triples using the normalised data and put them into our triplestore.

## 3.4 Design decisions

There are several decisions we make to guarantee the final results come out smoothly.

Firstly, we decide to separate the profile downloader module with the profile parser module. It means we first run the module and download enough profiles, then we start to call parser module. The reason behind that is downloading profiles will consume a large amount of Google and LinkedIn resources, which implies that our connection can be switched off at any time by remote server. Therefore we don't want to do parsing together with downloading, since we don't want to spend extra effort in Network problem handing. Another point is that our final result will be in RDF format, hence if we do parsing together with downloading, extra storage and data structure is required to store this intermediate result.

Secondly, a simple database is required to keep track of which url is downloaded, which profile is parsed and had been converted into RDF format. Because we are handling a very large amount of profiles, and the correctness of the parse has to be adjusted during parsing, so it's unrealistic to assume that our program can parse and convert all the data with one-click. For example, if an unhandled exception occurred and stop the program, we can start parsing the remaining profiles if we have a database keeps track of the status.

Thirdly, just as the first reason we highlighted, the parser module, data normalisation module and RDF conversion module will form a pipeline. It means the output from previous module will be feed as input of the next module. The reason is, we don't have to store the intermediate result from the previous two modules.

## 3.5 Chapter Summary

In this chapter, we clarified our functional and non-functional system requirements. We introduced our knowledge model for LinkedIn.com personal public profile and company public profile, and went through the structure of the mode. We discussed the existing, well-known ontologies we used. And finally, we specified some decisions that will guide our implementation.

# Chapter 4

# Implementation

In the previous chapter, we specify the requirements of the system. Therefore we can start implementing each module to meet the needs of our design. In this chapter, we will discuss the programming languages and corresponding libraries we used, and some reusable strategies. We also provide the system environment as reference.

## 4.1 Programming languages and corresponding libraries

### 4.1.1 Python and its libraries

As discussed in previous chapter, We used Python as the main programming language. During the implementation period, the below is a list of Python libraries that helped to finish this project:

**Beautiful Soup**

A easy-to-use library for parsing HTML documents. It provides both flexibility and performance for parsing HTMl or XML files.

**RDFLib**

RDFLib is a library that allow Python to manipulate RDF files. It has RDF parser, serializer and also SPARQL 1.1 implementation. This library is all we need for RDF format conversion. And we can also test our data with its built-in SPARQL query API.

### 4.1.2 Java and Lucene text search engine

For data normalisation, the final decision is to use Lucene text search engine. According to [32], Lucene is a simple and power API for full-text indexing and searching. But we

don't need the most powerful part of it, we only take the advantage of its keyword search and fuzzy search (by using Edit Distance[33]).

Another option is pylucene. It is the Python wrapper for Lucene text search engine. However, the project seems to only working on some particular version of Java Virtual Machine (JVM), which means that it is hard to migrate to arbitrary machine.

Therefore, we use Java to implement our data normalisation engine and use socket to communicate with our Python modules. The process of the socket communication in shown in Figure 3.2.

### 4.1.3 Modular porogramming paradigm

Instead of using popular Object Oriented paradigm, we use Modular programming in developing our system. Python is particularly suitable for this programming paradigm. The main idea is to break a large system into each individual, separated module to accomplish one specific function. The advantage for using the paradigm is that we can debug each module separately so that error data (as it's very common in Data Mining) will not propagate across the system.

### 4.1.4 System environment

The technologies and libraries used in this project are all open source. In order to utilise the command line and built-in tools, UNIX-like system is used in development but there is not problem in running the code on Windows machine. It requires Python2.7 and Java1.7.

The production environment is Amazon EC2 64 bit Ubuntu12.04, Intel Xeon Central processing unit E5-2650 2.00GHZ, 4G memory.

## 4.2 Strategy

Two strategies are worth mentioned during the development. One is the strategy of scalable profile downloading. With this approach, one can easily download any number of profiles in any LinkedIn subdomain.

### 4.2.1 Profile download strategy

In LinkedIn API documentation[1], there is no method for one to download public profile without OAuth Authentication. One thing to note here is we will not disclose LinkedIn

[1]LinkedIn Developers' Documentation: https://developer.linkedin.com/apis

users personal information and we are downloading the public profiles, therefore we are legally valid to perform this action.

Inspired by [4], we decide to combine google keyword search and google site operator to get the profile urls. Hence, the google query is constituted by two parts: the keyword part and the query domain part. In order to get more query result to improve the performance of the profile download module, we decide to use common Irish names as keywords. For the second part, after we carefully investigate same samples, we decide to use "http://ie.linkedin.com/pub/" as the site operator. The url means: "public profile directory for LinkedIn Ireland". Therefore, the final query string is: "Name site:http://ie.linkedin.com/pub/".

Besides, every downloaded LinkedIn personal public profile has a section called "Viewers of this profile also viewed...", in which LinkedIn will suggest around six to eight similar users to the viewer. Therefore, one downloaded profile can link to 6 to 8 profiles, we can do this again and again. Even conflicts might happen but ideally, we can download most of the profiles in LinkedIn Ireland.

### 4.2.2 City extraction strategy

We need city information in our RDF triplestore for every profile, if possible. The reason for that is the upper layer user interface(s) is designed around comparing different numbers across cities. Therefore, without the city information, our triplestore cannot reflect facts of interest.

The strategy is quite simple, it constitutes of three cases:

1. Case 1: If the person's current living city is shown on the profile, we use our HTML parser to get it.

2. Case 2: If the information is not in the profile, we can sort the companies that this person had worked in, in reverse chronological order. For each of the company, we query goldenpages.ie[2], which contains most of the companies' information. We get the first return city as the person's current living city, and also set all return cities as the company's located city. (As a large amount of company will have offices in different city.)

3. Case 3: If the person has no work experience or the company he worked in doesn't register on goldenpages.ie, we get the city base on his education experience, in reverse chronological order, again.

[2]http://www.goldenpages.ie/

In the next chapter, statistics will be shown to demonstrate the strategy is working well based on user study.

## 4.3 Chapter Summary

In this chapter, we discussed the programming languages and libraries we used. We introduced Modular programming paradigm and justified our choice. We also discussed about the production system environment, profile download strategy and city traction strategy.

# Chapter 5

# Evaluations

It's very important to evaluate our work after the development. This chapter, we will evaluate our system in two aspects: data quality and system performance. Both aspect are critical to our dataset, since: without high data quality, our dataset cannot not trust by users; without high performance our approach will not be accepted by other developers and is not queryable by front-end users.

## 5.1 Evaluation, the rationale

As discussed in Chapter 2, we need to evaluate the quality of the data we extracted. [27] suggest a number of quality metrics, in this project, we will use Data Completeness, Data Accuracy, User perceived data quality and Conformance to Expectation (Data fitness) to evaluate our work. The reason for choosing them is because:

**Data Completeness** can reflect how complete the LinkedIn profile is. It's an important statistics we can get from LinkedIn.com as people always interest in how complete for these profiles in general. It also a hint for future research since we can quickly identify sparse data fields and intensive data fields.

**Data Accuracy** is where we introduce volunteers to manually extract the data out from the HTML files and ask them to rate our results from 1 to 5. By calculating prediction precisions, recalls, f-measures (will be discussed later) and user average ratings, we can know how well our parser and our data normalisation is.

**User Perceived Data Quality** is simplified version of Data Accuracy. We present our results and show the user extracted results, then ask user rate our result from 1 to 5. As precision, recall and f-measure are too complex for people who do not have mathematics background, simple ratings can be more user friendly.

**Data fitness** is to measure how well our knowledge model and our dataset match the requirements of the upper layer user interface. In this part, we collect feedbacks from the developer of the Data Visualisation project, and the drawbacks will be presented in our future works.

### 5.1.1 Data completeness

In this section, we present the completeness of profiles in LinkedIn.com. Researchers who also interested in LinkedIn.com public profiles can use this statistics as a measure, to avoid the fields that are too sparse.

#### Definition

[27] defines Data completeness as: A degree of metadata contains all information required to have ideal presentation. To get it we can simple count the number of fields that contains data and divided by total number of instances:

$$C = \frac{\sum_{i=1}^{N} F(i)}{N} \tag{5.1}$$

In Equation 5.1, F(i) is 1 when the field has data and 0 when the field is empty. N is the total number of instances. Notice that the definition of total number of instances can be changed in later paragraph.

Table 5.1 shows the total number of personal profiles, company profiles and total number of skills in all of the profiles. These numbers are base numbers that will be used to calculate the percentage in the following tables.

| | |
|---|---|
| Total number of public personal profiles | 13014 |
| Total number of company profiles | 24778 |
| Total number of skills | 15917 |

Table 5.1: Total number of personal profiles, company profiles and skills

#### Public personal profile completeness

Many people do not fill their complete work experiences and education backgrounds into LinkedIn, therefore, in our 13014 randomly download and selected profiles, we can have an overview of the percentage of people have sections that are missing.

| | Number | Percentage(of personal profiles) |
|---|---|---|
| Profiles that have work experiences | 11501 | 88.4% |
| Profiles that have education | 9913 | 77% |
| Profiles that have skills | 10511 | 80% |
| Profiles that have city information | 10158 | 78% |
| Profiles that have academic degree information | 5230 | 40.2% |
| Profiles that have college major information | 7825 | 60.1% |

Table 5.2: Personal profile completeness

In Table 5.2, the N in Equation 5.1 is the total number of public profiles, and F(i) is each field, i.e. work experience, education, skill, city, degree and major. Notice that the percentage of profiles that have degree and major information are relatively low, it implies that people usually skip fill in their information into their profile. The degree information is the lowest, that is because our Lucene text search engine does not accept any string that cannot be classified, in which case the normalisation result simply return empty string. Then our RDF converter just skip this triple. In order to increase the percentage, we need to have more degree abbreviations and full names to cover every possible degree in University worldwide.

