UNIVERSITY OF BERGEN

MASTER THESIS

AUTOMATIC QUIZ GENERATION FOR ELDERLY PEOPLE

Author: Jeanette Samuelsen
Supervisor: Weiqin Chen

in the
Department of Information Science and Media Studies

June 1, 2016

Abstract

Studies have indicated that games can be beneficial for the elderly, in areas such as cognitive functioning and well-being. Taking part in social activities, such as playing a game with others, could also be beneficial. One type of game is a computer-based quiz. One can create quiz questions manually; however, this can be time-consuming. Another approach is to generate quiz questions automatically. This project has examined how quizzes for Norwegian elderly can be automatically generated using online resources. To get a better understanding of the topic, a multiple choice quiz application (prototype) has been developed, which automatically generates questions and answers. The target users for the quiz are elderly people above the age of 60. The quiz can ask factual questions about people (actors and musicians) that were well-known and works (movies and literature) that were created during its target users' past. The quiz can be played alone or with another person. Creating the quiz involved retrieving web pages from Wikipedia, using information extraction to extract quiz data from those web pages, and representing and storing the data with Semantic Web technologies. Some of the quiz data was obtained from DBpedia, which makes available (mostly) structured data from Wikipedia for storage. In addition, two approaches were used in this project for identification or extraction of data, which could then be represented and stored to support or enable further question generation. The first approach is to create a Document Object Model tree from relevant web pages, and use extraction rules to extract data from semi-structured HTML elements (lists and tables). The second approach is to detect and classify entities and their relationships in unstructured text. All quiz data was extracted from Wikipedia, but the approaches used by the quiz application and DBpedia all target information from different parts of Wikipedia articles. The quiz generated in this project is in Norwegian, but methods used are generic and it is easy to generate quizzes in other languages. The quiz was evaluated by means of usability testing with participants, to get an indication of its usability according to specific usability measures. The majority of the participants thought the quiz was useful and effective to use, and expressed satisfaction with the quiz. Some potential improvements were also identified during the evaluation.

Keywords: DBpedia, information extraction, question generation

Acknowledgements

I would first and foremost like to thank my supervisor Weiqin Chen. For all of her help, patience and positive encouragement. I am very grateful.

Thank you to my significant other, Erik Gjertsen. For reading and commenting on my drafts, and for being his supportive and accepting self.

Thank you to Kine Lohne de Nijs. For taking time out of her busy schedule to give valuable feedback on this thesis.

Thanks to my grandparents, Laila and Audun Samuelsen. For always believing in me, and for encouraging my interest in computers.

I would also like to thank all the participants who helped to evaluate the quiz.

Table of contents

List of figures

List of tables

Chapter 1 - Introduction

1.1 Background

The amount of elderly people in Norway is increasing. Currently, 14% of the Norwegian population are 67 years or more, while people that are 70 years or more constitute approximately a tenth of the population. Following World War II, there was a baby boom that continued until the middle of the 1960's. The first children born during this post-war period have now reached 70 years (Statistics Norway, 2016). Statistics Norway (2016) have stated that there will be a particular increase in the elderly population as the post-war baby boomers continue to become elderly. In addition, the mortality rate has generally declined for the last 100 years, which is reflected in an increased life expectancy for Norwegians.

Growing old is not necessarily without its challenges. According to the research literature, normal ageing is associated with decline in areas such as sensory (for example hearing and vision), motor, and cognitive[1] abilities (Chen, 2013). Regarding the decline in cognitive abilities, one of the affected abilities is memory. The way memory is affected by normal ageing differs according to the type of memory. Two types of memory are short-term memory and working memory. Short-term memory temporarily stores mental representations of informational or sensory input; the amount of items it can hold at a time is very limited. Items are thought to be stored in different ways, for example as representations of visual or phonetic information (Passer & Smith, 2008). Working memory uses short-term memory as a component process, it can be viewed as a mental workspace where temporarily stored information (items) is actively manipulated. Working memory, which is itself a cognitive function, also provides support for other cognitive functions, for example problem solving (Hedden & Gabrieli, 2004; Passer & Smith, 2008). Studies have found that short-term memory generally declines somewhat throughout adult life, with more distinct declines from the age of 70. Working memory has been found to have a linear life-long decline, with no evidence of distinct declines in later life (Hedden & Gabrieli, 2004).

[1] The meaning of the term cognitive in this thesis is: “of or relating to the mental processes of perception, memory, judgment, and reasoning, as contrasted with emotional and volitional processes” (“cognitivity”, 2016).

In addition to the effects of normal ageing on cognitive abilities, older age is also strongly associated with risk for most brain disorders that can lead to dementia. Dementia refers to "impaired memory and other cognitive deficits that accompany brain degeneration and interfere with normal functioning" (Passer & Smith, 2008, p. 275). For people over 65 years old, Alzheimer's disease is the most common cause for dementia. It is a progressive brain disorder, gradually spreading to different areas of the brain (Passer & Smith, 2008). In Norway, it has often been assumed that approximately 70.000 people, or 1.4% of the population, suffer from dementia. In reality there is no good Norwegian data for dementia, the estimate is based on a study from the Netherlands and is uncertain. Nevertheless, it is expected that dementia will become more prevalent in Norway as life expectancy increases (Norwegian Institute of Public Health, 2015).

Another possible aspect of growing old is that of loss. We can imagine that with older age, loss is a natural consequence, whether it be the loss of a career because of retirement, the loss of a spouse, or the loss of peers (such as a friend or family member). These types of loss can lead to a decreased social network. Age-related health issues, for instance because of reduced vision or motor skills, can lead to reduced mobility, which could result in isolation. There is generally a positive association between age and loneliness (Hansen & Slagsvold, 2015). Hansen and Slagsvold (2015) analyzed survey data from the "Generations and Gender Survey[2]", and found that approximately one tenth of the Norwegian respondents between the age of 71 to 80 years met criteria for self-reported loneliness.

Studies have suggested that loneliness can have adverse effects on health. For example, loneliness has been found to be associated with an increased risk of developing late-life dementia (Wilson, Krueger, et al., 2007); and when coupled with isolation, loneliness has been found to be associated with increased mortality (Steptoe, Shankar, Demakakos & Wardle, 2013). On a related note, Grav, Romild and Stordal (2012) conducted a cross-sectional survey with participants aged 20 to 89 years old, living in Nord-Trøndelag County. They found that perceived social support is significantly associated with depression across different age groups, something that is consistent with findings by researchers in other countries. Perceived social support involves believing that one is loved and cared for, and that one is a part of a network involving mutual reciprocity. Depression can cause a large amount of burden, and is associated with a

[2] http://www.ggp-i.org/

significant loss of quality of life (Grav et al., 2012). These findings indicate that maintaining social networks and taking part in social activities can be beneficial for the health of elderly people.

Several studies have found "higher level of cognitive activity to be associated with reduced cognitive loss, in the form of incident mild cognitive impairment, incident dementia or AD [Alzheimer's disease], or cognitive decline" (Wilson, Scherr, Schneider, Tang & Bennet, 2007). One type of cognitive activity is that of playing games (Wilson, Scherr, et al., 2007). Studies have been done to examine the effects from playing digital games, in areas such as cognitive functioning, reaction time, and well-being (Goldstein et al., 1997). Goldstein et al. (1997) examined the effects of letting non-institutionalized participants aged between 69 to 90 play tetris minimum 5 hours a week for 5 weeks. This study found that participants had significantly improved reaction times after 5 weeks of playing, and that the subjective well-being of the participants was relatively better after 5 weeks of Tetris playing than it was for a study control group that did not play any games. A later study by Basak, Boot, Voss and Kramer (2008) reported that training older participants (mean: 70 years) in a real-time strategy (RTS) game was correlated with improvements in the participants' executive functions. Executive functions are the more high-level cognitive functions; they are used when we direct our behavior in an adaptive manner, for example when we try to accomplish a goal (Passer & Smith, 2008). Among the executive functions that were found to be affected after training in the RTS were working memory (switching between temporarily stored items) and reasoning. Training in the RTS was also correlated with improvements in visual short-term memory (Basak et al., 2008).

The two digital games mentioned above, that were tested with elderly people, were not aimed directly at an elderly population. Chen (2013) designed a suite of tangible tabletop games, made specifically for elderly people. The games could help the elderly to remember, and talk about their past. The games could be played alone or in a group setting. Among the games in the suite were Norwegian quizzes about well-known people and events. Different historical periods were covered by the quizzes. The questions in the quizzes were created manually, something that can be time-consuming.

Another approach to developing quizzes is to generate the questions automatically. This type of automatic question generation is of great interest to this master project. An automatic question generation application is a special type of question answering system. Question answering is a type of information retrieval (IR). The field of IR is concerned with finding materials that satisfy an information need within (computer-based) collections (Manning, Raghavan & Schütze, 2008).

A typical example of an IR application is a Web-based search engine, such as the Google[3] search engine. Here the user enters keywords in accordance with their information need, and the search engine returns materials (such as links to documents) from its collection that are found relevant to the information need. In a question answering system, the user expresses their information need as a natural language statement or question, and the question answering systems returns a specific piece of information (such as a word or sentence) from its collection as the answer.

A relevant field for question answering is natural language processing (NLP). NLP is a field that "aims to convert human language into a formal representation that is easy for computers to manipulate" (Collobert & Weston, 2008). NLP is a vast field which can be said to be working on many tasks (or problems) of varying granularity. The NLP tasks may themselves be divided into sub-tasks. One NLP task that is of importance to this thesis is information extraction (IE). At a more basic level, IE is concerned with recognizing and extracting specific information from documents (Wimalasuriya & Dou, 2010). IE has a variety of sub-tasks, such as Web IE, named entity recognition (NER) and relation detection and classification (see Section 2.2.1). Different techniques have been developed to help solve distinct NLP tasks. The techniques can for instance be based on rules, or employ methods from machine learning[4]. In later times, there has been some convergence between the fields of IR and NLP, for example some of the NLP techniques are also used by the IR community (Nadkarni, Ohno-Machado & Chapman, 2011).

Varying technologies can be used to represent and store data used for automatic quiz generation. A more traditional method would be to represent data as data instances (rows) in the relational model, and store them in a relational database. In the case of an automatic question generation system that involves document search, it might be suitable to represent and store textual data in special-purpose index files (Zeng, Sakai, Yin, Suzuki & Hirokawa, 2013). Another possibility, which is of special interest to this thesis, is to use Semantic Web technologies. The Semantic Web can refer to "the W3C's vision of the web of linked data" (W3C, 2015a). Linked data is about turning the World Wide Web (hereafter referred to as the Web) into a global dataspace, using best practices to publish and connect the data. Different technologies are used for the Semantic Web; at the core of the Semantic Web is the RDF data model, which can be used to represent and connect data (Heath & Bizer, 2011). To add data descriptions and

[3] http://google.com

[4] Arthur Samuel, as cited in Munoz (2014), defined machine learning as "the field of study that gives computers the ability to learn without being explicitly programmed".

meaning to the data it is possible to add an ontology (for example in OWL format). When RDF data and ontological information are stored in a RDF database, the data can be queried with a query language (such as SPARQL), and inferences can be made.

Automatic question generation differs from traditional question answering in that the questions are automatically generated. When creating automatic question generation applications we have many techniques at our disposal, from fields such as IR and NLP. There are also different ways to represent and store the data used for question generation. As this suggests, there is a lot of variety in how to develop an automatic question generation system. Several projects have focused on automatic question generation (Zeng et al., 2013; Xu, Goldie & Seneff, 2009; Kim & Kim, 2008; Fjellanger & Armstrong, 2013). These projects differ in techniques and technologies used. The techniques used in the projects are related to various NLP tasks, and IR (search). Among the technologies used are relational database technologies, and Semantic Web technologies. In contrast to the quizzes made by Chen (2013), none of the automatic quiz generation projects were aimed directly at an elderly population, and none of them were in Norwegian.

This master project was suggested by Weiqin Chen; who was my professor in the course "INFO323: Data Architectures for Information Retrieval and Web Intelligence" at University of Bergen; and who had previously developed the tangible tabletop quizzes that were made specifically for elderly people. As we have seen earlier in this section, studies have indicated that games can have benefits for the elderly, for instance through reducing cognitive loss (such as cognitive decline and dementia), increasing relative well-being, or even by (depending on the game) improving cognitive functions (such as short-term or working memory). Based on the ill-effects of loneliness and isolation, we can also imagine that elderly people participating in social activities with their personal network, for example playing a quiz, could have health benefits for some.

The aim for this project was to create a Norwegian quiz application for elderly people, where the questions and answers were automatically generated from online sources. The questions should be factual, and they could for example ask about people that were well-known, past events, or works (such as movies and literature) that were created during the quiz contestants past. The quiz would have multiple choice alternatives, and it should be possible to play it both alone and with other people. We wanted to make a quiz that could help the quiz contestants to remember,

and which enabled them to spend time together while participating in a social activity. Since the quiz should help quiz contestants to remember, it should not have a very high difficulty level.

1.2 Research question and method

Based on the aim for this master project, the following research question has been examined:

> "How can quizzes for Norwegian elderly be automatically generated using online resources?"

In order to answer this question, I have developed a quiz application (high-fidelity prototype) for Norwegian elderly people (also referred to as the elderly quiz), where questions and answers are generated automatically. Different ages are used to define a person as elderly; two of the more common definitions start from 60 and 65 years old, respectively (World Health Organization, 2016). The target audience for the quiz are Norwegian elderly people above the age of 60.

The development model used for developing the quiz was the incremental model, combined with iterative development (see Section 3.1.1). Each iteration has aimed to deliver a new increment of functionality. Different tasks, techniques and technologies have been researched and applied to answer the research question; largely within the field of NLP (IE) and the Semantic Web. The source for quiz data has been Wikipedia.

1.3 Organization of the thesis

The organization of this thesis is as follows: Chapter 1 (this chapter) has provided an introduction to the thesis; this includes presenting the research question that the thesis will attempt to answer. Chapter 2 presents relevant theory for the thesis. Chapter 3 first presents development method and requirements for the elderly quiz. Thereafter, the chapter presents the quiz architecture, and provides implementation information for the individual quiz components. The final part of the chapter gives an overview of the development process. Chapter 4 presents the evaluation of the quiz, which was conducted by means of usability testing. Chapter 5 concludes the thesis, and suggests future work for the quiz application.