### Company profile completeness

Not every company will register in LinkedIn company to have a company profile. If a company in a person's work experience registered on LinkedIn.com, there's a hyperlink that link the company name to the complete company profile. If the company is not registered, there will be not such hyperlink. So we can easily draw a conclusion from Table 5.3, around 46% of companies in Ireland register in LinkedIn.com.

| | Number | Percentage(of company profiles) |
|---|---|---|
| Company profiles that have industry type | 11868 | 47.9% |
| Company profiles that have organisation type | 11351 | 45.8% |
| Company profiles that have company size info | 11343 | 45.8% |

Table 5.3: Company profile completeness

At here, the N in Equation 5.1 is the total number of companies in our dataset. F(i) is each fields, i.e. industry type, organisation type and the size of the company.

### Data linkage

The definition of RDF data linkage is: average_linkage=$\frac{total\_number\_of\_links}{total\_number\_of\_objects}$. It's a measurement of how "sparse" of the RDF data is. Generally, high linkage means high corre-

lation between objects.

| Total number of objects | 160251 |
|---|---|
| Total number of links | 415916 |
| Average linkage | 2.595 |

Table 5.4: Data Linkage: Total and Average

As we can see in Table 5.4, in average, every object has 2.5 number of links to other objects.

### 5.1.2 Data Accuracy (extracted metadata quality)

#### Evaluation setup

We recruited 10 users, divided them into 5 groups, so each group have 2 participants. Each group of users will view same 10 randomly selected profiles. They were asked to manually extract city, work experiences (including company names, job titles, job start dates and job end dates) and education backgrounds (including college names, majors, degrees, college start dates, college end dates).

Then after the user fill in the data, we display what they entered as well as what we automatically extracted data, and ask them to rate our results, from 1 to 5, where score 5 is highest (the result of this user perceived data quality will be discussed in the next section).

Figure 5.1 is the screenshot for the user interface that asking user to transfer data from LinkedIn profile to our evaluation system:

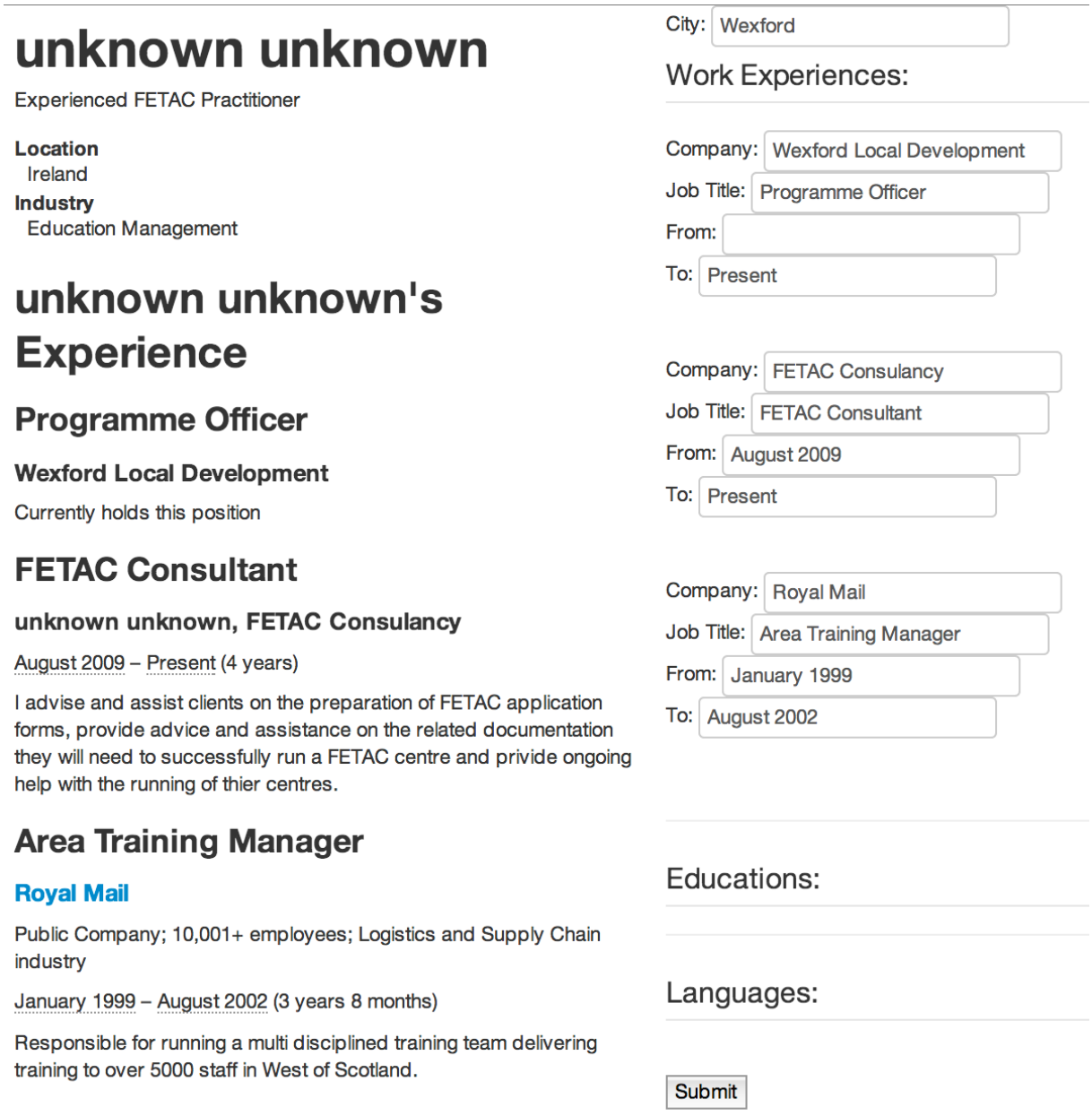


Figure 5.1: Online user evaluation website: User manually extracted data

### Results

After the user evaluation, we take user input as the ground truth, we use string matching to compare entered data with automatic extracted data. If the string matching return false, we manually examine the data and decide whether the extracted data is correct.

The metrics here we use are precision, recall and f-measure.

$$precision = \frac{correctly\_predicted}{predicted} \tag{5.2}$$

$$recall = \frac{correctly\_predicted}{total} \tag{5.3}$$

$$f - measure = \frac{2 * precision * recall}{precision + recall} \tag{5.4}$$

The meaning of these metrics can be explained as follows[34]: Precision, or confidence, is focus on how good we are predicting; Recall, or sensitivity is a measure of the proportion we correctly predicted over total data size. F-measure, or F-score is designed to capture both precision and recall. In order to get high F-score, precision and recall must be high.

| User | Precision | Recall | F-Measure |
|---|---|---|---|
| User 1 | 0.8889 | 0.8889 | 0.8889 |
| User 2 | 0.75 | 0.6667 | 0.7059 |
| User 3 | 0.8571 | 0.6667 | 0.75 |
| User 4 | 0.8571 | 0.6667 | 0.75 |
| User 5 | 0.8889 | 0.8889 | 0.8889 |
| User 6 | 0.8889 | 0.8889 | 0.8889 |
| User 7 | 1 | 1 | 1 |
| User 8 | 1 | 1 | 1 |
| User 9 | 1 | 0.7778 | 0.875 |
| User 10 | 0.875 | 0.7778 | 0.8235 |
| Average | 0.9006 | 0.8222 | 0.8571 |

Table 5.5: Precision, recall and f-measure scores for city information

According to Table 5.5, users are quite satisfy with our city information extraction strategy. Even some lazy volunteers didn't fill in the ground truth, we still getting average of 0.85 F-score. It's acceptable for us to do complex query using person's city information.

| User | Precision | Recall | F-Measure |
|---|---|---|---|
| User 1 | 0.8947 | 0.8947 | 0.8947 |
| User 2 | 0.9737 | 0.9737 | 0.9737 |
| User 3 | 0.96 | 0.96 | 0.96 |
| User 4 | 0.84 | 0.84 | 0.84 |
| User 5 | 1 | 1 | 1 |
| User 6 | 0.9556 | 0.9556 | 0.9556 |
| User 7 | 1 | 1 | 1 |
| User 8 | 1 | 1 | 1 |
| User 9 | 0.9333 | 0.9333 | 0.9333 |
| User 10 | 0.9556 | 0.9556 | 0.9556 |
| Average | 0.9513 | 0.9513 | 0.9513 |

Table 5.6: Precision, recall and f-measure scores for company information

We are getting very high score in company name field according to Table 5.6. The reason for that is because participants normally copy and paste the company name to fill in our survey forms, the exact matching is very high.

| User | Precision | Recall | F-Measure |
|---|---|---|---|
| User 1 | 0.8947 | 0.8947 | 0.8947 |
| User 2 | 0.9474 | 0.9474 | 0.9474 |
| User 3 | 0.96 | 0.96 | 0.96 |
| User 4 | 0.92 | 0.92 | 0.92 |
| User 5 | 0.9333 | 0.9333 | 0.9333 |
| User 6 | 0.8667 | 0.8667 | 0.8667 |
| User 7 | 0.9545 | 0.9545 | 0.9545 |
| User 8 | 1 | 1 | 1 |
| User 9 | 0.8889 | 0.8889 | 0.8889 |
| User 10 | 0.9556 | 0.9556 | 0.9556 |
| Average | 0.9321 | 0.9321 | 0.9321 |

Table 5.7: Precision, recall and f-measure scores for job title information

For the job title field in Table 5.7, the result is similar to the company name field. Users normally copy and paste the text without any term generalisation (e.g. change product manager to manager as it's more general), that's what our parser do as well. So the score is very high.

| User | Precision | Recall | F-Measure |
|---|---|---|---|
| User 1 | 1 | 0.8684 | 0.9296 |
| User 2 | 1 | 0.8947 | 0.9444 |
| User 3 | 1 | 0.84 | 0.913 |
| User 4 | 1 | 0.84 | 0.913 |
| User 5 | 1 | 0.8889 | 0.9412 |
| User 6 | 1 | 0.8889 | 0.9412 |
| User 7 | 1 | 0.7727 | 0.8718 |
| User 8 | 1 | 0.7727 | 0.8718 |
| User 9 | 1 | 0.9556 | 0.9773 |
| User 10 | 1 | 0.9556 | 0.9773 |
| Average | 1 | 0.8678 | 0.9281 |

Table 5.8: Precision, recall and f-measure scores for experience start date information

| User | Precision | Recall | F-Measure |
|---|---|---|---|
| User 1 | 1 | 0.8684 | 0.9296 |
| User 2 | 1 | 0.8684 | 0.9296 |
| User 3 | 1 | 0.8 | 0.8889 |
| User 4 | 1 | 0.8 | 0.8889 |
| User 5 | 1 | 0.9556 | 0.9773 |
| User 6 | 1 | 0.9556 | 0.9773 |
| User 7 | 1 | 0.8182 | 0.9 |
| User 8 | 1 | 0.7727 | 0.8718 |
| User 9 | 1 | 0.9778 | 0.9888 |
| User 10 | 1 | 0.9778 | 0.9888 |
| Average | 1 | 0.8794 | 0.9341 |

Table 5.9: Precision, recall and f-measure scores for experience end date information

Table 5.8 and Table 5.9 illustrate our prediction on start date and end date. The precision is high is because LinkedIn always use same datetime pattern to represent the start date and end date. The recall is low is because some profiles do not have the these fields.

The previous four tables (Table 5.6, Table 5.7, Table 5.8 and Table 5.9) illustrate our parsed result for work experiences (company name, job title, job start date and job end date). With the average F-score greater than 0.9, we can accept the parse result. The reason for such high result in both company name and job title is, we didn't perform data normalisation in these two fields. Basically, volunteers copy and paste these information to our survey form, and that's what our parser do as well. Since strings are fully matched, the scores are high.

| User | Precision | Recall | F-Measure |
|---|---|---|---|
| User 1 | 1 | 1 | 1 |
| User 2 | 0.9286 | 0.9286 | 0.9286 |
| User 3 | 1 | 1 | 1 |
| User 4 | 1 | 1 | 1 |
| User 5 | 0.7778 | 0.7778 | 0.7778 |
| User 6 | 0.9444 | 0.9444 | 0.9444 |
| User 7 | 0.8462 | 0.8462 | 0.8462 |
| User 8 | 0.9231 | 0.9231 | 0.9231 |
| User 9 | 0.6538 | 0.6538 | 0.6538 |
| User 10 | 0.9231 | 0.9231 | 0.9231 |
| Average | 0.8997 | 0.8997 | 0.8997 |

Table 5.10: Precision, recall and f-measure scores for college information

We are getting high score for college name field(Table 5.10, the explanation for high score is the same as company name field, users just copy and paste the college name from the profile. However, one difference is that we also use our data normalisation tool to classify college name to our ground truth college name. Because in our implementation of college name normalisation, if we couldn't find any similar string, we create a new entry in our search engine database and assume it's a new college name.

| **User** | **Precision** | **Recall** | **F-Measure** |
|---|---|---|---|
| User 1 | 0.8571 | 0.8571 | 0.8571 |
| User 2 | 0.8571 | 0.8571 | 0.8571 |
| User 3 | 0.5714 | 0.5 | 0.5333 |
| User 4 | 0.4286 | 0.375 | 0.4 |
| User 5 | 0.8571 | 0.6667 | 0.75 |
| User 6 | 0.8571 | 0.6667 | 0.75 |
| User 7 | 0.6667 | 0.6154 | 0.64 |
| User 8 | 0.75 | 0.6923 | 0.72 |
| User 9 | 0.7083 | 0.6538 | 0.68 |
| User 10 | 0.4 | 0.2308 | 0.2927 |
| Average | 0.6954 | 0.6115 | 0.648 |

Table 5.11: Precision, recall and f-measure scores for major information

According to Table 5.11, our scores for major field are very low. If we look at the low scores and high scores carefully, we can see that they come in a pair. That means the user input data is consistent, and in some groups of profiles, our data normalisation fail to normalise degree information correctly. The reason for that is people do some data cleaning in their minds so the major information they fill in is cleaned and well known. But our system didn't do any language processing, so the result of a simple copy and paste approach is different from the result generated by human mind.

| User | Precision | Recall | F-Measure |
|---|---|---|---|
| User 1 | 1 | 1 | 1 |
| User 2 | 1 | 1 | 1 |
| User 3 | 1 | 0.875 | 0.9333 |
| User 4 | 1 | 0.75 | 0.8571 |
| User 5 | 0.9444 | 0.9444 | 0.9444 |
| User 6 | 1 | 0.8889 | 0.9412 |
| User 7 | 0.9167 | 0.8462 | 0.88 |
| User 8 | 0.8333 | 0.7692 | 0.8 |
| User 9 | 1 | 0.8846 | 0.9388 |
| User 10 | 1 | 0.9615 | 0.9804 |
| Average | 0.9694 | 0.892 | 0.9275 |

Table 5.12: Precision, recall and f-measure scores for degree information

We are getting very high degree score in Table 5.12, which means our ground truth degrees in working properly and classified most of the degree information correctly.

| User | Precision | Recall | F-Measure |
|---|---|---|---|
| User 1 | 0 | 0 | 0 |
| User 2 | 0 | 0 | 0 |
| User 3 | 1 | 0.125 | 0.2222 |
| User 4 | 1 | 0.125 | 0.2222 |
| User 5 | 0 | 0 | 0 |
| User 6 | 0 | 0 | 0 |
| User 7 | 1 | 0.3077 | 0.4706 |
| User 8 | 1 | 0.2308 | 0.375 |
| User 9 | 1 | 0.5 | 0.6667 |
| User 10 | 1 | 0.7308 | 0.8444 |
| Average | 0.6 | 0.2019 | 0.2801 |

Table 5.13: Precision, recall and f-measure scores for education start date information

| User | Precision | Recall | F-Measure |
|---|---|---|---|
| User 1 | 0 | 0 | 0 |
| User 2 | 0 | 0 | 0 |
| User 3 | 1 | 0.125 | 0.2222 |
| User 4 | 1 | 0.25 | 0.4 |
| User 5 | 0 | 0 | 0 |
| User 6 | 0 | 0 | 0 |
| User 7 | 1 | 0.3077 | 0.4706 |
| User 8 | 1 | 0.2308 | 0.375 |
| User 9 | 1 | 0.5 | 0.6667 |
| User 10 | 1 | 0.7308 | 0.8444 |
| Average | 0.6 | 0.2144 | 0.2979 |

Table 5.14: Precision, recall and f-measure scores for education end date information

The scores for college start date (Table 5.13) and college end date (Table 5.14) didn't perform well. This evaluation helps us discover a problem in extracting these two fields. Since our system makes a wrong assumption about the format of the start date and end date(as 'yyyy-mm-dd'), we could not capture the fact that the start date and end date format in education background is 'yyyy'. This finding also prove that doing user study is very important in system evaluation.