Chapter 2 - Literature Review

Creating the elderly quiz involved retrieving web pages from Wikipedia, using IE to extract quiz data from those web pages, and representing and storing the quiz data with Semantic Web technologies. It should be noted that when the Wikipedia derivative DBpedia (see Section 2.3.3) was used as a source for quiz data; retrieval, extraction and representation of the information had already been carried out by the DBpedia community. This made it possible to reuse some of their collective efforts; storing desirable data for the quiz.

The first part of this chapter will focus on representation and storage of data, with emphasis on Semantic Web technologies. The second part of this chapter will present information extraction (IE). Because data in the elderly quiz comes from Wikipedia, it will be the focus of the third part of this chapter. The third part of the chapter also includes information about the two Wikipedia derivatives, based on Semantic Web technologies, that were used in the elderly quiz; DBpedia and YAGO.The fourth part of this chapter will present question answering, and will review some earlier research on automatic question generation.

2.1 Information representation and storage

Upon extracting data of interest, we will often want to store it in a type of database system. In addition to ensuring persistence, such a system can allow for further benefits such as query capabilities, representing complex relationships among data and enforcing data integrity constraints. We can store not only data instances, but also a description of our database and the type of data in it. The latter is often referred to as a schema, letting us add structure to the data (Elmasri & Navathe, 2011). With regards to Semantic Web technologies: Data instances can be represented as triples, using the Resource Description Framework (RDF) model. The schema can be represented using ontologies, which are also suitable for adding semantics (meaning) to data (Ma et al., 2007). Instance data and ontologies can be stored together in a RDF database.

2.1.1 RDF

"RDF provides a generic, graph based data model with which to structure and link data that describes things in the world" (Bizer, Heath & Berners-Lee, 2009, p. 3). In RDF, a resource is described using triples. Triples are statements (facts) that link data about things in the world. Each triple consists of a subject, predicate and object. The subject is the resource being described with the triple. The object is either another resource or a literal. Resources are expressed using a Uniform Resource Identifier (URI). When used together with the HTTP protocol, a URI can be used to uniquely identify a resource across the Internet. A literal can represent values such as a number or a string. The subject and object in a triple are bound together by the predicate (property), which specifies the relationship between them. The predicate is represented using a URI (Heath & Bizer, 2011).

An example of a triple from the DBpedia dataset follows below. It describes a fact about the United States (subject); i.e., that its largest city (predicate) is New York City (object).
Subject: *http://dbpedia.org/resource/United_States*
Predicate: *http://dbpedia.org/ontology/largestCity*
Object: *http://dbpedia.org/resource/New_York_City*

If we use namespace[5] prefixes, we can omit writing the full URIs for RDF triples. Prefixes must be declared before they are used; each prefix must refer to a unique namespace. The triple above could be written as *<dbr:United_States dbo:largestCity dbr:New_York_City>* (this would require the *dbr:* prefix to refer to *http://dbpedia.org/resource/*, and the *dbo:* prefix to refer to *http://dbpedia.org/ontology/*, which is the convention). Prefixes are used throughout this thesis, for brevity.

2.1.2 Ontology

An ontology can be defined as: "A formal, explicit specification of a shared conceptualization" (Guarino, Oberle & Staab, 2009). An ontology can be used to describe one or more domains, with domain specific facts. Such a description may include what type of concepts (classes) are in the domain(s), which properties the concepts may have, and the relationships between the concepts (Chandrasekaran, Josephson & Benjamins, 1999). When we store ontological facts in

[5] A namespace is "an identity space on the Internet", and is the first part of a URI (W3C, 2011).

a RDF database, they can be used for inferencing. In a Semantic Web context, inferencing means that given some stated facts, we can state one or more new and related facts. A simple example of an inference is that if we know that James is a musician, and we know that every musician is a person, then we also know that James is a person. Such an inference can be made using a subsumption hierarchy, where we state that the class musician is a subclass of person (Allemang & Hendler, 2011).

Ontologies in Semantic Web technologies are often represented in the ontology languages RDF Schema (RDFS) and Web Ontology Language (OWL). While OWL is more expressive than RDFS, both languages make available capabilities such as subsumption hierarchies and let us do basic inferencing over data (Allemang & Hendler, 2011). We can relate a RDF data resource to an ontology class using RDF triples, for instance we can say that James is a member of the musician class using this statement: *<person:James rdf:type dbo:Musician>*. Such a relationship is possible when instance data and ontologies are stored together in a RDF database. *rdf:type* is a common predicate which indicates an "is a" relationship directed from the subject to the object of a triple[6].

2.1.3 RDF database

A RDF database is a type of database that models the structure of RDF data. It is tuned for access to and retrieval of RDF triples (Allemang & Hendler, 2011). The RDF database automatically merges facts from different data sources. A RDF database can be seen as being part of a larger system which is based on Semantic Web technologies, a Semantic Web framework (Hebeler, Fisher, Blace & Perez-Lopez, 2009). In addition to storage, the framework can also consist of components for access and inferencing. Access components are often RDF APIs or query processors. RDF APIs enable information to be retrieved and changed programmatically. The query language that is typically used for querying of RDF is SPARQL, which is a W3C standard (Allemang & Hendler, 2011). Querying with SPARQL involves defining one or more (triple) patterns, which are matched against actual RDF triples. SPARQL triples contain one or more variables in place of RDF resources. A query engine returns resources in place of variables, for all triples in the RDF database that matches the patterns (W3C, 2015b).

[6] The convention is for the *rdf:* prefix to refer to: *http://www.w3.org/1999/02/22-rdf-syntax-ns#*.

2.2 Information extraction

Information extraction (IE) can be defined as the process of extracting structured data from the content of input documents, which typically contain semi-structured or unstructured text (Chang, Kayed, Girgis & Shaalan, 2006). Structured data can be viewed as data which have well-defined semantics and relationships, and are often suitable for storage in a database. Unstructured text (or data) is not inherently without any structure, but is "expressed in a form that is difficult for computers to interpret directly" (Kolomiyets & Moens, 2011).

2.2.1 IE sub-tasks

The following text will give an overview of a selection of IE sub-tasks, which can all be used to extract structured data from the content of input documents. While technically an IE subfield, Ontology-based information extraction (OBIE) will also be presented under IE sub-tasks, due to its similarities with Web IE.

Traditional IE aims at extracting data from unstructured text which is written in natural language. Documents on the Web (such as HTML documents), however, are semi-structured; often being dynamically generated by a server-side application[7]. The task of conducting IE on web documents can be referred to as Web IE. Web IE tends to apply extraction rules and machine learning techniques to exploit the syntactical patterns or layout of a web document (Chang et al., 2006). "Programs that perform the task of IE are known as extractors or wrappers" (Chang et al., 2006). An extractor[8] can be viewed as wrapping an information source, for example a web server. In this way the information source can be accessed, and structured data can be collected and integrated into another data source, such as a database (Chang et al., 2006).

As mentioned earlier in this section, OBIE is a subfield of IE. An OBIE system is "a system that processes unstructured or semi-structured natural language text through a mechanism guided by ontologies to extract certain types of information and presents the output using ontologies" (Wimalasuriya & Dou, 2010). In OBIE an ontology is used as a model to guide the IE process, such as what information should be extracted and how the information should be output.

[7] While other researchers might classify web pages as unstructured, Chang et al. (2006) consider them to be semi-structured, due to data often being embedded regularly within HTML tags.
[8] While Chang et al. (2006) suggest that the terms extractors and wrappers can be used interchangeably, this thesis will only use the term extractors.

Instances are commonly only extracted with respect to classes and properties of a given ontology ("ontology-population"). Some potential areas for OBIE is to improve the quality of ontologies, create metadata for the Semantic Web and automatically process information in Natural Language text (Wimalasuriya & Dou, 2010).

A common IE sub-task, sometimes as a starting point in an IE application, is that of named entity recognition (NER). A named entity refers to "anything that can be referred to with a proper name" (Jurafsky & Martin, 2014, p. 741). NER involves finding spans of text that make up a proper name, and to classify these entities based on their type (where a named entity system can classify proper names as a fixed number of entity types). Factors such as the language and genre of the text need to be taken into account, as a proper name in a specific language or genre may be signalled in a way that differentiates it from the other text. Generic named entity systems (which are typically news-oriented) tend to detect entities such as people, locations and organizations. A more specialized named entity system can focus on detection of other entity types, for example works of arts or proteins (Jurafsky & Martin, 2014). An example can illustrate NER: Given the sentence "Kari Simonsen was born in Oslo", a (generic) named entity system might (correctly) detect that "Kari Simonsen" is a person, and that "Oslo" is a location. A common challenge with NER is ambiguity. For instance the term "Nordahl Grieg" may refer to a person (the Norwegian writer Nordahl Grieg), or to an organization (Nordahl Grieg upper secondary school).

One task that often relies on the results of NER is that of relation detection and classification. The goal of relation detection is "to discern relationships that exist among the entities detected in a text" (Jurafsky & Martin, 2014, p. 748). It is then possible to classify the type of relationship that exist among the entities. Text used for relation detection is unstructured. To give an example of a relationship between entities, we can return to the sentence "Kari Simonsen was born in Oslo". The named entities here are, as mentioned earlier, Kari Simonsen (person), and Oslo (location). We see that there is an explicit relationship between the two entities in this sentence, Kari Simonsen was *born in* Oslo. This relationship can be expressed more formally as a relational triple; *BornIn(Kari Simonsen, Oslo)*. The relation is binary, since the relationship is between two entities in a text. Binary relations are the most common to detect (Jurafsky & Martin, 2014). The relational triple is suitable for representation in a RDF triple, where the subject is "Kari Simonsen", the predicate is "BornIn" and the object is "Oslo" (resource URI's not included).

2.2.2 IE techniques

Different techniques are used for IE. Among other, these techniques include analyzing HTML/XML tags, hand-written rules, and the broader category of machine learning techniques. Which technique is the most appropriate will depend on the type of IE sub-task being accomplished (Wimalasuriya & Dou, 2010). For the elderly quiz, analyzing of HTML/XML tags was used for Web IE. Hand-written rules (represented with a variant of regular expressions) were used for NER, relation detection and classification, and general post-processing of extracted text. While not being explicitly used in the elderly quiz, machine learning techniques are widely applied in IE sub-tasks, such as NER (Nadeau & Sekine, 2007). The following text will give an overview of the mentioned IE techniques.

2.2.2.1 Analyzing HTML/XML tags

Data from web pages (HTML documents) is not particularly well-suited for querying and manipulation, compared to data in a database. Thus, it may be advantageous to extract data from web pages and let the data populate databases before being managed further[9] (Laender, Ribeiro-Neto, da Silva & Teixera, 2002). When analyzing HTML tags, a parse tree is constructed from a web page. Extraction rules can thereafter be applied to the tree.

The HTML parse tree will typically be a Document Object Model (DOM) node tree. As described in the DOM 4 recommendation, DOM is an interface that enables programs to access the structure, content and style of documents[10] (Kesteren, et al., 2015). The DOM interface is not limited to HTML, other document formats (such as XML[11]) are also supported. Based on DOM tree constraints, different nodes can (or must) appear in the node tree. DOM nodes include the document node (the root in a node tree), which specifies attributes such as document file format and encoding; element nodes (referred to as elements); and text nodes, which collectively contain an element's text. An element has a tag name, possible attributes, and element children. An element child can be another element or a text node. Text nodes are leaf nodes, meaning they can not have child nodes.

[9] If necessary after some post-processing.
[10] While different browsers and programming libraries tend to have some variation in how they implement DOM, this text aims to give a high-level description of some aspects of DOM, in adherence to the W3C DOM4 recommendation and how the DOM is implemented in jSoup (see Section 3.3.8 for more information about jSoup).
[11] Extensible Markup Language.

Extraction rules, in the context of HTML extraction, are often highly accurate rules to extract a particular web page's content, such as information within specific HTML tags (Wimalasuriya & Dou, 2010). Before extracting from the parse tree, extractors need to recognize the appropriate data, among a lot of uninteresting pieces of text. Another challenge is that different web pages will have structural variations which need to be taken into account. These challenges may complicate the writing of efficient extraction rules. Since web documents are given as input, and rules are applied, we can infer that the technique of analyzing HTML tags, in the context of extraction, is typically used in Web IE. A similar technique of parsing and extraction based on tags can be used for XML documents (Laender, et al., 2002). The actual parsers used for HTML and XML documents will tend to differ, due to markup language differences.

2.2.2.2 Hand-written rules

As mentioned earlier, hand-written rules are used for NER, relation detection and classification, and general post-processing of extracted text in the elderly quiz. These rules all use patterns that should be matched against. For NER the hand-written rules are classification rules. A rule-based NER system tends to have a lexicon (gazetteer lists each containing terms for a given entity type), and grammars that checks if a term is or is not within a lexicon. If a term is found in a gazetteer, classification rules can decide if it should be classified with the entity type of the gazetteer or not. Farmakiotou et al. (2000) note that hand-written classification rules can take into account factors such as the words or symbols that make up the term (internal evidence) and the context where the term appears (external evidence). Hand-written NER rules may achieve good performance, but at a high engineering cost. According to Nadeau and Sekine (2007), hand-written NER rules are the preferred alternative when we do not have access to training examples to use for supervised learning.

For relation detection and classification, we can write rules as patterns that match segments of text which are likely to contain relations. There is generally a trade-off between pattern generality and pattern precision. More general patterns tend to result in more matches, while more specific and precise rules tend to yield less false positive matches. If we want better coverage without making our patterns more general, we need more patterns (Jurafsky & Martin, 2014). The rules for NER and post-processing used for the elderly quiz are both written as regular expressions[12].

[12] The NER regular expressions currently only consist of character strings that are to be matched literally, in a gazetteer.

Relation detection and classification rules are written in a syntax that extends regular expressions (TokensRegex). Because variants of regular expression rules are used in the elderly quiz, the following text will present this concept:

Regular expressions are widely used in IE applications for defining extraction rules. They were first developed in the 1950s, although there have been major changes to their notation and expressivity since then. "A regular expression is an algebraic notation for characterizing a set of strings" (Jurafsky & Martin, 2014, p. 18). Regular expressions can be used to define search strings. To search we need a regular expression pattern that we want to search for, and a corpus (document collection) that will be searched through (Jurafsky & Martin, 2014). When we find text that matches the pattern we might for instance want to extract all or parts of the text; replace all or parts of the text with another text before extracting it; or split the matching set of strings into several new sets before extracting them. Exactly which capabilities are available depends on the programming language (or library) being used (Goyvaerts & Levithan, 2012).

A pattern can consist of characters that are to be matched literally, and characters that have a special meaning (metacharacters). Whether the characters should be matched literally or not is based on where they are in the pattern. In most cases alphanumeric characters are to be matched literally. Some of the regular expression metacharacters are the quantifier characters (quantifiers), parentheses, and brackets.

Quantifiers let us specify how many times a part of the pattern should be matched. They include characters such as the plus character (match one or more times) and braces (let us specify the minimum and maximum, or exact, amount of times to match). Parentheses let us group a set of characters. Putting parentheses around a part of the pattern acts as a capture group. When we have parts of a pattern in a capture group, it makes it easy to extract only that part. Capture groups are numbered, starting from 1, by their occurrence in the pattern (from left-to-right). Brackets are used to specify character classes, meaning that the text needs to match one of the characters specified within the brackets in the pattern. When containing a hyphen, character classes can specify ranges, such as [0-9] (matches one character between zero and nine). When the characters inside of the character class start with a caret, it negates the character class (matches any character which is not the character within the brackets). Shorthand syntax exists for some character classes, for instance \d which is a shorter way of writing [0-9]. Metacharacters must be escaped if we want them to be interpreted literally, this is typically done with a backslash character (Goyvaerts & Levithan, 2012).

To give a simple example, we can consider the following:
/Hello World (\d{4})!+/

In this example we have the following metacharacters: Plus and braces are quantifiers, parentheses are used for grouping, and \d is shorthand for the character class [0-9]. The plus tells us that we need to match one or more instances of the preceding character (the exclamation mark). The \d followed by braces with the number 4 in them, tells us that we need to match exactly four consecutive occurrences of a digit (such as a year). The parentheses put the matching year in capture group 1 (the first and only capture group in this example). The alphanumerical characters and the spaces are to be matched literally. The starting and trailing forward slashes are part of the perl syntax for regular expressions (used in this thesis for readability) and signify a pattern that should be matched. The pattern above will match the following text in a document collection: "Hello World 1984!". Here the year is 1984, and we have one exclamation mark. It will also match "Hello World 2016!!" (the year is now 2016, we have two instances of the exclamation mark), and so on. We can access the matched year separately using capture group 1.

2.2.2.3 Machine learning techniques

This section will present two broad machine learning techniques that are used for IE, supervised learning and semi-supervised learning, giving IE relevant examples of each. The section will start with supervised learning.

Supervised learning

Supervised learning techniques train a classifier using manually annotated examples; a training set. The classifier uses patterns learned from the training set to classify new data (Kolomiyets & Moens, 2011). To give an example of a supervised learning technique widely used for NER classification, we can look at sequence labeling.

NER sequence labeling involves manually labeling all individual tokens (words or characters) in selected documents with their named entity type and named entity boundary (because named entities might span several consecutive tokens), or with the absence of a named entity type. The annotated documents make up a document collection. To build a sequence labeling classifier one also needs to develop some features that can be extracted from the document collection (Jurafsky & Martin, 2014). Features represent distinguishing attributes of words that are

designed to be used by algorithms (Nadeau & Sekine, 2007). It is important to find features that will be useful for classification.

While the usefulness of features will vary in different NER systems; features might for example include shape features (capitalization and other word patterns), gazetteer presence, and features based on predictive words in the context window (where the context window might for instance include the two tokens before and the two tokens after the token being labeled). Other features related to tokens might also be used, such as their part of speech (POS). The individual tokens in the documents, their selected features, and their NER label can thereafter be encoded in a modified document collection; a training set. The training set is used as input to a (machine) learning algorithm for building a (statistical) sequence labeling classifier. When given sentences from new (unseen) documents, the sequence labeling classifier can sequentially label tokens with their predicted type of named entity and boundary, or with the absence of a named entity type (Jurafsky & Martin, 2014). Different learning algorithms can be used for building a sequence labeling classifier, including Hidden Markov Models (HMM) and Conditional Random Fields (CRF).

To evaluate supervised learning learning algorithms in a way that makes them comparable to one another, it is possible to train and test a classifier using (separate parts of) a standard evaluation corpus. Among such corpora is the Automatic Content Extraction (ACE)[13] corpus, which can be used both for NER and relation detection and classification. Using evaluation on standard corpora, it has been shown that state-of-the art named entity recognizers can label (English) text at high accuracy. Relation detection and classification is a more complex task, and has not yet achieved the same levels of accuracy. A possible drawback with using supervised learning techniques for classifier induction is that the process requires a large, manually annotated document collection. Classification can also be computationally burdensome (Bach & Badaskar, 2007).

Semi-supervised learning

As mentioned earlier, it is possible to write hand-crafted patterns for different types of IE, for example when there are not enough training examples to train a supervised learning classifier. Another alternative to supervised learning, that does not require a lot of training examples, is that of semi-supervised (lightly supervised) learning. The main technique of semi-supervised

[13] https://catalog.ldc.upenn.edu/LDC2006T06

learning is bootstrapping. Here we start the learning process using a small seed set (Nadeau & Sekine, 2007).

As an example, Jurafsky and Martin (2014) show how we can extract relations (combining tasks of NER and relation detection and classification), starting from either a set of tuples, or a set of seed patterns. Tuples contain two named entities that occur together in a given relationship, while patterns typically contain the textual context that occurs before, between, and/or after, two named entities that are in a relationship. Patterns can for instance be represented with regular expressions. Using a document collection, entities in relationships can be found that are in the initial tuple set, or that fit with the initial pattern set. In addition to extracting the relations, we can expand on the pattern list with other contexts in the document collection that the entities appear in. Then we can do another search given the new patterns, find new relationships between entities that fit the pattern, extract the relations, expand the pattern list with new contexts in the document collection where such entities appear together in a relationship, and so on. This process can be continued until a stop criterion is reached.

The quality of the document collection and the patterns will obviously affect the accuracy of using such a semi-supervised method. Adding erroneous relations could again lead to new patterns that capture erroneous contexts. Therefore, one might make additional rules for the accepting of new relations and new patterns. According to Jurafsky and Martin (2014), there are no standard evaluations publicly available for this form of relation extraction, but the technique has gained acceptance as a way to quickly populate databases with relations from sources such as the Web.

2.3 Wikipedia

Wikipedia is the world's most widely used encyclopedia. The website was launched in 2001; as of May 2016 it is the seventh most popular website in the world[14]. At the time of writing, 292 different language editions are available. The English version has the highest number of articles; 5.16 million in total. The different language versions also include the two official forms of written Norwegian; Norwegian "Bokmål" and Norwegian "Nynorsk". Most of the text and a lot of the images on Wikipedia are licensed under a Creative Commons license (CC-BY-SA 3.0[15]). This

[14] http://www.alexa.com/siteinfo/wikipedia.org
[15] Attribution-ShareAlike 3.0 Unported; https://creativecommons.org/licenses/by-sa/3.0/.

license allows for sharing and adapting of information (for instance by including it in another application), but in turn requires attribution of the author (if one exists) or of the licensor (Wikipedia). Articles in Wikipedia are written collaboratively, by volunteers. Most articles can be written and edited by anyone on the Internet; anonymously if desired. A small number of articles are protected or semi-protected. An article being protected means only certain editors are allowed to make changes. Semi-protected articles require users to fulfil certain criteria (namely having an account for more than four days, and having made at least ten previous edits) before making changes (Wikipedia, 2016). The Wikipedia article used as a reference in this thesis is semi-protected.

2.3.1 Wikipedia data accuracy

Using Wikipedia as a data source brings up the questions about how accurate Wikipedia is. Since most Wikipedia articles can be edited by anyone, anonymously, there is always a chance that upon data extraction, information in a given article can be inaccurate. Although inaccuracy can have different dimensions; for data used in a quiz it is important that it does not contain factual errors and it is preferable to avoid spelling mistakes. Other dimensions such as errors of omission and lack of reliable references (for correct facts) affect the quality of an encyclopedia negatively, yet they will not affect the elderly quiz notably (except that errors of omission can limit and possibly skew the data available for question generation).

Different studies have addressed Wikipedia accuracy, focusing on different domains of knowledge. One study had 42 relevant experts peer review one article from Wikipedia and one article from Encyclopaedia Britannica[16], about a single science subject. The study found that Wikipedia had on average around four inaccuracies per article. The inaccuracies were mainly factual errors, omissions and misleading statements (Giles, 2005). A study by Rector (2008), that used content analysis to examine nine historical articles, found Wikipedia's accuracy rate to be 88 percent (when excluding inaccuracies due to lack of reliable references for correct information)[17]. Wikipedia has been found to be close to or as accurate as an established, centrally-controlled encyclopedia (Giles, 2005; Rosenzweig, 2006); while there have also been studies that found that established encyclopedias were more accurate (Rector, 2008). In one

[16] http://www.britannica.com/

[17] When including inaccuracies due to lack of reliable references the accuracy for Wikipedia was 80 percent.

study from 2012, that used expert evaluation of mental health topics, Wikipedia was rated more highly on accuracy than both a psychiatry textbook and Encyclopaedia Britannica (Reavley et al., 2012).

Given the scope of Wikipedia it is difficult to make broad generalizations about its data quality based on individual studies, but they can point out some trends. Findings in the mentioned studies can indicate that the material in Wikipedia should be satisfactory ("good enough") for use in a quiz. Wikipedia was also found to be suitable for the elderly quiz because it was cross-domain, meaning it could be used to extract information to create questions about different types of entities.

2.3.2 Wikipedia Infoboxes

A Wikipedia article largely consists of free text, but some of its parts are structured information in the form of wiki markup. Wiki markup is a markup language that is used in Wiki-based websites. It works as a simplified alternative and an intermediate to HTML. The structured information in Wikipedia includes infobox templates and categorisation information (Lehmann et al., 2015). Such information, having been extracted, is available in DBpedia. Contents extracted from infoboxes are perceived as the most valuable information for DBpedia extraction (Bizer, Lehmann et al., 2009). A lot of the information used in the elderly quiz originates from Wikipedia infoboxes.

An infobox is a table that can be added to a Wikipedia article at the top-right corner (for western languages). It may contain facts and statistics that pertain to the resource the article is about, in addition to an associated image. Infoboxes are created using existing infobox templates. A template contains information that is common to a number of articles; it defines parameters (properties) that each can be given a value. For example, the infobox template "musical artist[18]" can be used in a Wikipedia article for a band or a musician. For a musician, the properties that can be given a value include "name", "birth_date" and "genre". The value can consist of several items if appropriate; for example, in the infobox for the musician Morten Harket[19], some of the values for genre(s) are "pop rock" and "alternative rock". When an infobox template is used for

[18] http://en.wikipedia.org/wiki/Template:Infobox_musical_artist
[19] https://en.wikipedia.org/wiki/Morten_Harket

an infobox in a Wikipedia article, it is not possible to display any other parameters than those already defined for that specific template.

2.3.3 IE from Wikipedia

As noted in Section 2.2.2.1, data from web pages is not particularly well-suited for querying and manipulation. To remedy this, DBpedia is a community project that extracts structured knowledge from Wikipedia web pages. Its goals include better search and querying capabilities of Wikipedia data. The knowledge in DBpedia is published using Semantic Web standards. DBpedia is multi-lingual, covering more than 100 different languages. Both English and Norwegian (Bokmål and Nynorsk)[20] datasets are available. The most downloaded dataset is the English dataset. DBpedia contains both instance data, and categorisation information in the form of an ontology. The ontology is maintained by the DBpedia community (Lehmann et al., 2015).

DBpedia uses its extraction framework to extract structured information from Wikipedia. Some other information, which is not as structured, can also be extracted, for instance the abstract of an article. The extraction framework includes extractors for different types of information that can be extracted from an article; such as image (first image of a Wikipedia article, and its corresponding thumbnail), title (denoted as "label" in DBpedia), and infobox information. In all there are 24 different extractors (Lehmann et al., 2015). Which languages are covered by an extractor differ; some extractors can be used for all languages available in DBpedia, others can only be used for a few different languages, while the rest are only available for English.

Extraction in DBpedia has four phases. In the first phase Wikipedia web pages are read from an external source, such as a Wikipedia dump. Dumps can be downloaded for different Wikipedia languages[21]. In the second phase each Wikipedia web page is parsed, using the source code to construct an abstract syntax tree[22]. In the third phase each abstract syntax tree is consumed by the different extractors; they in turn return extracted data in the form of RDF triples. In the fourth and final phase all generated RDF triples are written to a dataset, in a given RDF serialization format (Lehmann et al., 2015). In the case where an XML dump of Wikipedia is used, we see

[20] http://data.dws.informatik.uni-mannheim.de/dbpedia/3.9/ - Bokmål dataset is available in NO folder, nynorsk dataset is available in NN folder.
[21] http://en.wikipedia.org/wiki/Wikipedia:Database_download
[22] An abstract syntax tree is a "tree representation of the abstract syntactic structure of source code written in a programming language" (Li & Zhong, 2010). The tree is abstract in the sense that some of the source code syntax is abstracted away.

that this process (phases 1 to 3) is similar to the IE technique of analyzing XML tags (see Section 2.2.2.1), with the exception of using an abstract syntax tree instead of a parse tree.

When a DBpedia dataset is made publicly available it can be inserted into a RDF database. The data might be accessible over the Internet, for example using a SPARQL endpoint[23] that can be queried; or one might have a locally installed RDF database containing its own copy of DBpedia. There are two types of infobox extractors in DBpedia. One extractor is for general extraction of infobox knowledge (raw infobox extraction), while the other is for extraction of infobox templates which have been manually mapped to DBpedia ontology concepts by the DBpedia community (mapping-based infobox extraction). With raw infobox extraction, there is a direct mapping from infobox to RDF (Lehmann et al., 2015). The name of the relevant infobox property is thus automatically used as the name of the predicate (not including the full URI) when writing information as a RDF triple.

Because Wikipedia allows different templates for the same concept, mapping-based extraction provides a way of unifying different templates that cover one concept. The mapping-based approach also allows for explicit data typing of different infobox properties. Raw infobox extraction, on the other hand, relies on heuristics to determine data types, resulting in lower data quality. Even though mapping-based extraction results in higher quality data than raw infobox extraction, raw infobox extraction has higher coverage, as not all templates have been mapped for a given language at a given time (Lehmann et al., 2015). As of May 2016 there are no mappings available for Norwegian language Wikipedia infoboxes to the DBpedia ontology[24].