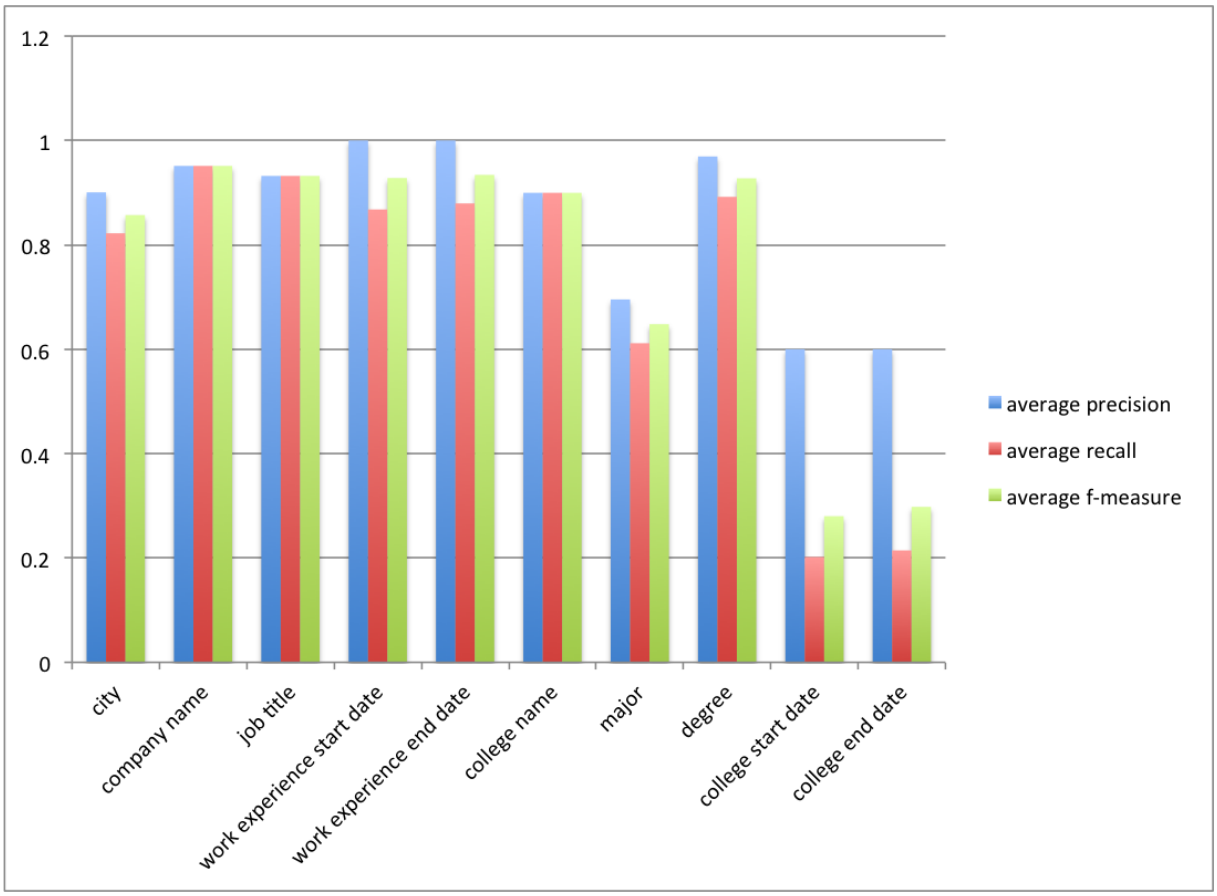


Figure 5.2: Summary: average precion, recall, f-measure for all fields

From Figure 5.2, we can see that city, company names, job titles, work experience start and end date, college names and degree information have high average scores. But for major, we got an average f-score a bit greate than 0.6, which means we cannot correctly classify the major names. In the future, we might need natural language processing and college subjects database to get a better results. And we need to fix the problem in extracting college start date and college end date.

### 5.1.3 User perceived data quality

Apart from precisions, recalls and F-measure, we also collect user ratings for each field and user overall rating for each profile. Figure 5.3 is the screenshot that asking user to compare their manually extracted data with our automatically extracted data and rating our results.

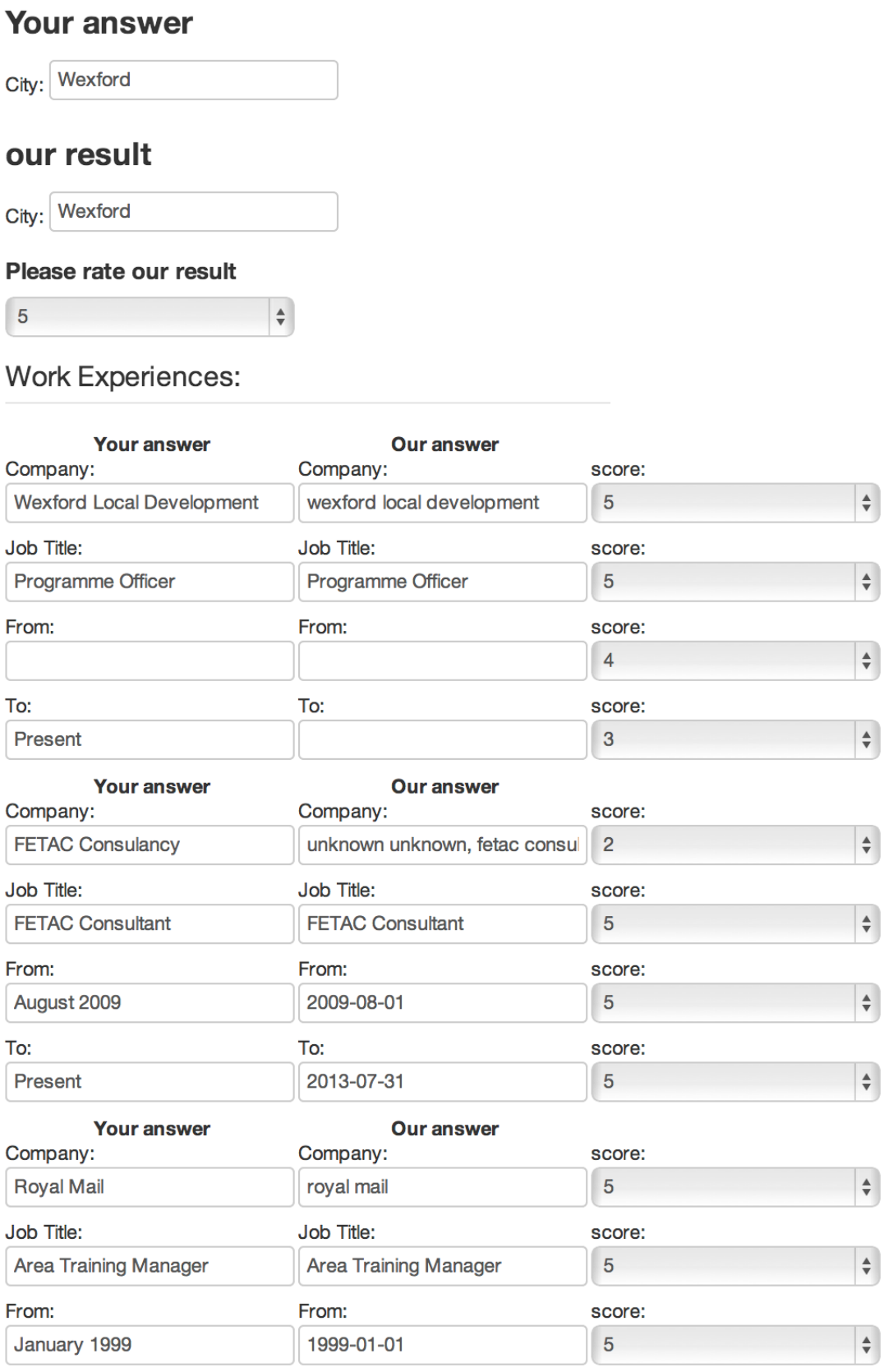


Figure 5.3: Online user evaluation website: Asking user compare manually extracted data with automatically extracted data

At here we only show average rating for each field. From these ratings, we want to get consistent results about how well is our parser and data normalisation working as in Data Accuracy (extracted metadata quality).

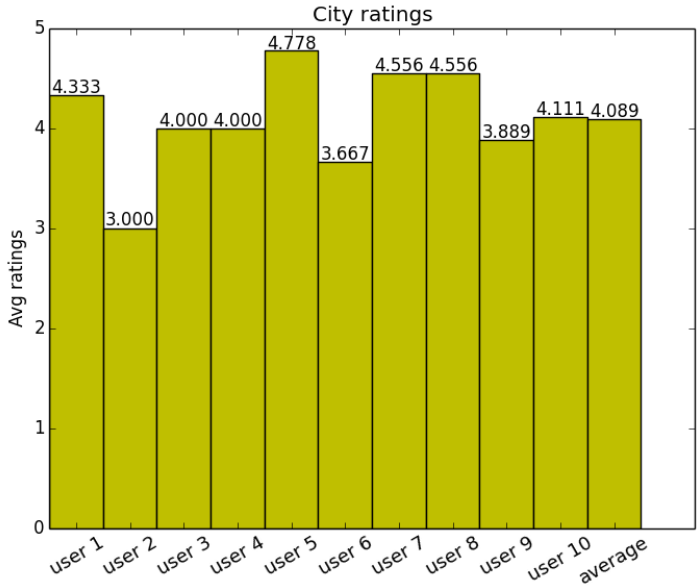


Figure 5.4: User average rating for city

In Figure 5.4 we find the user satisfaction ratings are not fully match with Table5.5. The reason for that is our city extraction strategy is setting the first return possible city from all cities as the person's current living city. Since the profile sometime contains too less information for us to guess where the person is actually located, some users simply don't like our result when they see "A person work for Tesco Ireland is now living in Limerick" as they think there's too few information to predict the correct city.