In addition to DBpedia datasets extracted from Wikipedia, there also exist DBpedia datasets that interlink DBpedia with external datasets. These links can be either incoming, where external datasets link to DBpedia resources, or they can be outgoing, where DBpedia resources point to external datasets. Several datasets link from DBpedia to different information in the semantic database YAGO[25]. YAGO is automatically built using Wikipedia. This is a project with similarities to DBpedia, they both contain RDF resources from all articles in Wikipedia. As DBpedia, YAGO also extracts structured information from Wikipedia, for example from infoboxes (Lehmann et al., 2015).

[23] A SPARQL endpoint “is an application that can answer a SPARQL query” (Allemang & Hendler, 2011).
[24] http://mappings.dbpedia.org/index.php/How_to_edit_DBpedia_Mappings
[25] http://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/

One way that YAGO is different from DBpedia is that Wikipedia leaf categories are used for inferring RDF type information for a resource. Wikipedia categories are organized hierarchically in a tree structure and aim to group articles about similar subjects. If a leaf category is used as a class, it becomes a part of the YAGO taxonomy (which is a part of the YAGO ontology). YAGO then utilizes WordNet[26] to infer further class information. For example, the article about the actor "Leif Juster" is in the category "Norwegian male singers". YAGO uses this information to infer that Leif Juster is a (type of) Norwegian male singer. Utilizing WordNet it can be inferred that "Norwegian male singers" is a subclass of the WordNet synset[27] "singer" (Hoffart, Suchanek, Berberich & Weikum, 2013). Leaf categories and WordNet lead to YAGO having more classes in its ontology than DBpedia does. Another difference between YAGO and DBpedia is that YAGO generally uses expert-designed declarative rules for integrating information from infoboxes to their respective ontology, while in DBpedia this is done by the community which manually define mappings. Fewer RDF triples are extracted by YAGO than by DBpedia. This may be due to YAGO being focused on high precision[28] and consistent knowledge (Lehmann et al., 2015).

DBpedia is a popular hub on the Semantic Web. It gives an increasing value in the form of data; as Wikipedia grows, so does the amount of facts in DBpedia. For the quiz itself the English version of DBpedia was chosen. This was due to factors such as data quality (infobox mapping dataset), community support (no active Norwegian DBpedia community yet) and availability of datasets with links from DBpedia resources to knowledge about them in other datasets. Datasets containing links from DBpedia resources to YAGO type information were used for selecting some of the DBpedia resources used in the quiz application's automatic question generation. The reason was that when conducting queries and exploration of DBpedia datasets with the DBpedia online SPARQL endpoint[29], it was found to be convenient to use resource class information from YAGO. For instance, one could then easily obtain all DBpedia resources that were of the type "NorwegianMaleSingers". One drawback of not using the Norwegian DBpedia version is that "non-English DBpedia editions comprise a better and different coverage of local culture" (Lehmann et al., 2015, p. 26).

[26] https://wordnet.princeton.edu/
[27] A synset is a group of synonyms that express a distinct concept.
[28] Precision can be defined as “the fraction of retrieved documents that are relevant” (Manning et al., 2008). In the context of extracting RDF triples, precision is related to the fraction of extracted triples that are relevant (correctly extracted).
[29] http://dbpedia.org/sparql

2.4 Question generation and answering

This section will start with a presentation of question answering, since an automatic quiz generation application is a special type of question answering system. Thereafter, some earlier projects regarding automatic question generation will be reviewed.

2.4.1 Question answering

As mentioned in Section 1.1, IR is about finding materials that satisfy an information need within (computer-based) collections (Manning et al., 2008). While materials are often unstructured documents, they can also be of another nature, such as structured or semi-structured records, and multimedia objects (Baeza-Yates & Rebeiro-Neto, 2011). Question answering is a type of IR where information needs can be expressed as natural language statements or questions by a user. Asking questions is a natural way of human communication with a computer. The field of question answering is several decades old, in the beginning often providing natural language interfaces to databases (structured data, narrow domains), but the field has recently had an increase in interest due to the increasing amount of information both on the Web and in more proprietary databases. A question answering system should return a specific piece of information as an answer to the user (for example in the form of a word or a sentence), according to the user's information need (Kolomiyets & Moens, 2011).

In contrast to a keyword based search, a question provides syntactic and semantic relationships between its terms. There are several types of questions; for the elderly quiz the type used is factoid questions. Factoid questions often start with a Wh-interrogative word ("what", "when", "where", "who"). They require answers that contain facts (Kolomiyets & Moens, 2011).

Questions and textual materials (for instance documents) in a question answering system are typically pre-processed and translated into a format that can be read by a computer. For example, in the case of using Semantic Web technologies, a question can be translated into a SPARQL query and the content of documents can be stored as RDF triples. An information source is a collection of information objects (for example documents or text) that is used by a question answering system for retrieval of answers. Information sources can be structured data sources (such as a database) or unstructured data sources (such as a collection of unstructured text documents). Question answering systems may be for open domains or restricted domains.

For open domain question answering on facts, using data redundancy may be a way to find the right answer, given that the right answer to a question will appear in more places in a large collection than a wrong answer (Kolomiyets & Moens, 2011).

A retrieval model consists of a user information need, an information source and a retrieval function for estimating the relevance between a user query (question) and a possible answer. Different question answering systems can have different retrieval models, using dissimilar approaches for data and query representation, and retrieval functions. Retrieval functions have varying degrees of complexity, computational complexity and success rate in finding the correct answer (Kolomiyets & Moens, 2011). While some retrieval functions include ranking of possible answers, others will use a boolean type of matching (where a candidate answer either meets the required conditions for being retrieved or it does not).

As mentioned in Section 1.1, automatic question generation differs from traditional question answering in that the questions are automatically generated. The elderly quiz is an automatic question generation application. The quiz is used for a subset of open domain question answering. The questions asked are factoid questions. Text is represented using RDF triples which are stored in a RDF database, being a part of a Semantic Web framework. In this way data for the question and the multiple choice alternatives (both the right answer to a question and distracters) can be retrieved using RDF APIs. Retrieval of appropriate distracters for a question is based on meeting certain similarity criteria compared to the correct answer. Which criteria is used depends on the type of entity being chosen as the answer.

2.4.2 Earlier research on automatic quiz generation

This section will present some of the earlier research regarding automatic quiz generation, where each project resulted in an instantiation (system). The areas of research, and the techniques used to generate questions (and if applicable, answers), vary greatly between the projects. The project in Section 2.4.2.1 mostly relies on IR techniques, using advanced search capabilities to generate questions and multiple choice answers that are appropriate with regards to a user information need. The project in Section 2.4.2.2 is more focused on applying techniques related to the NLP tasks of language generation and language understanding. The project in Section 2.4.2.3 implements basic search capabilities, but focuses on NLP tasks such as NER for generating questions and corresponding answers. The project in Section 2.4.2.4

reuses information that had previously been extracted about movies, and uses Semantic Web technologies to generate questions and multiple choice answers.

2.4.2.1 Automatic generation of tourism quiz using blogs

Zeng et al. (2013) made a proposal for, and developed an automatic multiple choice quiz generation system, using tourism blogs. The system generates Japanese language quizzes, related to tourism. The quizzes are made after a user enters a topic keyword in a search field. The aim of this system was to make the users remember information easily.

Since this system uses search, the system has a full-text index and a sentence index, to optimize speed and performance while searching. To prepare each term in the document collection for indexing, the NLP technique of morphological parsing was applied[30]. The indices were constructed and saved in index files using the Japanese tool GETA[31]. GETA is specialized in associating groups of text with other group of text, based on similarity (associative search). A feature word is a word that appears in the same sentence as the user's keyword (having removed uninformative words, such as function words). After the user enters a keyword in the search field, a document list (containing all documents where the keyword appears), a sentence list (containing all sentences with the keyword) and a feature word list related to the keyword are generated (Zeng et al., 2013).

Feature words are found with the help of the sentence list. Answers to the quiz are found using GETA, which calculates a similarity score between a keyword and feature word by means of System for the Mechanical Analysis and Retrieval of Text (SMART). Thus, the IR retrieval function used by GETA includes ranking. The feature word with top 1 rank is the correct answer. Top 2-5 ranking words are the distracters. The questions are generated from the sentence list. First the questions that contain distracters are removed from the sentence list, then a random sentence is chosen as the basis for a question. Each question is given as a statement, where the correct answer (the top ranking feature word) is replaced by a set of empty parentheses (Zeng et al., 2013).

[30] Morphological parsing is used to identify and remove inflectional endings for a term, aiming to leave only the dictionary form (Manning et al., 2008).
[31] Generic Engine for Transposable Association, see: http://geta.ex.nii.ac.jp/e/.

By evaluation the developed system was found to be more suitable for information provision than for intelligence testing.

2.4.2.2 Automatic question generation and answer judging - Game for language learning

A prototype of a question and answer game for (English speaking) students learning Mandarin chinese was developed in 2009. A student reads isolated text statements, and answers questions about individual statements using speech. Statements are generated from lesson templates (based on the difficulty level chosen by the student), using a grammar (containing rules for generating strings in a specific language).

To generate a question, the first step is to produce a meaning representation called a linguistic frame from a generated statement, according to rules. A linguistic frame is a hierarchical representation which encodes both syntactic and semantic information. Analyzing and transforming natural language strings into a representation that is more easily handled by a computer is related to the NLP task of language understanding. The next step for generating a question is to transform the linguistic frame from the statement into a new linguistic frame for a question (modifying parts of the statement linguistic frame with elements that are more appropriate for a question). This is done by means of formal rules, pertaining to different types of statements. Finally, the linguistic frame is transformed into a natural language string of the question, which is presented to the user (Xu et al., 2009). This process of transforming a meaning representation (such as a linguistic frame) into a target string is related to the NLP task of language generation (Baptist & Seneff, 2000).

When a student answers a question using speech, their speech is transformed into a string. This answer string is then used to produce an answer linguistic frame. For answer checking, the elements from the question linguistic frame are refined into a set of compact key-value pairs (kv-frames). To find if the student answer is correct, the answer linguistic frame is also refined into a kv-frame, then augmented, and then compared to one or more kv-frames of the question (allowing for different ways of articulating the correct answer). This answer checking happens by means of a judgment algorithm. With evaluation the judgment algorithm was shown to be effective; having a false rejection rate for answers at 9% while having a false acceptance rate close to 0% (Xu et al., 2009).

2.4.2.3 Design of question answering system with automated question generation

A prototype of a question answering system, combined with a search engine, was developed in 2008. The prototype uses sentences within English documents as a source for generating English questions with corresponding answers. A user first enters one or more keywords into a web user interface. Thereafter a set of questions, with corresponding answers, are displayed to the user. The displayed questions are expected to be significant, since they contain the keyword(s) entered by the user (Kim & Kim, 2008).

A document collection of Wikipedia web pages about famous people was analyzed and processed to generate the pairs of questions and answers. Each individual question and answer pair was then stored in a relational database (MySQL). The following steps were conducted to generate and store the questions and answers: Sentence splitting, NER, sentence filtering, question generation, question filtering and question/answer indexing (Kim & Kim, 2008).

The IE sub-task of NER is used to identify named entities that can possibly be asked about in a sentence. For example, if an entity in a sentence is identified as a person, then it is possible to ask about the person. The questions generated can be divided into "sentence questions" and "entity questions". Sentence questions ask about a particular entity in a sentence, keeping the sentence structure and content. Only the answer is removed when converting the sentence into a question, being replaced with a Wh-interrogative word (Kim & Kim, 2008). For example, given the sentence "Margaret Mitchell wrote Gone with the Wind in 1936", a sentence question could be "Who wrote Gone with the Wind in 1936?". Here the answer is the entity that was replaced when converting the sentence into a question, Margaret Mitchell. Entity questions are also questions that ask about a particular entity in a sentence, but they use only parts of the sentence's structure and content, discarding other parts of it. Question filtering uses different rules; these rules use different POS information to remove sentences that can not be used for question generation. For instance, one of the rules state that the application should not ask questions about the subject of a sentence if it is a pronoun.

The researchers behind this prototype system found that meaningful questions were generated, but they note that automatically generated questions are also dependent on the NER tool which is in use. They also note that it would be interesting to have a measure for the user interest for the questions (Kim & Kim, 2008).

2.4.2.4 Semantic quiz

During the fall of 2013, two students at University of Bergen made a Semantic Web quiz application in the course INFO310 (Fjellanger & Armstrong, 2013). A dataset from LinkedMDB[32] was used, where the domain of interest is movies. This meant that movie information that had previously been extracted could be reused. The quiz was developed with Java, using Semantic Web technologies. Complex SPARQL queries are executed against the LinkedMDB dataset, which is stored in a local Jena[33] TDB RDF database, for efficiency. These queries are used to find movie information such as movie name, genre, director name and question difficulty level (based on gross movie income). The movie information is used for automatically generating questions and multiple choice answer alternatives.

[32] http://www.linkedmdb.org/
[33] http://jena.apache.org/

Chapter 3 - Development

This chapter focuses on the system that was developed during the master project. To put the development process into context, the chapter first presents the development method that was used. The section about development method also covers prototyping, as the quiz was developed as a prototype. Thereafter, the chapter presents requirements for the quiz. Next, to give a better understanding of the system, the quiz architecture is explained, covering its data sources and different components. Implementation information is also given for the components. The chapter then gives a brief presentation of the tools used during the master project. Finally, the development iterations are described, to give an overview of the development process.

3.1 Development method

The stages system development projects go through are very similar. They are: project initiation, requirements analysis, system design, construction, implementation, and maintenance (Weaver, 2004). The stages generally follow each other in succession, where one stage typically will be dependent on the stage in front of it (Cockburn, 2008). These stages collectively constitute the System Development Life Cycle (SDLC).

During the project initiation stage we decide on the type of system that we need. The requirements analysis stage involves deciding what our system should do and how it should perform. The design stage involves deciding on the system's user interface and functioning. During the construction stage the actual system is built and tested. The SDLC implementation stage involves providing the system to actual users. The maintenance stage involves actual usage and maintenance of the system. Maintenance may for instance be due to changes in the system's technical environment or due to change requests from users (Weaver, 2004).

3.1.1 The incremental model

Although all system development projects go through the SDLC, there are different models for going through the SDLC stages. Life cycle models provide a framework for planning a project. One life cycle model is the incremental model. In the incremental model a system is developed one working increment at a time, in separate phases that go successively through SDLC stages

(for instance requirement analysis, system design, construction and implementation). The development of an increment can be viewed as its own small project, where each increment adds functionality to the system developed. To accommodate new increments, it may at times be necessary to make changes to increments developed in earlier phases (Weaver, 2004). Sometimes, after developing a new increment, one may also find it necessary to make changes to other parts of the system, such as its previously defined requirements. Making these types of changes is made possible when combining the incremental model with iterative development, where iterative development involves setting aside time to revise and better different parts of the system (Cockburn, 2008). Given the possibilities to make changes in previous SDLC stages and development phases, we see that the incremental model combined with iterative development is fairly flexible.

Early in the planning of the master project it was decided that the elderly quiz would be developed in iterations, where each iteration would be focused on delivering added functionality. This was a good fit with the incremental model. As I did not know beforehand exactly how much functionality I would be able to implement in the elderly quiz, and I wanted the possibility to make changes to requirements and existing functionality based on evolving ideas and feedback from others, the incremental model was combined with iterative development. While life cycle models can guide us through the stages of SDLC, they do not prescribe any techniques or tools for developing a system (Weaver, 2004).

3.1.2 Prototyping

Prototypes can be used to answer questions about the design of an artifact. Letting stakeholders interact with a prototype may help them to get a clearer view of the idea, enabling reflection and discussions. High-fidelity prototypes may look like a finished system, or implement a number of the functions that one would expect to include in a finished system. They can be used for purposes such as testing the technical feasibility of an idea, or for usability testing and other types of evaluation. When we make a prototype we may choose to ignore some of the requirements that would need to be in place for a finished system, for instance security, or user interface development (Rogers, Sharp & Preece, 2011).

The elderly quiz was developed as a software-based high-fidelity prototype, evolving throughout iterations. During development the focus was mostly on implementing functional requirements

(see Section 3.2.2). Non-functional requirements which would have been important for a finished product were of less importance now. When initiating the work on the prototype it was made clear that the focus would be on the quality of the questions and answers, in preference over user interface development. Therefore the graphical user interface (GUI) for the elderly quiz is fairly simplistic. After being developed the prototype could be evaluated by means of usability testing.

3.2 Requirements

Since this project was suggested by my supervisor, Weiqin Chen, a meeting was held with her during project initiation, to get a better understanding of her vision. The first requirements for the quiz were defined shortly after the project initiation stage. Some more requirements were added and existing requirements were refined as the quiz prototype evolved (typically upon starting a new iteration, after receiving feedback related to the current state of the project). The elderly quiz requirements will be presented in this section with the help of a representative scenario, and by listing functional requirements and a non-functional requirement for the completed prototype.

3.2.1 Scenarios

Scenarios make it possible to give an informal description of human activities and tasks related to the use of a system. They let us explore aspects of the system such as its context, user needs and requirements. Scenarios do not necessarily describe the technical environment needed to achieve a task (Rogers et al., 2011). The following scenario describes a potential use of the elderly quiz, as it was envisioned early in the master project:

"An elderly married couple, both retired, want to spend some quality time together at home. Since they both like to play games they decide to play the elderly quiz. They gather around their preferred electronic device and start the quiz software. Before the questions appear on screen they make some choices about the quiz they want to play. They choose that they want to play competitively, and that they want to be asked about about (Norwegian) actors, musicians, movies and literature. After having made these configurations, they start a new quiz game. The couple take turns at answering questions, each time selecting the answer they think is correct

among four multiple choice alternatives. After a fixed number of questions the game ends, and one of them is declared as the winner."

While the scenario above gives an example of two players competing when playing the quiz, other scenarios could include a single-user playing the quiz and two players playing the quiz co-operatively. While the scenario above does not make any specific constraints about which device the quiz should be played on, it was developed for and tested on computers. While the quiz might have been web-based, the prototype was of a system that would have to be locally installed (Internet connectivity was also required).

3.2.2 Functional and non-functional requirements

Two types of requirements are functional and non-functional requirements. Functional requirements define what the product should do, while non-functional requirements describe factors such as performance and constraints for the product. Functional requirements cover areas such as data (what data the system will need to store and process), processing (what the system needs to do with the data, for example CRUD[34] operations) and algorithms. Performance-related non-functional requirements tend to apply to one or more functional requirements, providing information that will inform their design and construction. They include areas such as speed (response times), availability and usability. The non-functional requirements which are not performance-related are related to development or to the whole system (Weaver, 2004).

As noted in Section 3.2, initial requirements were gathered right after project initiation. Existing requirements were modified and new requirements were added throughout the project iterations. Below, the functional requirements and a non-functional requirement for the completed quiz system are listed. A distinction is made between the core quiz system (the components of the quiz that are used when users play the elderly quiz) and the extended quiz system (see Figure 3.1). The extended quiz system consists of the DBpedia data obtainer, the Scrape utility, and the Relation detector classifier module (RDC module). These shall run only once, before the quiz is used. The collective purposes of these components include obtaining data that can be used in the quiz, and distinguishing the obtained data. The results from these utilities can be distributed

[34] Create, read, update and delete.

with the core quiz system (for instance in an executable file), the utilities are not meant to be executed by the users of the quiz.

Functional requirements for the core quiz system

1. The core quiz system shall be able to generate Norwegian multiple choice factoid questions about Norwegian people (actors and musicians), movies and literature. Each question shall have four answer alternatives; one correct answer and three distracters.
2. The core quiz system shall generate questions and answer alternatives using RDF data stored in a RDF database.
3. The data used for question generation in the core quiz system shall be restricted to target elderly people. The people (actors and musicians) selected for question generation shall have a birth year in the range from 1890 to 1950. The movies and literature selected shall have a release or publication year in the range from 1940 to 1980.
4. The core quiz system shall allow users to select which question categories will be used for question generation when starting a new quiz. Users shall also be able to further restrict data available for question generation, by restricting the data available for the different question categories to narrower year ranges than the ranges stated in the requirement above (which are the default ranges).
5. The core quiz system shall allow users to choose quiz game type when starting a new game. The available game types shall be 1 player, 2 player co-operative and 2 player competition.
6. When a game is being played the core quiz system shall keep track of the number of players, the points of each player, and how many questions have been asked in the game. When 10 questions have been answered the game being played shall cease, possibly after announcing a winner (if two players).
7. The core quiz system shall generate a log for each game played. The log will contain general information about game type, and the selected question categories/years chosen for them (e.g. movies with release years from 1960 to 1980). For each question asked the log will also contain information about the following: Current player number, question number for current player, question asked, answer given by player, correct answer, and the current player points (after having answered the question).
8. The core quiz shall have a status indicator indicating if the quiz is online and ready to ask more questions.

Functional requirements for the extended quiz system

1. The DBpedia data obtainer shall obtain data from DBpedia; obtaining smaller databases from a local DBpedia RDF database, about Norwegian actors, musicians and movies, which can be used by the core quiz system.
2. The Scrape utility shall extract data about Norwegian authors, Norwegian literature and Norwegian last names from Wikipedia, by applying extraction rules on Wikipedia articles' HTML DOM parse trees. It shall generate gazetteers for authors, literary works and last names. A dataset of literary works shall also be made.
3. The RDC module shall distinguish literary works extracted by the Scrape utility, which have been referred to in unstructured text in selected Wikipedia articles, using patterns over sentences (tokens and strings). The patterns shall use rule-based NER to find literary works and their authors, based on gazetteers made by the Scrape utility.
4. The RDF generator shall insert RDF triples into a literature database, based on the dataset created by the Scrape utility. This database shall be used by the core quiz system. The RDF generator shall also be able to insert RDF data into the database for the RDC module that distinguishes that a literary work has been referred to in unstructured text in Wikipedia.

Non-functional requirements

1. The quiz GUI shall use a font size larger than average, to make text more readable for people with reduced vision.

3.3 Architecture and Implementation

This section will present the architecture of the quiz, and give implementation information for the quiz components. Figure 3.1 shows an overview of the quiz system and its components, making a distinction between the core quiz system and the extended quiz system, as explained in Section 3.2.2 (see Appendix A for a more detailed version of Figure 3.1, containing explanations about the individual components). This section first gives more detail about the data used for the quiz. It is followed by a brief presentation of the Jena framework. With these details covered, the different parts of the quiz, as they are shown in Figure 3.1, are explained. The RDF generator is used by both the Scrape utility and the RDC module. Its functionality will therefore be described in these two sections.

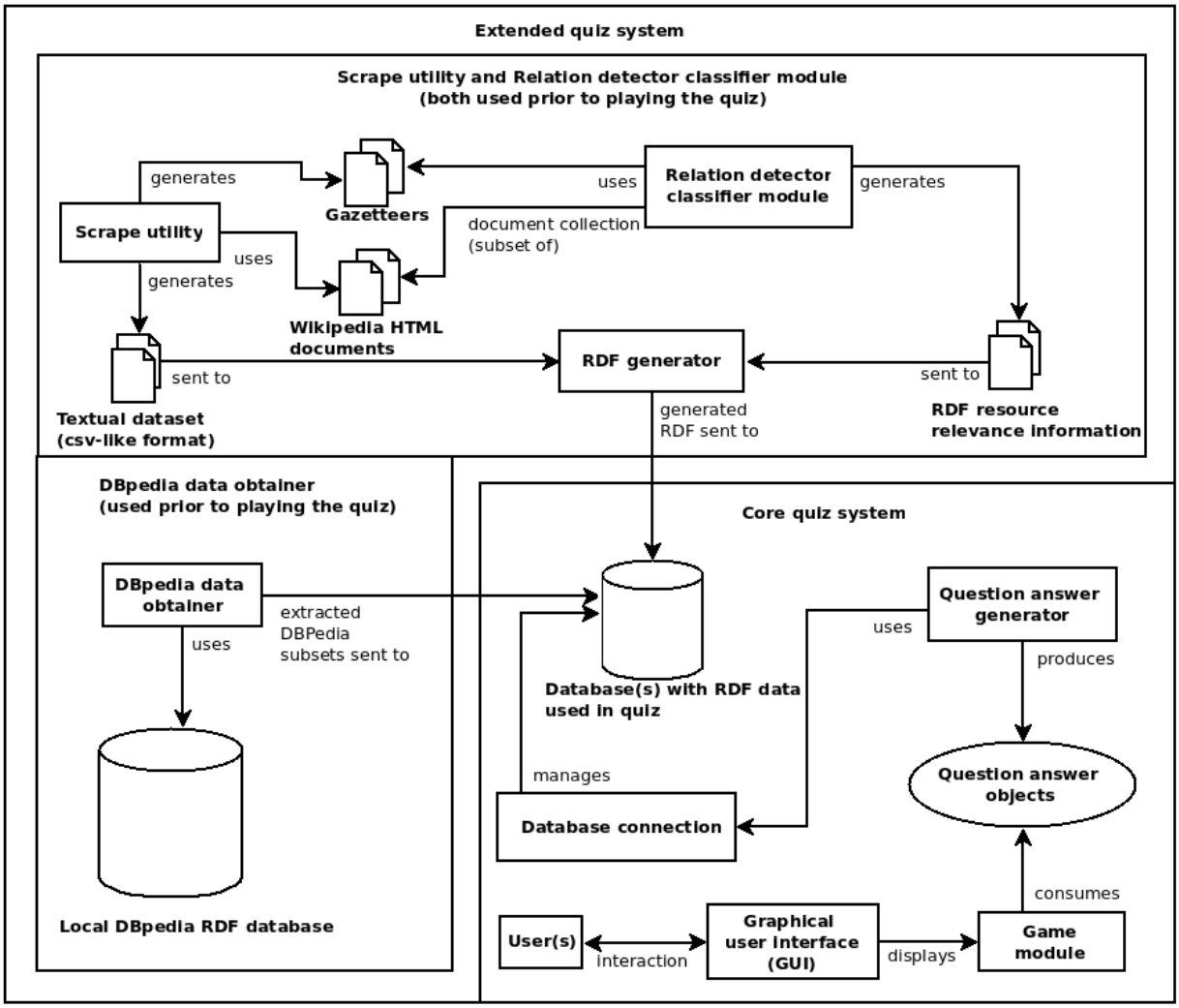


Figure 3.1: The quiz system architecture

3.3.1 Data source

As mentioned in Section 1.1, Wikipedia was used as the data source for the elderly quiz. Ideally I would also have liked to use blogs, archives, or other data sources containing information relevant for the elderly quiz. The reason why other sources were not used was two-fold: 1) Lack of functioning Norwegian tools for NER, which I was trying out to generate questions (more information in Section 3.3,9; paragraph: Process leading to RDC module) made me decide to use English language data sources. 2) There were problems finding other English language data sources than Wikipedia which covered topics that I found to be relevant for the elderly quiz (such as data sources covering Norwegian history and culture).

Most of the Wikipedia data came from DBpedia, but some information was also extracted upon constructing DOM parse trees, from a subset of Wikipedia HTML pages. The text below will give information about different types of data used in the quiz.

A local RDF database contains DBpedia data. It has numerous DBpedia datasets loaded, based on what seemed suitable for the elderly quiz during development. The datasets were loaded using tdbloader2 (see Section 3.4, paragraph: tdbloader2). Information for the datasets includes article title, article infobox information (for instance obtained through mapping-based infobox extraction, as described in Section 2.3.3), shortened article abstract (maximum 500 characters long), image URLs for first article picture (thumbnails and larger size), article inlink[35] counts (used to filter out some resources in the quiz based on the assumption that articles with fewer inlinks are more obscure) and YAGO type information (relating resources to YAGO categories). Information about all DBpedia datasets loaded into the local DBpedia RDF database can be found in Appendix B. The DBpedia data obtainer (see Section 3.3.7) obtains smaller databases that can be used for automatic question generation, from the local DBpedia database. Currently it can obtain Norwegian actor, musician and movie databases. More information about these databases can be found in Appendix C.

As mentioned above, some information used for quiz generation is extracted from Wikipedia HTML pages (articles), after applying different DOM extraction rules to DOM parse trees (see Section 2.2.2.1 for more information about this technique). A Web IE extractor, the Scrape utility (see Section 3.3.8), is responsible for the extraction. After extraction the data are post-processed and a dataset containing literature is created. This dataset is inserted as RDF data into a literature database. More information about the literature database can be found in Appendix C.