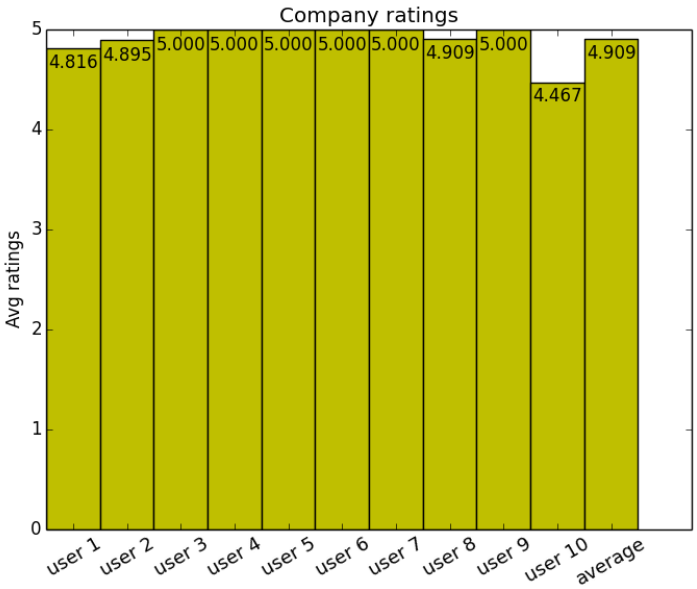


Figure 5.5: User average rating for company name

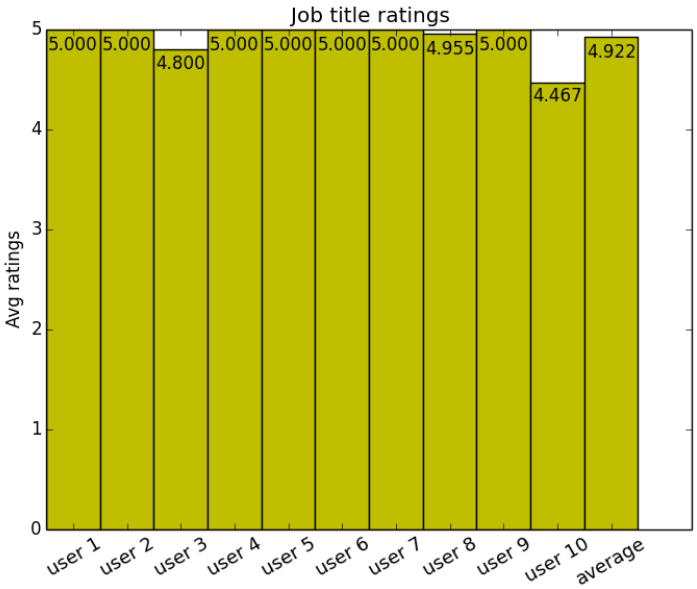


Figure 5.6: User average rating for job title

In Figure 5.5 and Figure 5.6, We get consistent result with Table 5.6 and Table 5.7. As we discussed earlier, a simple copy and paste approach is accepted by our participants.

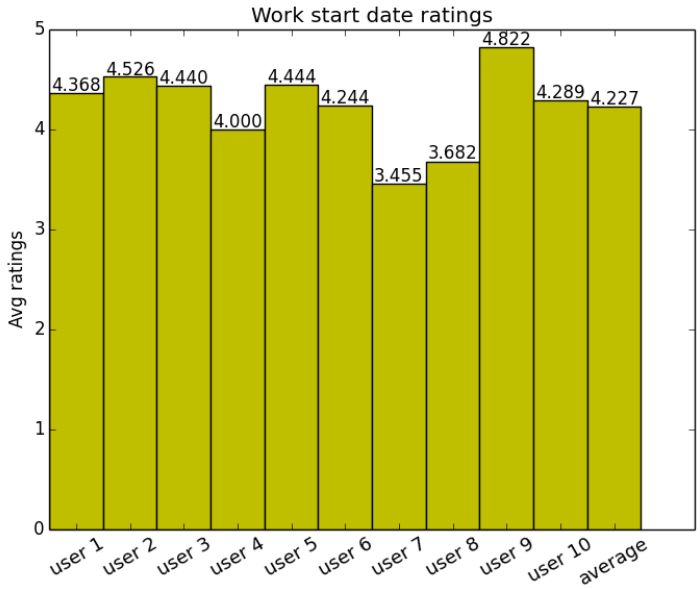


Figure 5.7: User average rating for experience start date

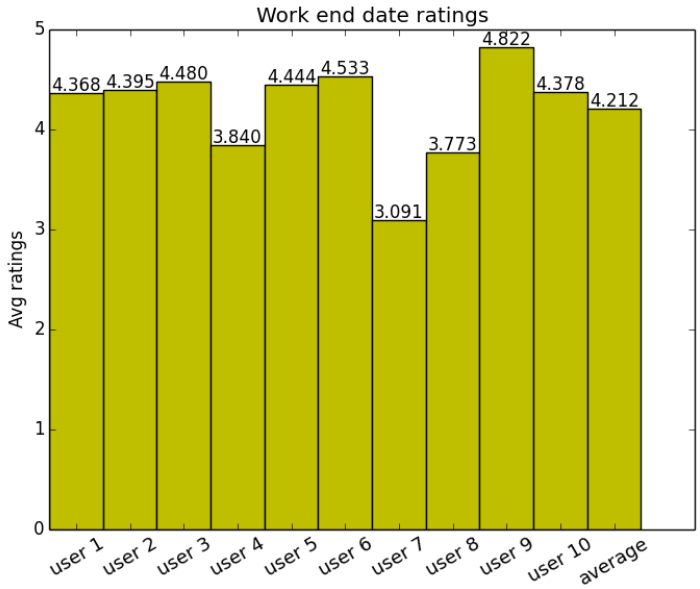


Figure 5.8: User average rating for experience end date

For the start date ratings (Figure 5.7) and end date ratings (Figure 5.8), we are getting lower user satisfaction comparing to what we get at the previous section (Table 5.8 and Table 5.9). The reason for that is, some volunteers do not happy we convert the datetime format from 'MM yyyy' to 'yyyy-mm-01'. What we are doing here is we assume all the start date and end date of a work experience is on the first day of the month. We explain

the reason to volunteers during the evaluation, as we need this format the match the date literal definition in XML Schema[35].

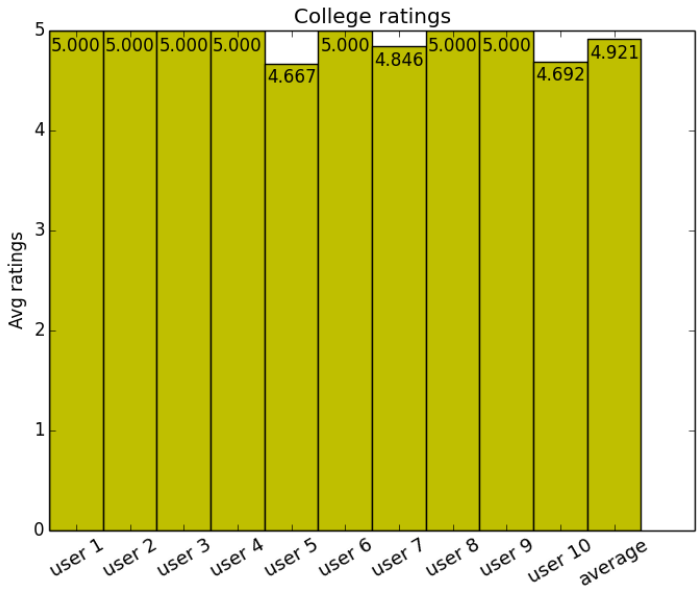


Figure 5.9: User average rating for college

Similar to Figure 5.5, our user average rating for college names (Figure 5.9) is widely accepted. Even we try to use our data normalisation module to classify the college name, but when we encounter a new unknown college name, we add it to our Lucene text search engine database instead of leaving the college name empty.

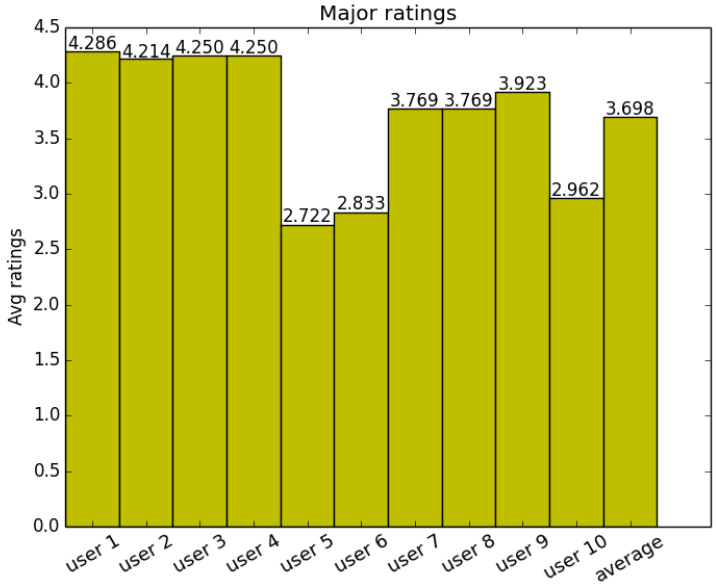


Figure 5.10: User average rating for major

Because every two users are viewing same 10 randomly selected profiles, therefore, for major field, the average ratings (Figure 5.10) from user 5 and user 6 are so low is because our parser perform bad in that 10 profiles. Notice that the user ratings for major field is actually higher than the scores we get in Table5.11, that is because participants are happy to know that our system is not that intelligent as human mind and fail to clean the major field correctly. So participants rate their satisfaction scores high.

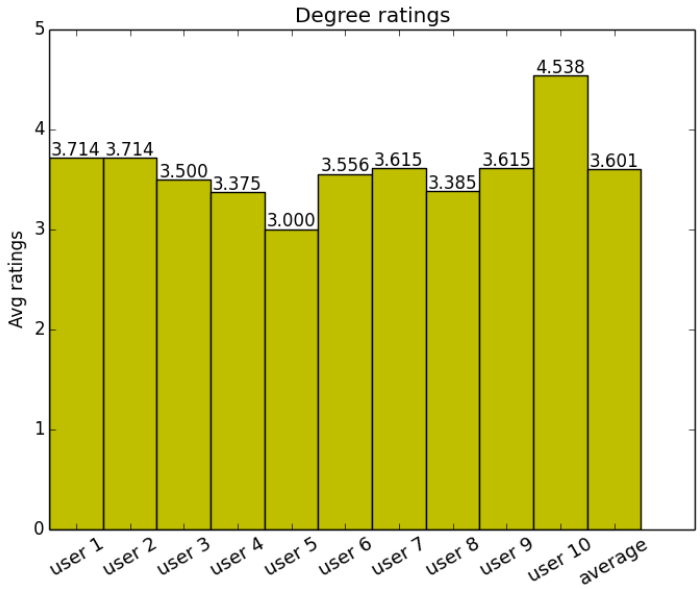


Figure 5.11: User average rating for degree

We are getting lower user ratings at degree fields (Figure 5.11) comparing to 5.12. That's because the data completeness for degree field is low. For every profile that has no degree field, we set the score to 3, which is the average score by default.

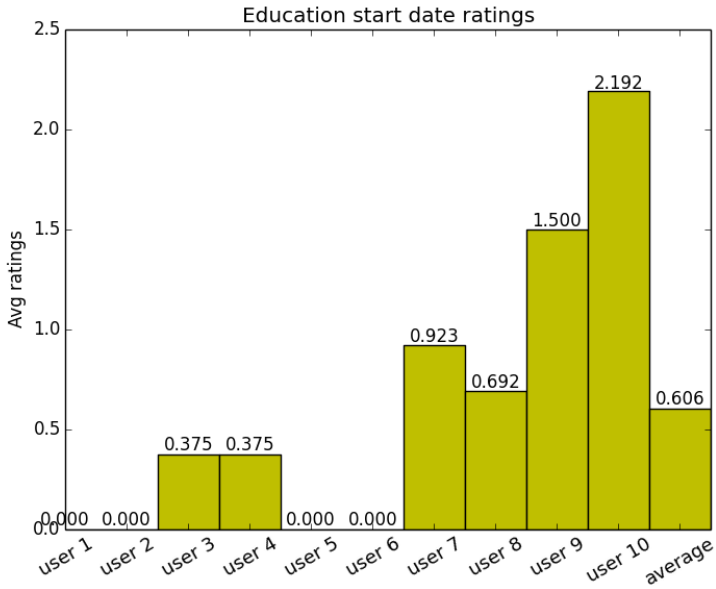


Figure 5.12: User average rating for education start date

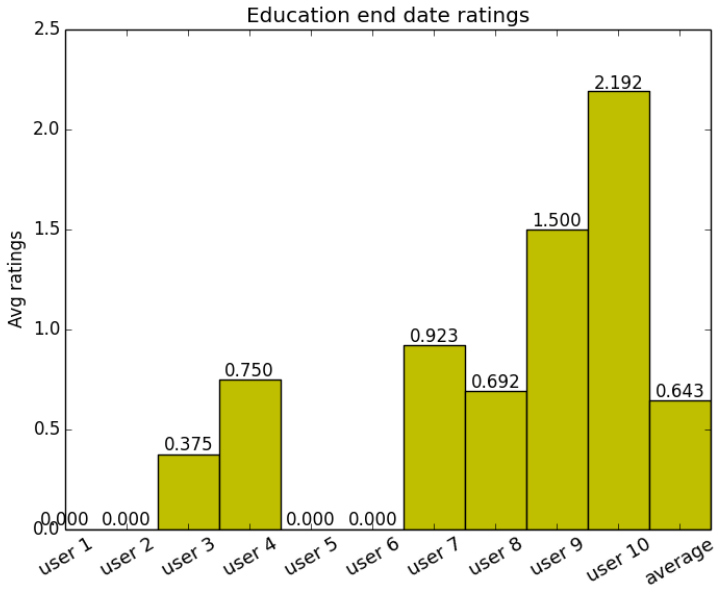


Figure 5.13: User average rating for education end date

As mentioned in the previous section, our parser failed to capture the fact that education start date and education end date is using 'yyyy' pattern in representing the datetime. Therefore, most of our volunteers just set the rating to 0 since there's no information available (Figure 5.12 and Figure 5.13).

Figure 5.14 shows average user ratings for all fields. As we can see, users are quite satisfy with our results in city, company name, job title, work experience start date and

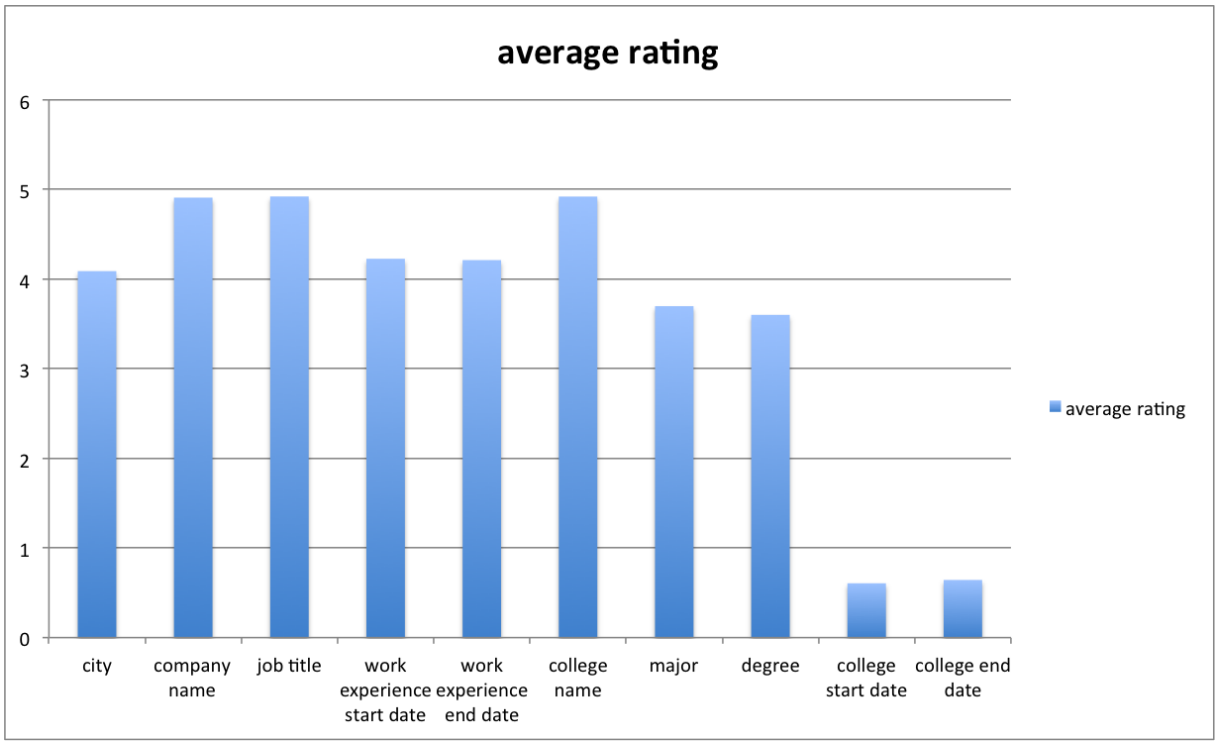


Figure 5.14: Summary: average user ratings for all fields

end date. And users are slightly not happy with our result in major and degree, which is what we need to improve in the future. Finally users are totally not accept the empty college start date and end date.

### 5.1.4 Metadata fitness

This section is a reflection on how the extracted triples fit to the data visualisation interface. The interface is divided into 3 scenarios:

Scenario 1, Government
: The first scenario is that the interface should reflect the needs of Government officers. The fields that used by them are: city, industry type, academic degree and company size.

Scenario 2, Company's human resource department
: The user interface also support daily queries from HR. The fields that used in this scenario are: city, degree, skill, work experience and start date.

Scenario 3, Job seekers and college students This part allows users get insights about the employment status by city. The fields that used in this scenario are: city, degree, skill and position.

During the development and evaluation, we found some drawbacks of the extracted data that makes the query and the user interface hard to develop and use:

1. Do not have information to group similar job title together. This is important as we want our query return more correct result and in the meantime keeps the query as simple as possible. For example, when one querying "software engineer" similar terms such as "application developer, Java software engineer" should all return as these job titles have no significant difference if we want to compare this job in IT over another industry. Doing this classification is hard, we will discuss it in future work section in the next chapter.

2. Company names may have aliases. One example is "Oracle" and "Oracle EMEA". But since we cannot find company names database as ground truth, our system cannot handle this problem. This issue is very similar to the previous one, it needs we have very accurate ground truth.