3.3.2 The Jena framework

Several quiz components, such as the Database connection and the DBpedia data obtainer, depend on the Jena framework. It will therefore be briefly presented before introducing the quiz components. The Jena framework can be used to create Semantic Web applications. The framework has capabilities for working with RDF data models. It also provides a native RDF database (TDB). RDF data models and TDB can be accessed with RDF APIs and SPARQL. In

[35] An inlink is a link directed from another Wikipedia article to the specified Wikipedia article.

addition, the framework supports ontologies, letting us add semantics to RDF data (Apache, 2015). Quiz data are stored in a Jena RDF data model. RDF data are accessed using RDF APIs. The data model is backed by a TDB RDF database, for persistency. DBpedia ontology information (in OWL format) is also loaded into the RDF database. Jena was used because is a well-known framework for Java which I had a positive experience of using previously.

3.3.3 Graphical user interface

To get an early idea of how users use and interact with the elderly quiz, the first component of the elderly quiz explained is the GUI. The quiz has a simple GUI made in Java swing. The font size used is larger than what is standard, adhering to non-functional requirement number 1 in Section 3.2.2. The two font sizes used are 18 point (used for most text) and 30 point (used for some headings). The font used is Arial. When opening the elderly quiz a window is first shown where the user(s) can configure initial quiz settings, before playing the quiz. Figure 3.2 shows this window. The window lets the user(s) choose the number of players for the quiz (1 player, 2 players co-operating, or 2 players competing). The user(s) can also select which question categories they want to be asked about. These question categories are actor (picture quiz), musician (picture quiz), movie, and literature. The actors and musicians that can be asked about must be born between 1890 and 1950, while the movies and literature must have been released or published between 1940 and 1980. The actor, movie and literature categories can be further restricted within these time periods (from decade to decade).

With the initial settings in place, a new quiz game can be started. This opens a new quiz GUI window which shows the current question and four answer alternatives (where one is the correct answer); in addition to displaying the score for the current player and a simple textual quiz status indicator. A game is set to end after 10 questions are answered (regardless of one or two players). Figure 3.3 shows an example of the quiz being played. The question category asked about in the figure is the actor category. All pictures in the picture quiz can be clicked on to open them for full-screen viewing.

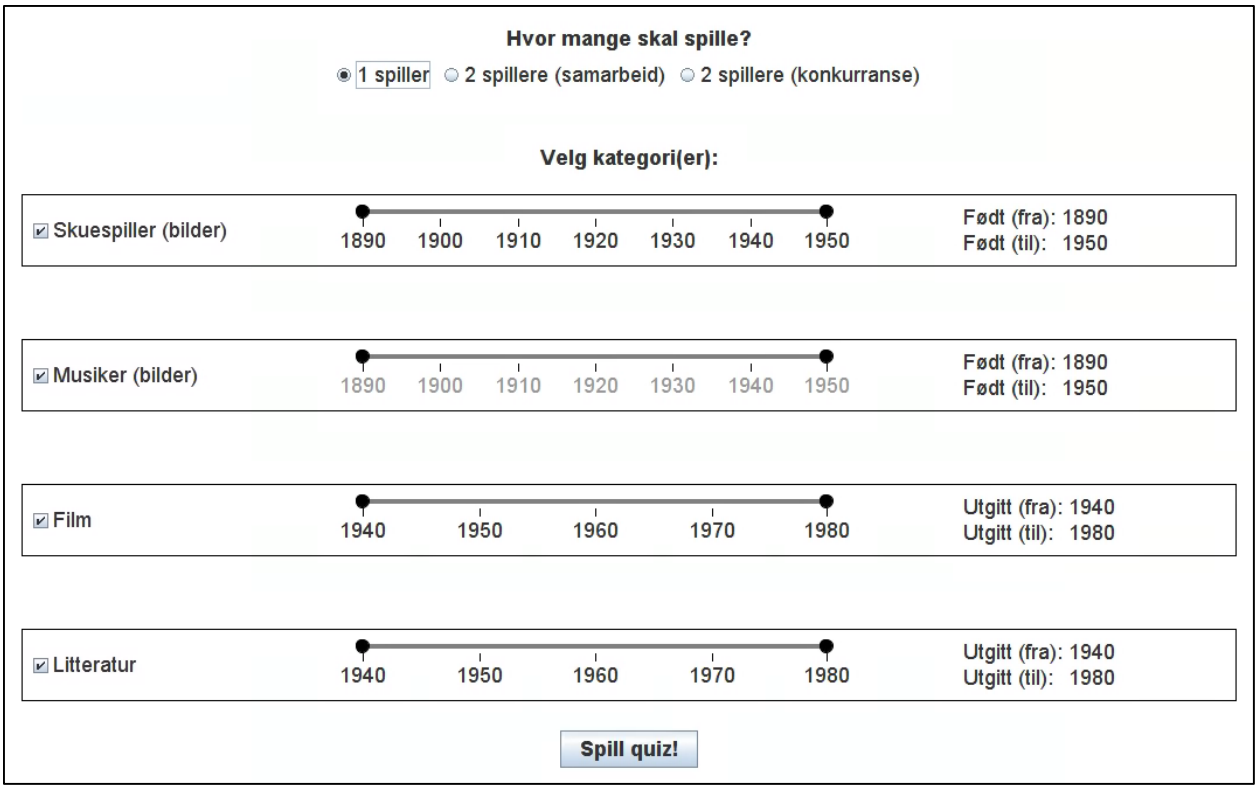


Figure 3.2: The initial configuration GUI window

Figure 3.3: The quiz GUI window

3.3.4 Question answer generator

The Question answer generator (QA generator) is responsible for generating question strings, selecting answers, and selecting distracters. The QA generator uses the following process: For each question with answer alternatives that should be generated, the QA generator asks the Database connection (see Section 3.3.6) for a random RDF resource, and maps triples where that RDF resource is the subject to a Java entity object (thus we have an object with facts about that resource). Entity objects differ according to resource entity type (person, movie or literary work). The QA generator uses the entity object to check if the resource is an appropriate answer, based on criteria for the selected resource's entity type. In addition, the resource can not be the answer for a question asked previously in a game. If the resource fits the constraints, it will be set as the answer; if not the resource will be discarded and the QA generator will ask for a new resource. This continues until an appropriate quiz answer is found.

After selecting an answer, the QA generator will ask the Database connection for new random RDF resources, mapping resource triples to Java entity objects, until it finds three resources that are suitable distracters for the given answer. Selection of suitable distracters is also based on criteria given the current answer's entity type. To avoid ambiguity and errors, for a distracter to be chosen it can not be same resource as the correct answer or one of the previously chosen distracters. The same distracter can, however, appear for multiple questions in the same game. When found, all four answer alternatives for the question, both the correct answer and distracters, are added to the same list of answer alternatives. The list is shuffled before being displayed in the quiz GUI, to ensure the answer alternatives are shown in random order. Finally, a question string is generated, based on the answer resource entity type, and information for that specific resource. The question string and answer alternatives are in Norwegian, but methods for obtaining data and creating the question string are generic, meaning it would be easy to use other languages as well. The QA generator stores the correct answer, distracters, question string and answer entity type in a Java question answer object (QA object). The Game module (see Section 3.3.5) can later interact with these QA objects.

To exemplify the steps of selecting answer alternatives and generating questions, we can look at how this is done for a person entity type, in a picture quiz: First the QA generator will select a random RDF resource that is a person, and map triples with that resource as a subject to a Java (person) object. If this person resource is an appropriate answer it must meet the following criteria: The resource Wikipedia article needs four or more inlinks; its gender must be known; its

birth year must be within the birth year boundaries of the initial configuration window (see Section 3.3.3); and it must contain an image URL that is reachable, referencing an image no larger than 7 megabyte (MB)[36], and not containing the substring "_og_" (English: "_and_"). As mentioned in Section 3.3.1, we assume that Wikipedia articles with fewer inlinks are more obscure. Thus we can also assume that the resources that are the topic of these articles are less well-known. The four or more inlinks criterion is implemented to remove some of the questions that are thought to be harder to answer, while still having enough available resources to ask questions about. Avoiding resources with image URLs with the substring "_og_" is a heuristic to filter out answer resources that have an image with more than one person. After ensuring that the selected RDF resource meets all criteria for an appropriate answer, and has not been asked about previously in the game, it is set as the answer.

To find suitable distracters for a person answer, we look for three unique person resources that are different from the answer, that fulfil the following criteria: Same gender as the answer, at least one profession[37] in common with the answer, and at least one inlink to their Wikipedia article page. With the one or more inlink criterion for distracters, distracter resources in the quiz do not need to be as well-known as answer resources. This means that the quiz has more distracter resources to choose from. We still filter out the resources we assume are the most obscure (having zero inlinks). Now we are ready to generate the question string. The string is based on the answer's entity type (here: person) and the answer's gender. If the answer resource's gender is male, the question will be: "Hvem er mannen på bildet?" (Who is the man in the picture?). If the answer gender is female, the question will be: "Hvem er kvinnen på bildet?" (Who is the woman in the picture?).

Generating questions and answer alternatives, where the answer entity type is movie or literary work, will follow a similar process as when generating questions and answer alternatives for the person entity type. The question string will, however, be different based on the entity type. The criteria for selecting an appropriate answer, suitable distracters, and which resource information is used in the resulting question string will also differ. An example question, for the movie "Tørres Snørtevold" is: "Hva heter komedie-filmen fra 1940, som ble regissert av Tancred Ibsen, hvor blant annet Alfred Maurstad, Folkman Schaanning og Anton Jessen medvirket?" (What is the name of the comedy movie from 1940, directed by Tancred Ibsen, starring among others

[36] The decision to avoid images above 7 MB was made to make the quiz more efficient.
[37] Available professions are actor and musician.

Alfred Maurstad, Folkman Schaanning and Anton Jessen?). An example question, for the literary work "Vindane" (The Winds) is: "Hva heter romanen som ble utgitt i 1952, og var forfattet av Tarjei Vesaas?" (What is the name of the novel that was published in 1952, and was authored by Tarjei Vesaas?).

3.3.5 Game module

For each game played the Game module retrieves new QA objects from the QA generator and stores old QA objects. It also keeps track of the game type, number of players, the current points of each player, how many questions have been asked in the current game, and how many questions will be asked in total. In accordance with functional requirement number 5 for the core quiz system (see Section 3.2.2), the available game types are 1 player, 2 player co-operative and 2 player competition. The Game module is also responsible for logging information about each game, in adherence to functional requirement number 7 for the core quiz system.

3.3.6 Database connection

The Database connection combines the smaller databases obtained by the DBpedia data obtainer, and the literature database made with the Scrape utility, into a quiz database that is used when playing a quiz. The Database connection is responsible for opening and closing all database connections. It also also loads and accesses the quiz database's RDF data model. In addition, it acts as an intermediate between the quiz database and the QA generator, enabling RDF resources to be retrievable by the QA generator for automatic question generation.

3.3.7 DBpedia data obtainer

As mentioned in Section 3.3.1, the DBpedia data obtainer obtains smaller databases, from a local DBpedia database. It should only be run at one time for each database one wants to obtain, prior to playing the quiz. Obtaining smaller databases from the local DBpedia database reduces the quiz storage requirements and makes it more efficient.

The DBpedia data obtainer uses Jena RDF APIs. For each run it obtains RDF triples and places them into a database, where the subjects of the RDF triples (the resources being described with

the triples) must have a specific entity type. The DBpedia data obtainer is run from the command line with the following syntax:
DataObtainer --dirname directory_name --type entity_type

Here "directory_name" should be replaced with the directory to store the database, and "entity_type" should be replaced with the entity type of described resources (subjects of RDF triples) that one wants to place in the database. The entity types that can currently be chosen are "actor", "musician" and "movie", but the DBpedia data obtainer could easily be extended to obtain databases for more entity types. For all three entity types, the DBpedia data obtainer will only obtain triples where the subject is a Norwegian resource. The subject resources must also fit certain criteria, depending on their entity type.

When run, the data obtainer first connects to the local DBpedia database. It will then load a list of RDF resources. Each entity type has its own list, which can be expanded or changed. For instance, actor currently has the following list:
[http://dbpedia.org/class/yago/NorwegianActors,
http://dbpedia.org/class/yago/NorwegianFilmActors,
http://dbpedia.org/class/yago/NorwegianTelevisionActors].

The DBpedia data obtainer searches for all resources in the local DBpedia database that are part of a RDF triple where the resource is the subject, and one (or more) of the resources in the entity type list is the object. The predicates in such triples express relationships between the subject and object, such as "is a" or "has topic"[38]. When all such resources are found, the data obtainer will keep all triples with those resources as a subject, not just the triples that relate them to the resources in the entity type list. The data obtainer then goes through the kept data, checking if the resources being described fit the criteria for their entity types. The criteria for person resources is that birth year must be between 1880 and 1955, and gender must be known[39]. The criterion for movie resources is that release year must be from 1940 to 1980. If one of the facts used as a criterion is missing for a resource, the DBpedia data obtainer can use heuristics to infer it. After going through the entire local DBpedia database, all triples for RDF

[38] “Is a” relationship is expressed with *rdf:type*, “has topic” relationship is expressed with *dct:subject*.
[39] Originally the thought was to ask questions about people born between 1880 and 1955, thus the DBpedia data obtainer (and the Scrape utility) uses this criterion; but later it was decided to only ask questions about people born from 1890 to 1950 (implementing this criterion for the question generator and GUI).

resources that match the entity type criteria are put in a smaller database in the directory specified when invoking the data obtainer.

The heuristics that can infer person birth year, person gender, and movie release year, were added to increase the amount of data in the databases. Some of the resources that would have been filtered out due to a lack of information, could now be added to the given database. The birth year heuristic uses the DBpedia short abstract for the resource one wants to decide the birth year for. It uses regular expressions to find the first occurrence of a year in the abstract, which will be set as the birth year for the resource. The gender heuristic works in two steps. First it looks for triples with the given resource as a subject, with an object containing either "NorwegianMale" or "NorwegianFemale". If a match is made, the appropriate gender is set for the resource. If a match is not made for the resource, its short abstract will be searched for the first word that is a gender-specific- pronoun or profession (for example actress). The gender will be set based on that word[40]. The release year heuristic is based on triple patterns. For instance, if the movie resource is the subject of an RDF triple that has the predicate *rdf:type*, and an object containing "yago", it is possible to check for a release year in the triple object by seeing if the object has a substring that matches the regular expression */(19\d{2})Films/.* This will match substrings containing "1900Films" to "1999Films"[41]. If the regular expression matches a substring in the object, then the year alone can be extracted for that resource using capture group 1, setting it as the release year for the specific movie resource (see Section 2.2.2.2 for information about capture groups).