3. As SPARQL has very limit function in datetime manipulation, the start date and end date approach in our model is not working really well. For example, if one user is looking for "How many people had been working in a company for more then 5 years?". The query is very hard to write so it would be easier if our model have "year between" field that address this requirement.

The details of possible solution of drawbacks will be discussed in next chapter, future works section.

## 5.2 System performance

In System environment, we list our software and hardware details. Here we want to show the performance of some critical modules, to provide more comprehensive details of the system. One important things to know is that the performance measurements does not include database accessing and file serialisation and other miscellaneous, therefore, in production environment, the system performance could be less than the result we got.

### 5.2.1 Parsing performance

We run our parser 10 times, each time it parses 100 randomly selected profiles, the average time spending on parsing 100 profiles is: 18.27 seconds.

### 5.2.2 Normalising and converting performance

We run our RDF converter 10 times, each time in try to normalise the data in 100 profiles and convert it into RDF triples, the average time spending on this is: 345.53 seconds.

### 5.2.3 Query performance

We tried several queries with different level of complexity. Notice that the complexity means the complexity of the query structure, it does not mean the complexity of the SPARQL engine in processing it.

**A query that only return subject, predicate and object**

```
select * where {?subject ?predicate ?object. }
```

Listing 5.1: Query1: A query that return all subjects, predicates and objects

The time spend on this query is: 13.179s, and it returned 819488 rows.

**A query that asks from subject, predicate and object then summing up the subject**

```
select (count(?subject) as ?total) where {
       ?subject ?predicate ?object.
}
```

Listing 5.2: Query2: A query that return all subjects, predicates, objects and count subjects

The time spend on this query is: 1.799s, and it returned 1 rows.

Notice that this query (Query 5.2) is 7 times faster that the previous query (Query 5.1) even though the complexity is higher. We don't know the implementation of 4store, but one possible explaination is that the COUNT method can be optimised so the program do not really need read all rows to get the actually result.

### A query that defines two relationships

```
select * where {
        ?person a foaf:Person;
                lk:skill ?skill.
}
```

Listing 5.3: Query3: A query that defines two relationships

The time spend on this query is: 1.795s, and it returned 196187 rows.

It is an interesting result because getting graph patterns (Query 5.3) is actually quicker than the first query, which only ask for all triples. One possible answer is the 4store SPARQL engine has special optimisation on graph matching.

### A query that defines two relationships with group by and count

```
select ?city (count(?person) as ?pCount) where {
        ?person a foaf:Person ;
                dbpedia-owl:city ?city .
} group by ?city
```

Listing 5.4: Query4: A query that defines two relationships and use group by and count

The time spend on this query is: 0.353s, and it returned 19 rows.

This time, data aggregation query (Query 5.4 is actually quicker than simple "print all" query (Query 5.1) and query with simple graph matching (Query 5.3). This result demonstrates 4store implementation of data aggregation has very high performance.

### A query that defines two relationships with group by, count and order by

```
select ?skill (count(?skill) as ?sCount) where{
        ?p a foaf:Person;
                lk:skill ?skill.
} group by ?skill order by desc(?sCount)
```

Listing 5.5: Query5: A query that defines two relationships and use group by, count and order by

The time spend on this query is: 2.204s, and it returned 16185 rows.

This query (Query 5.5) takes more than 2 seconds to execute. That is because to generate the correct result, the SPARQL engine has to split all the skill into groups, sum them up and finally sort the results.

The performance seems is reasonable if we are not trying get all raw triples from the server. However, the performance can be even better if we have a better hardware (our production environment is Amazon EC2 64 bit Ubuntu12.04, Intel Xeon Central processing unit E5-2650 2.00GHZ, 4G memory). We don't have enough budget to get a high performance CPU instance on Amazon EC2. Figure 5.15 shows a comparison on query performance between our production server and our development server. The parameters of our development machine are: MacBook Pro OSX 10.8 Mountain Lion, Intel core i7 2.7GHz CPU, 8GB 1600MHz DDR3 memory.

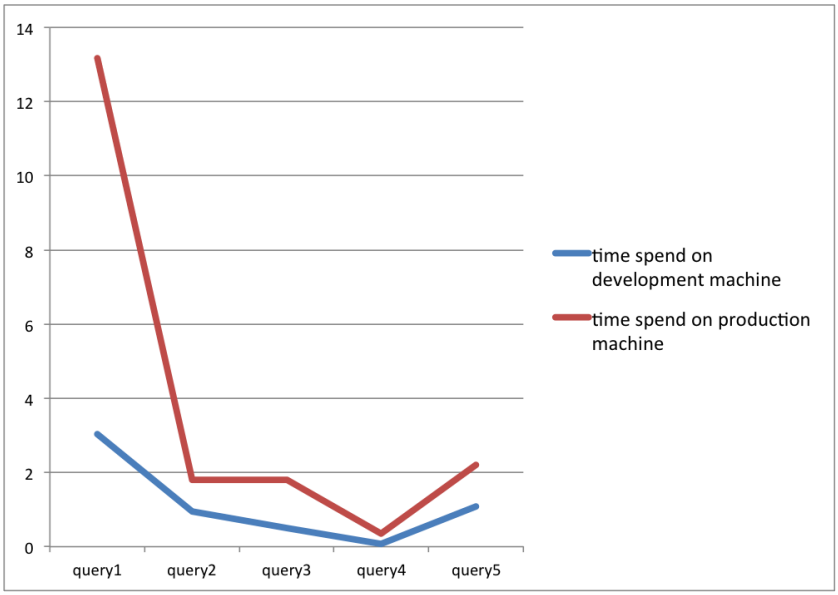


Figure 5.15: Query performance of all 5 queries on development machine and production machine

Notice that if we run our query on our development machine, the time spend on getting all triples is around 3 second, which is 4 times faster than our production server.

### Conclusion

We can see that for complex group by, count and order by query, our production server takes about 2 seconds to run. One important factor that result in slow queries is the system hardware. Because we test the performance in production environment, the machine is an Amazon EC2 m1.medium instance, which is relatively low comparing to current standard server. We also illustrate the performance issue can be potentially alleviated by upgrading to a higher Central processing unit. With these query times, our server should be able to respond to public queries.

## 5.3 Chapter Summary

In this section, we evaluated our result set and SPARQL server thoroughly. We performed evaluations on Data completeness, Data accuracy, User ratings and System performance. We found the fields that people normally missing in their profiles, the fields that are converted with high accuracy and also identify issues that need to be fixed or improved. We examined the performance of our production server and make suggestions for future improvement.

# Chapter 6

# Conclusions and Future Work

## 6.1 Conclusions

Generally speaking, UGC, is inherently unstructured but informative. Semantic Web is one of the best technologies that tries to solve this problem as it doesn't make any assumption about the form of the data. High flexibility can be guaranteed in creating and maintaining the generated knowledge graph.

In this case study, we take LinkedIn.com public profiles as research subject, investigate the possibility of converting semi-structure profiles to a RDF dataset. We create a knowledge graph that can express the semantics behind the profiles and the relations between nodes. Our project and the upper layer user interface (the project "leveraging power of social media and data visualisation") has proved that Semantic Web technologies can be a good solution to represent unstructured data. In addition, we provide an simple way to download a large amount of LinkedIn.com public profiles and extract the user's current living city information from the raw HTML files. Finally, we contribute a public SPARQL endpoint that allow everyone query about facts in LinkedIn Ireland.

## 6.2 Future work

### 6.2.1 Company names and job tiles classification

In chapter 5 we discuss the data fitness of our generated triples. The major problem is that the company names and job titles do not have semantics. One example would be: if we query about "Microsoft" there's no hint about "Microsoft Ireland" is also a valid result for this query. So our query cannot always obtain the correct answers because some possible aliases might not be included in the result set.

One possible solution is to find ground truth datasets that include a large amount of company names and job titles. But this approach is unrealistic because the possible combinations for job title, for example, are infinite. We might look for a machine learning approach, which first extract a large amount of metadata from the profiles we extracted, then create a "game" that ask volunteers manually link the data with same meaning. With this validation dataset, we can easy to apply different machine learning algorithms to see which one is better.

### 6.2.2 Expand the current dataset

At the moment, the public SPARQL endpoint has only 13,000 LinkedIn Ireland personal public profiles. We still want to include most of the profiles in LinkedIn Ireland to get a complete understanding about Irish industry. Downloading and parsing profiles at such large scale is nearly impossible without LinkedIn's help, if possible, we hope we can work with LinkedIn Data Engineering team to develop a knowledge model for LinkedIn.com officially. Fully integrated our work to LinkedIn can be very helpful and interesting.

# Abbreviations

**AI** Artificial Intelligence. 6

**AIISO** Academic Institution Internal Structure Ontology. 13, 20

**API** Application programming interface. 21

**CPU** Central processing unit. 24, 50

**CSS** Cascading Style Sheets. 7

**DBMS** Database Management System. 7

**DOM** Document Object Model. 6

**ETL** Extract, Transform and Load. 6

**FOAF** Friend of A Friend. 20, 21

**HR** Human Resources. 2

**HTML** HyperText Markup Language. 6

**HTTP** Hypertext Transfer Protocol. 10

**IE** Information Extraction. 14

**IR** information retrieval. 8

**IT** Information Technology. 1, 19

**JVM** Java Virtual Machine. 24

**LDIF** Linked Data Integration Framework. 15

**LOD** Linked Open Data. 11

**NTLK** Natural Language Toolkit. 7

**OWL** Web Ontology Languag. 7, 10

**RDF** Resource Description Framework. 5

**RDFS** Resource Description Framework Schema. 10

**SKOS** Simple Knowledge Organization System. 20

**SPARQL** SPARQL Protocol and RDF Query Language. 5

**UGC** User Generated Content. 1, 2, 52

**UI** user interface(s). 7, 17, 18, 25

**URI** Uniform resource identifier. 10

**URL** Uniform resource locator. 21

**XML** Extensible Markup Language. 7

# Bibliography


[1] J. Krumm, N. Davies, and C. Narayanaswami, "User-generated content," *Pervasive Computing, IEEE*, vol. 7, no. 4, pp. 10–11, 2008.

[2] N. Shadbolt, W. Hall, and T. Berners-Lee, "The semantic web revisited," *Intelligent Systems, IEEE*, vol. 21, no. 3, pp. 96–101, Jan.-Feb. ISSN: 1541-1672. DOI: 10.1109/MIS.2006.62.

[3] S. Harris, N. Lamb, and N. Shadbolt, "4store: the design and implementation of a clustered rdf store," in *5th International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS2009)*, 2009, pp. 94–109.

[4] Y. Li, Y. Shi, X. Fan, and M. Bhavsar, *Careergalaxy  a planner for future*, 2012.

[5] C. Bizer, T. Heath, and T. Berners-Lee, "Linked data-the story so far," *International Journal on Semantic Web and Information Systems (IJSWIS)*, vol. 5, no. 3, pp. 1–22, 2009.

[6] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "Dbpedia: a nucleus for a web of open data," *The Semantic Web*, pp. 722–735, 2007.

[7] E. Ferrara, P. De Meo, G. Fiumara, and R. Baumgartner, "Web data extraction, applications and techniques: a survey," *arXiv preprint arXiv:1207.0246*, 2012.

[8] C. Bizer, P. Boncz, M. Brodie, and O. Erling, "The meaningful use of big data: four perspectives–four challenges," *ACM SIGMOD Record*, vol. 40, no. 4, pp. 56–60, 2012.

[9] D. Bradbury, "Data mining with linkedin," *Computer Fraud & Security*, vol. 2011, no. 10, pp. 5 –8, 2011, ISSN: 1361-3723. DOI: 10.1016/S1361-3723(11)70101-4. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1361372311701014.

[10] M. Russell, *Mining the Social Web: Analyzing Data from Facebook, Twitter, LinkedIn, and Other Social Media Sites*. O'Reilly Media, 2011.

[11] W. Gatterbauer, P. Bohunsky, M. Herzog, B. Krüpl, and B. Pollak, "Towards domain-independent information extraction from web tables," in *Proceedings of the 16th international conference on World Wide Web*, ACM, 2007, pp. 71–80.

[12] T. Atapattu, K. Falkner, and N. Falkner, "Automated extraction of semantic concepts from semi-structured data: supporting computer-based education through the analysis of lecture notes," in *Database and Expert Systems Applications*, Springer, 2012, pp. 161–175.

[13] A. Hemnani and S. Bressan, "Extracting information from semi-structured web documents," *Advances in Object-Oriented Information Systems*, pp. 389–396, 2002.

[14] D. Lange, C. Böhm, and F. Naumann, "Extracting structured information from wikipedia articles to populate infoboxes," in *Proceedings of the 19th ACM international conference on Information and knowledge management*, ACM, 2010, pp. 1661–1664.

[15] M. Sah and V. Wade, "Automatic metadata extraction from multilingual enterprise content," in *Proceedings of the 19th ACM international conference on Information and knowledge management*, ser. CIKM '10, Toronto, ON, Canada: ACM, 2010, pp. 1665–1668, ISBN: 978-1-4503-0099-5. DOI: 10.1145/1871437.1871699. [Online]. Available: http://doi.acm.org/10.1145/1871437.1871699.

[16] M. Bergman, "Advantages and myths of rdf," *AI3, April*, 2009.

[17] T. Berners-Lee, *Design issues: linked data*, 2006.

[18] L. Ding, L. Zhou, T. Finin, and A. Joshi, "How the semantic web is being used: an analysis of foaf documents," in *Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05) - Track 4 - Volume 04*, ser. HICSS '05, IEEE Computer Society, 2005, pp. 113.3–, ISBN: 0-7695-2268-8-4. DOI: 10.1109/HICSS.2005.299. [Online]. Available: http://dx.doi.org/10.1109/HICSS.2005.299.

[19] T. Heath and C. Bizer, "Linked data: evolving the web into a global data space," *Synthesis lectures on the semantic web: theory and technology*, vol. 1, no. 1, pp. 1–136, 2011.

[20] S. Hellmann, C. Stadler, J. Lehmann, and S. Auer, "Dbpedia live extraction," in *Proceedings of the Confederated International Conferences, CoopIS, DOA, IS, and ODBASE 2009 on On the Move to Meaningful Internet Systems: Part II*, ser. OTM '09, Vilamoura, Portugal: Springer-Verlag, 2009, pp. 1209–1223, ISBN: 978-3-642-05150-0. DOI: 10.1007/978-3-642-05151-7_33. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-05151-7_33.

[21] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer, "Dbpedia spotlight: shedding light on the web of documents," in *Proceedings of the 7th International Conference on Semantic Systems*, ser. I-Semantics '11, Graz, Austria: ACM, 2011, pp. 1–8, ISBN: 978-1-4503-0621-8. DOI: 10.1145/2063518.2063519. [Online]. Available: http://doi.acm.org/10.1145/2063518.2063519.

[22] P. Mika, "Ontologies are us: a unified model of social networks and semantics," *Web Semant.*, vol. 5, no. 1, pp. 5–15, Mar. 2007, ISSN: 1570-8268. DOI: 10.1016/j.websem.2006.11.002. [Online]. Available: http://dx.doi.org/10.1016/j.websem.2006.11.002.

[23] S. Wang, Y. Zeng, and N. Zhong, "Ontology extraction and integration from semi-structured data," *Active Media Technology*, pp. 39–48, 2011.

[24] F. Sebastiani, "Machine learning in automated text categorization," *ACM Comput. Surv.*, vol. 34, no. 1, pp. 1–47, Mar. 2002, ISSN: 0360-0300. DOI: 10.1145/505282.505283. [Online]. Available: http://doi.acm.org/10.1145/505282.505283.

[25] S. godbole, I. Bhattacharya, A. Gupta, and A. Verma, "Building re-usable dictionary repositories for real-world text mining," in *Proceedings of the 19th ACM international conference on Information and knowledge management*, ser. CIKM '10, Toronto, ON, Canada: ACM, 2010, pp. 1189–1198, ISBN: 978-1-4503-0099-5. DOI: 10.1145/1871437.1871588. [Online]. Available: http://doi.acm.org/10.1145/1871437.1871588.

[26] C. Bohm, F. Naumann, M. Freitag, S. George, N. Hofler, M. Koppelmann, C. Lehmann, A. Mascher, and T. Schmidt, "Linking open government data: what journalists wish they had known," in *Proceedings of the 6th International Conference on Semantic Systems*, ACM, 2010, p. 34.

[27] X. Ochoa and E. Duval, "Quality metrics for learning object metadata," in *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2006*, E. Pearson and P. Bohman, Eds., Chesapeake, VA: AACE, 2006, pp. 1004–1011. [Online]. Available: http://www.editlib.org/p/23127.

[28] P. Mendes, H. Mühleisen, and C. Bizer, "Sieve: linked data quality assessment and fusion," in *Proceedings of the 2012 Joint EDBT/ICDT Workshops*, ACM, 2012, pp. 116–123.

[29] E. Simperl, "Reusing ontologies on the semantic web: a feasibility study," *Data & Knowledge Engineering*, vol. 68, no. 10, pp. 905 –925, 2009, ISSN: 0169-023X. DOI: http://dx.doi.org/10.1016/j.datak.2009.02.002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0169023X0900007X.

[30] A. Miles and J. R. Pérez-Agüera, "Skos: simple knowledge organisation for the web," *Cataloging & Classification Quarterly*, vol. 43, no. 3-4, pp. 69–83, 2007.

[31] J. Golbeck and M. Rothstein, "Linking social networks on the web with foaf: a semantic web case study.," in *AAAI*, vol. 8, 2008, pp. 1138–1143.

[32] M. McCandless, E. Hatcher, and O. Gospodnetic, *Lucene in Action, Second Edition: Covers Apache Lucene 3.0*. Greenwich, CT, USA: Manning Publications Co., 2010, ISBN: 1933988177, 9781933988177.

[33] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*, 2nd. Cambridge, MA, USA: MIT Press, 2001, ISBN: 0-262-03293-7, 9780262032933.

[34] D. Powers, "Evaluation: from precision, recall and f-measure to roc., informedness, markedness & correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011.

[35] P. Biron, A. Malhotra, W. W. W. Consortium, *et al.*, "Xml schema part 2: datatypes," *World Wide Web Consortium Recommendation REC-xmlschema-2-20041028*, 2004.