Implementing the birth year heuristic made the obtained actor resources increase from 125 to 199, and the musician resources increase from 61 to 125. The gender heuristic could infer the gender for all relevant resources. The release year heuristic made the amount of obtained movies increase from 26 to 97.

One problem that might occur when obtaining a movie database with the DBpedia data obtainer is that for a few of the obtained movie resources, the Norwegian title is not available in DBpedia, only the English title. This could lead to some erroneous answer alternatives when asking

[40] Later I found out that there is a NLP DBpedia dataset that contains a predicate for person resources called GrammaticalGender. This dataset uses a simple heuristic that counts the amount of gender-specific pronouns. The gender of the person is inferred to be the gender that has the highest amount of gender-specific pronouns. As of now the quiz is still using the heuristic described in the text.
[41] To be more specific the regular expression could instead have matched substrings from “1940Films” to “1980Films”.

questions about movies. During development it was decided to leave movies with English-only titles in the dataset, as filtering them out would lead to filtering out movies with Norwegian titles from the quiz as well.

3.3.8 Scrape utility

Using selected Wikipedia articles, the Scrape utility extracts data about Norwegian authors (born from 1880 to 1955) and literary works written by those authors. Extraction of this type of data was chosen because there seemed to be limited information about it in DBpedia. The utility also extracts common Norwegian last names using Wikipedia articles (more information about all Wikipedia articles used by the Scrape utility can be found in Appendix D). Extraction is done from semi-structured HTML elements (mainly lists and tables), which are not covered by DBpedia extractors. The library jSoup[42] works as a HTML parser library. For each Wikipedia HTML file, jSoup constructs a DOM parse tree. Relevant elements in the DOM tree can be selected using a CSS[43]-like selector syntax. Data are extracted from element text node(s), according to different extraction rules which cover variations in HTML structure and content. Varying structure includes placement of relevant data within different HTML elements. Varying content includes aspects such as varying heading names covering similar concepts (for example a "Bibliography" heading name can also be referred to with the heading name "Works").

After extraction, the extracted data is post-processed. Post-processing is extra important for literary works, because the structure of literary work entries tend to vary, within, and between, author articles. Figure 3.4 shows an excerpt from the bibliography in the Wikipedia article about Jens Bjørneboe[44]. Here we see some of the types of information that can be available in literary work entries in a bibliography; for instance Norwegian title, English genre[45], English title (should not be used in a Norwegian quiz) and publication year. In addition to being extracted, genre information is used to filter out some literary works[46]. We also see some differences in the

[42] http://jsoup.org/
[43] Cascading Style Sheets.
[44] https://en.wikipedia.org/wiki/Jens_Bj%C3%B8rneboe
[45] English genre is translated programmatically to Norwegian genre, using a hand-written dictionary containing mappings from English to Norwegian for relevant terms. The initial plan was to use the Google Translate API for translations, but I decided against it as using the API required payment of a usage fee.
[46] The assumption is that literature with these genres are not appropriate for the elderly quiz. Examples of genres that cause literary works to be filtered out are songbook, radio play and travel writing.

structure of the literary work entries; in that the entry for "Jonas" only contains one title, while the other entries contain two (both English and Norwegian title).

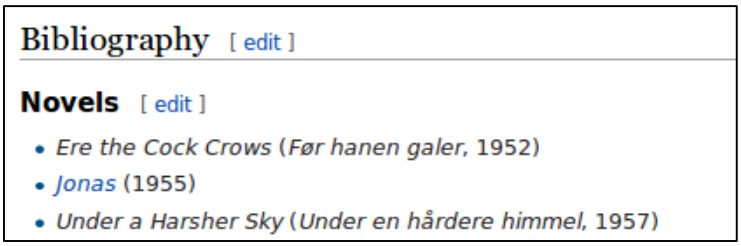


Figure 3.4: Excerpt from Wikipedia article about Jens Bjørneboe

For authors and their literary works, a comma-separated values (CSV)-like[47] file (dataset) is created after post-processing, which can be used with the RDF generator to import RDF information about authors and literary works into a literature database (thus semantically lifting the data). For each literary work, the dataset contains literary work title, author, publication year, and genre (if applicable). Three separate gazetteer files are also made from extracted data, containing authors, literary works and last names. The gazetteers can be used for rule-based NER in the RDC module (more information about the gazetteers in Appendix D). While the Scrape utility served its purpose at the time of development, changes in the HTML pages used for extraction with such a tool might lead to frequent updates of extraction rules; something that can be time consuming.

3.3.9 Relation detector classifier module

The Relation detector classifier module (RDC module), as its name implies, is responsible for the IE sub-task of relation detection and classification. It uses patterns over sentences (tokens and strings), to find binary relationships between authors and literary works, in unstructured text in a document collection. These patterns include named entity types. The patterns are defined and matched against using the generic TokensRegex framework[48], thereafter selected parts of sentences that match the patterns are extracted. There are five patterns in total; they each match against one, two, or three sentences. As we see in Figure 3.1, the RDC module is closely tied to the Scrape utility. Because the RDC module uses gazetteers made by the Scrape utility, it must be executed after it.

[47] Comma was not a good separator for these data, therefore "::" was used as a separator instead.
[48] http://nlp.stanford.edu/software/tokensregex.html

Currently Wikipedia author web pages, that were first used with the Scrape utility (see Appendix D), are used as the RDC module's document collection. While the Scrape utility uses DOM patterns for scraping from semi-structured HTML elements, the RDC module works with unstructured text in the documents of the document collection. Because TokensRegex patterns are generic, they could also be expected to work with documents from other data sources; but this was not thoroughly tested, due to limited access to relevant documents, and project time constraints. When a relationship in unstructured text is found between an author and literary work that are included together in the literature database, a new RDF triple is added about the literary work in the literature database, indicating that it is referred to in unstructured text (currently in the form "subject :referredToInWikiArticle true"). Creating and adding the triple is done by the RDF generator, which uses information it retrieves from the RDC module.

The reason for adding information about a title being referred to in unstructured text in Wikipedia, is that in many Wikipedia articles only the most significant literary works of an author, and the debut of the author, are written about in unstructured text. My assumption was therefore that finding textual relations through the unstructured text of an author's Wikipedia article could be used to make the quiz easier and more relevant for the users, because the literary works included in these relations seem to have a higher likelihood of being known. For instance, one could have different difficulty levels, where an easier quiz would have more questions about the titles that were referred to in unstructured text than a more difficult quiz; or only questions could be asked where the correct answer was referred to in unstructured text (using other titles, not referred to in unstructured text, as distracters).

Before unstructured text can be matched with TokensRegex patterns, the text must be run through a pipeline. This is done using a "pipeline annotation framework"; Stanford CoreNLP[49] (Manning, et al., 2014). Documents in the document collection are run through the pipeline, one at a time. Before entering the pipeline the documents are cleaned programmatically (removing HTML markup, headings, tables, and so on), ensuring that only complete sentences are left. CoreNLP contains a collection of NLP tools, which can be executed sequentially in the pipeline. When given an input document, each tool will process the document (directly, or indirectly using results from tools run earlier in the pipeline), creating a set of document annotations. Annotations are available to tools later in the pipeline, and for use in the application that uses the

[49] Version 3.5.1 of CoreNLP was used; http://nlp.stanford.edu/software/corenlp.shtml.

CoreNLP framework. The pipeline used in the RDC module consists of the following tools: Tokenizer, sentence splitter, and rule-based named entity recognizer (RegexNER[50]).

The tokenizer divides unstructured text into a sequence of tokens, where a token more or less corresponds to a word or a punctuation. The sentence splitter breaks the text into individual sentences, using the annotations made by the tokenizer. RegexNER produces rule-based NER annotations for individual tokens, using the annotations made by the tokenizer and sentence splitter. While generic named entity systems tend to annotate a limited number of broader-ranging classes (as mentioned in Section 2.2.1); RegexNER allows for annotation of user-chosen, more specialized named entities.

Gazetteers with terms for specific entity types are provided to RegexNER, acting as a NER lexicon. As mentioned earlier, these gazetteers were created automatically by the Scrape utility and contain authors, literary works and last names. RegexNER is responsible for checking if a term is within one of the gazetteers; thus it is responsible for the NER grammars. The classification rules in RegexNER are simply to annotate each term found in a gazetteer with the given gazetteer entity type. This is possible because the TokensRegex patterns will later check in which context the annotated named entity occurs, thus wrongly annotated entity types (for example due to ambiguity) are more likely to be filtered out later.

After the tools in the NLP pipeline have annotated a document, we can find text in the document that matches against the TokensRegex patterns. TokensRegex extends the regular expression language to match sequences of tokens. In this way we can match not only strings (as sequences of tokens), but also token-level features, such as POS label and named entity type. The syntax for matching strings of text is similar to the regular expression syntax seen earlier (in Section 2.2.2.2). Additional syntax is provided to match tokens and token-level features (Chang & Manning, 2014).

As mentioned earlier, the RDC module contains five patterns that find binary relationships between authors and literary works in unstructured text from selected Wikipedia articles. All of the patterns look for relationships in text between authors (full names or last names) and (their) award-winning literary works; since these literary works seem to be mentioned more often in Wikipedia articles than other literary works. The patterns work as intended; with the help of the

[50] http://nlp.stanford.edu/software/regexner.html

RDF generator additional RDF triples are added about titles in the TDB literature database, signifying that the titles have been referred to in unstructured text (from Wikipedia). Even though the patterns work as intended, there is a need for more patterns before it is possible to use the pattern results specifically to make the quiz easier or more relevant. With the present document collection, five literary work titles in total are matched as award winning. New patterns could encompass more unstructured text contexts for extraction, for instance best selling titles written by an author, and author debut titles. Adding more relevant documents to the document collection could also yield more results.

Process leading to the RDC module

When generating questions and answers from the datasets created with the DBpedia data obtainer, I had mostly worked with Semantic Web technologies. To add variety and technical challenge to the project I now wanted to start with question generation that was supported by NLP. After some research I wanted to try a question generation approach inspired by the one used by Kim and Kim (2008). As explained in Section 2.4.2.3, the approach converts English sentences into questions, where the questions keep various degrees of sentence structure and content. NER identifies named entities that can possibly be asked about in a sentence, and POS information filters out sentences that can not be used for question generation.

My idea was to use an approach similar to that of Kim and Kim (2008) on unstructured Norwegian text, to see if it would be good enough to generate Norwegian questions for the elderly quiz. Through research I found that a Norwegian grammatical tagger, the Oslo-Bergen tagger[51], had support for both POS tagging and NER (Johannessen, Hagen, Lynum & Nøklestad, 2012). I tried to test the Oslo-Bergen tagger on Norwegian text, but NER information was not shown, only POS information. E-mail correspondence with one of the Oslo-Bergen tagger developers[52] revealed that the Oslo-Bergen tagger NER module had been removed during an upgrade, and that there was seemingly no working Norwegian NER tool at the time[53]. Therefore, I decided to try to generate Norwegian questions using NER and POS tagging on unstructured English text (regarding Norwegian topics) instead of on Norwegian text. I spent some time reading about tools that would possibly suit this purpose. This led me to start testing

[51] Version number for Oslo-Bergen tagger N/A (downloaded November, 2014); http://tekstlab.uio.no/obt-ny/english/index.html.
[52] Anders Nøklestad at UiO.
[53] Some researchers at UiB and UiO had shown an interest in making a new executable NER tool, but at the time of development that was still work in progress.

the Stanford Named Entity Recognizer (2014) tool, and the GATE Developer Environment tool (with the IE system ANNIE)[54]. For testing I used some Wikipedia articles about Norwegian actors. The Stanford NER tool uses sequence labeling NER classification based on the CRF algorithm[55]. It was distributed with three CRF classifiers that could only be used separately. The classifiers could each classify entities used in generic named entity systems, such as person, location and organization. The NER tool was tested with one classifier at a time. The GATE Developer Environment tool uses rule-based NER.

A problem I noticed with both the GATE and Stanford NER tool was that when attempting to recognize Norwegian person names with NER, many times the tools would not recognize person entities at all, and other times they could not correctly identify boundaries for names that spanned consecutive tokens. Since the tools were not satisfactory for my use I tested using the GATE API programmatically for NER and POS, but I found its API to be poorly documented and its language for defining NER extraction rules would require a steep learning curve. It was therefore decided to try to do some experimentation with the Stanford CoreNLP Java library, which included tools for both statistical NER based on the CRF algorithm (CoreNLP NER) and rule-based NER (RegexNER). At this point I tested only CoreNLP NER, and found that it worked much better for Norwegian name recognition than using the Stanford NER tool. The reason was likely that CoreNLP NER used a combination of the three sequence-labeling CRF classifiers that could only be run separately in the Stanford NER tool (Manning et al., 2014; Stanford Named Entity Recognizer, 2014).

Stanford CoreNLP also had tools for POS classification and coreference resolution. The results from the coreference resolution contained information about all the text in a document that refers to the same named entity (for example the same person), something that might be useful when generating questions. Ultimately I decided against using the coreference tool from the Stanford Core NLP library, as it I did not find it accurate enough to use in the quiz. While I had wanted to convert selected sentences from English documents into Norwegian questions with the help of NER and POS information, in a way that was inspired by the Kim and Kim (2008) approach mentioned earlier, I found that this could not be accomplished when using English (unstructured) text documents. Such an approach would only be suitable for generating English questions.

[54] Version 8.0 of GATE was used; https://gate.ac.uk/download/.
[55] The NER tool also includes capabilities for rule-based matching against numeric and time patterns, but these capabilities were not used during the master project.

I now searched for a different method for generating Norwegian questions. I decided to look for a solution which searched for patterns over strings in English unstructured text. The idea was that information could be extracted from sentences adhering to the patterns, more specifically entities and their relationships. Thus the solution I searched for was a technique of relation detection and classification, followed by extraction of detected relational triples. In addition to strings, the patterns should include named entity types and possibly POS tags; where POS tags for instance could be used to match words that are determiners, or identify personal pronouns. The relational triples extracted could be represented as RDF triples which could be inserted into a database, thereby enabling further question generation based on RDF triples. As noted in Section 2.2.2.2, there is generally a trade-off between pattern generality and pattern precision. For the elderly quiz I thought that precision was of more importance, thus the patterns would need to be specific.

The search for an appropriate pattern matching solution led me to TokensRegex. TokensRegex patterns could match all CoreNLP NER annotations for a given unstructured text, both the ones from CoreNLP NER (sequence labeling with three combined classifiers) and RegexNER; it could also match POS tags. To test, I set up a CoreNLP pipeline, defined a few TokensRegex patterns and extracted binary relations (relational triples) from a few Wikipedia articles about Norwegian famous people. The NER tool used was then CoreNLP NER. I also tested matching against POS tags, but found them to be too general for the use-case (for instance I wanted to match against a few determiners, and a few personal pronouns, but not all of them). Using TokensRegex for defining patterns seemed to work well.

To add more information to the quiz I decided to use TokensRegex patterns to identify authors and their literary works. CoreNLP NER could only recognize more generic entities (such as person, location, organization), unless training a new CRF classifier with a large annotated corpus (something I did not have access to). For this reason, I decided to use RegexNER for identification of literary works and authors. The Scrape utility was created to scrape gazetteers for RegexNER, but because of the data it had access to, it was also used to create a literature dataset.

When the gazetteers and the literature database were created, work began on defining patterns. It was now decided that the patterns should distinguish relevant triples in the literature database, as noted earlier. The five patterns that find relationships between authors and their award-

winning literary works were created. Four of the patterns match a limited set of personal pronouns ("he", "she") that refers to an author (for example if the author name, or author last name, is used in the start of a sentence; and the next sentence starts with such a personal pronoun). In this way the four patterns give a limited, but seemingly more accurate, alternative to using coreference resolution.

3.4 Tools

This section will give an overview of tools used during the programming project.

Ubuntu

The operating system (OS) used during the programming project was the popular Linux distribution Ubuntu. I chose Ubuntu because it is my preferred OS for development. Version 14.04 was used, which is a long term release. This meant that it would not be necessary to upgrade to a newer version of the OS during the master project.

Java

The elderly quiz was developed in Java. The main reason why I chose Java was its support for Semantic Web and NLP frameworks. It is also the programming language that I have the most experience with, it is platform independent, and extensive API documentation is available.

IntelliJ IDEA

IntelliJ IDEA[56] was chosen as the integrated development environment (IDE) for development of the quiz. The reason for the choice was that IntelliJ had good integration with the tools git and maven.

Git/Bitbucket

Git[57] was used as the version control system for the quiz. Bitbucket[58] was chosen as the Web-based service to host the Git repository, as it allowed free private access to the repository.

[56] https://www.jetbrains.com/idea/
[57] https://git-scm.com/
[58] https://bitbucket.org/

Maven

Maven[59] was used to manage the elderly quiz dependencies, making it possible to specify them all in one file. This was much more convenient than doing manual imports in IntelliJ. Maven also automated the process of packaging the elderly quiz into an executable .jar file.

tdbloader2

The unix-based command line application tdbloader2[60] was used to bulk-load RDF data from different DBpedia datasets into a local DBpedia database. Initially I tried to populate such a database programmatically, using Jena libraries. The latter method was very inefficient, taking days to complete. In contrast, tdbloader2 did the loading within hours.

Wget

Wget[61] is a command line tool with capabilities for retrieval of HTML files. It was used to download lists of HTML files from Wikipedia, which would later be used for IE with the Scrape utility and as a document collection for the RDC module.

3.5 The development process

This section gives an overview of the activities in the iterations that were conducted when developing the elderly quiz. Thus it gives a high-level view of the development process, including the sequence followed when developing different parts of the prototype.

Iteration #	Length	Activities
1	3 weeks	Setup of PC and programming environment. Research of possible quiz data sources. Exploration of DBpedia datasets using SPARQL endpoint[62]. Initial bulk-load of relevant DBPedia datasets into local DBpedia TDB database, using tdbloader2. Creation of actor database, containing filtered data from local DBpedia database.

[59] https://maven.apache.org/
[60] https://jena.apache.org/documentation/tdb/commands.html
[61] https://www.gnu.org/software/wget/
[62] http://dbpedia.org/sparql

2	2 weeks	Creation of Database connection, QA generator, Game module, and GUI (core quiz system). Made heuristics that inferred birth year and gender for person RDF resources (as described in Section 3.3.7). Had working actor picture quiz at the end of the iteration.
3	1 week	Created the DBpedia data obtainer, for obtaining smaller databases from the local DBpedia database. The functionality for obtaining an actor database was refactored into the data obtainer, and capabilities for obtaining a musician database were added. After some changes to the core quiz system, the picture quiz could now also ask questions about musicians.
4	2 weeks	Work on movie quiz. Added new datasets to the local DBpedia database, as there was not enough movie information. Expanded on the DBpedia data obtainer to obtain a movie database. Made heuristic to infer movie release year (as described in Section 3.3.7). After some changes to the core quiz system, the quiz could also ask questions about movies.
5	8 weeks	Research of NLP supported automatic question generation. Started working on what would later become the RDC module.
6	3 weeks	Initially wanted to add new information to the quiz about authors and literary works, through extraction of relations between authors and literary works found with TokensRegex patterns. For this approach I created the Scrape utility to scrape data for NER gazetteers so that NER information could be added to the patterns. Extraction and post-processing of data scraped for gazetteers turned out to be time-consuming. The Scrape utility now had access to a lot of other data than just literary work titles and authors, therefore I decided to add all literary work information as RDF data; leading to the creation of the RDF generator and the literature dataset.

7	1.5 weeks	Since data about authors and their literary works had been stored in a literature database using the Scrape utility, I decided to use relational detection and classification to find and add information about literary works in the database that could be assumed to have a higher likelihood of being known. I created patterns identifying award-winning literary works written by Norwegian authors, as noted in Section 3.3.9.
8	4 weeks	This iteration was conducted during planning of the evaluation and execution of the pilot test (see Section 4.4). Added literary works to quiz after some changes to the core quiz system. Improved on GUI and implemented threads for efficiency. Prepared quiz for testing by implementing functionality such as logging (saving usability testing quiz results) and status indicator. Increased font size and enabled viewing of full-screen pictures in picture quiz.

Chapter 4 - Evaluation

The elderly quiz was evaluated by means of usability testing. This chapter will present the evaluation. It first gives an overview of usability testing in general; which is followed by information about goals, participants and procedure for the evaluation. Then the analysis of collected data is explained, and the results from the usability testing are presented. Finally, findings from the usability testing will be discussed, and some improvements will be suggested.

4.1 Usability testing in general

As stated by Oates (2006), evaluation can lead to increased insights about a product, and may suggest areas of the product that need improvement. Different criteria of a product can be evaluated; which criterion to evaluate depends on what aspects of a product one wishes to gain insight about. One possible criterion is usability. According to Rogers et al. (2011, p. 19), usability refers to "ensuring that interactive products are easy to learn, effective to use, and enjoyable from the user's perspective". Usability can be assessed through usability testing, which involves letting people that are representative of the target audience use or review a product, to evaluate the degree to which the product meets specific usability measures (Rubin & Chisnell, 2008, p. 21). Such usability measures can be a part of an operational definition of usability, made specifically for testing a product, and will be the focus of evaluating usability for the elderly quiz. Section 4.2 will describe usability measures.

To assess the degree to which a product meets specific usability measures, data must be collected from users. Collected data can be divided into two major categories: preference data and performance data. The former represents measures of participant opinion or thought process, while the latter represents measures of participant behavior. Both preference data and performance data can be presented and analyzed qualitatively or quantitatively, depending on what is the intent of testing (Rubin & Chisnell, 2008). Qualitative data is data which is not inherently numeric. Quantitative data is numeric data, either inherently or through transformation of qualitative data (Rogers et al., 2011).

The main method for collecting data in usability testing is to observe users when they use or review a specific representation of a product, such as a prototype. To capture additional

preference data during this type of observation, it may be appropriate to use the "thinking aloud" technique. This technique involves asking a participant to think aloud while using or reviewing a product, thus providing a running commentary of their thought process. Frustrations and points of confusions from this commentary can be especially helpful for identifying usability issues. To gain a better understanding of data that is collected during the observation, it may also be appropriate for the person responsible for conducting the test (the test moderator) to ask unscripted questions, depending on the design of the test (Rubin & Chisnell, 2008). To complement data from the observation, other methods can also be used, such as interviews and questionnaires. Employing two or more data collection methods to investigate a product from a broader perspective is known as methodological triangulation. This is generally viewed as a good practice (Rogers et al., 2011).

Usability tests vary in their formality. Formal tests could be set up as classical experiments, with complicated designs and large sample sizes. Such tests could aim to be statistically significant. Naturally, this type of testing is time- and resource-intensive. The focus of this chapter will be on less formal tests, which do not aim to be statistically significant. Less formal tests do not necessarily require a large number of participants, and have a simpler design than more formal tests, yet they still have rigorousness at their core. These types of tests are often suited for obtaining both preference and performance data, related to the degree to which the product meets specific usability measures. Specific test objectives (which can be formulated as questions that the usability test aims to answer) guide the data collection. Presentation and analysis of the data provide a basis for offering recommendations on how to improve upon product design (Rubin & Chisnell, 2008).

Usability tests take place in a test environment, where participants are being observed while they use or review a product. The traditional approach is to conduct sessions in a controlled environment (usability lab). In addition to allowing the test moderator to control how the test proceeds, such an environment is suitable for minimizing distractions and outside influences. In recent times it has become increasingly more common to conduct usability testing in more natural settings, due to the affordable price of portable equipment (such as laptop computers, audio recorder and video camera). This makes it possible to test a product in a context where it is meant to be used, such as a home or work environment. While the test moderator has less control over outside influences in more natural settings, these settings are less artificial and can be of more convenience for participants (Rogers et al., 2011).

It is possible to conduct iterative testing to inform the design of a product, from its early beginnings, until it is nearing completion. How usability testing is conducted will vary according to factors such as test objectives, and the completeness of the tested product. These factors will influence aspects of the test, such as how much interaction there is between test moderator and participant, and what the ratio is between collection of preference versus performance data, when a participant is being observed while using or reviewing a product. When usability testing is conducted early or midway in the process of developing a product, after the high-level design has been decided on, it is fairly common to ask participants to perform realistic tasks for the product. This type of usability test is also known as an assessment test. Lower-level functionality of the product is evaluated, to establish the effectiveness of implemented concepts, and to identify usability issues. Due to the focus on performing tasks, there is usually less interaction between test moderator and participant during observation. The assessment test will always generate performance data, and quantitative measures (for instance related to task performance) will be collected (Rubin & Chisnell, 2008).

Thus far, this section has emphasized the method of observation while performing usability tests. As mentioned earlier in this section, it is possible to complement data collected through observation with data obtained from other methods, such as interviews and questionnaires. In the following, interviews will be presented.

4.1.1 Interview

An interview is a "particular kind of conversation between people" (Oates, 2006, p. 186). Interviews can be divided into three types: structured, unstructured, and semi-structured. Structured interviews use the same standardized questions, for each interviewee. Interviewees answer questions, but there is no room for discussion between interviewer and interviewee. Possible answers are often pre-coded. In a sense, a structured interview is much like an oral version of a questionnaire. Unstructured interviews are based on introducing a topic to an interviewee (possibly from a list of topics). The interviewee should explore their ideas and talk freely about the topic, going in depth. The interviewer asks open questions, avoiding to constrain the type of answers the interviewee can give. During the interview, the interviewer should obtain answers to questions that were decided in advance. When needed, the interviewer should also follow new lines of examination (Oates, 2006; Rogers et al., 2011).

As the name suggests, semi-structured interviews are a hybrid of the two previous techniques. Such interviews let participants speak their mind, making them suitable for in-depth discovery. In a semi-structured interview there is a list of themes to be covered and questions you want answered, but the order of the questions can be changed depending on the conversational flow. If a question is not found to be relevant it can be removed, and additional questions can be asked if issues arise that were previously unknown. Due to their exploratory nature, semi-structured and unstructured interviews are unsuitable for drawing conclusions about entire user populations (Oates, 2006).

Interviews can be conducted both before and after participants use or review a product. An interview before a test (pre-test interview) can be used to establish background information about a participant, such as their level of expertise which may be relevant to product usage. The pre-test interview can also be used to find out more about participant attitudes and first impressions about a product, prior to using it. Additionally, the pre-test interview may be used to learn more about if a participant sees the value of a product. A post-test interview can be conducted as a debriefing. The aim of a debriefing is "exploring and reviewing the participant's actions during the performance portion of a usability test" (Rubin & Chisnell, 2008, p. 229). A debriefing is typically conducted as a semi-structured interview. Debriefings can explicate why problems occurred during a usability test, and how they might be fixed.

In some cases, it can be interesting to compare a participant's answer on the pre-test interview against their answer on the post-test interview. This might for example be to see if what they think about the value of a product has changed after trying out the product. If two or more people are participating in a usability test together, it may also be suitable to interview the participants together. In this case, emphasis should be put on discussion between the participants when answering questions. It is important that all participants speak their mind during such interviews (Rubin & Chisnell, 2008).

4.2 Goals

As mentioned in Section 4.1, we can evaluate the usability of a product with the help of usability measures. The goals of this evaluation were to get an indication of the usability of the quiz, according to specific measures which will be presented at the end of this section. Rubin and

Chrisnell (2008) list the following usability measures: Usefulness, efficiency, effectiveness, learnability, satisfaction, and accessibility.

Usefulness is an indicator that the system lets the user achieve his or her goals, and is an assessment of the user's willingness to use the product at all. Efficiency is how fast the user can complete their goal with the product. Effectiveness refers to the extent to which the product behaves in the way it is expected to and the ease to which the participants can use it to do what they intend. Learnability is related to how well a user can operate a product after a given amount of time and (if applicable) training. Satisfaction refers to the user's perceptions, feelings, and opinions of the product. Accessibility is a sibling of usability. In this context it is about making products usable specifically for people with disabilities (Rubin & Chisnell, 2008).

Since the test moderator interacted with the prototype for participants during the test (as will be explained in Section 4.4.2), it was not found appropriate to assess the degree to which the elderly quiz met efficiency, learnability and accessibility. To evaluate the usability of the quiz, three relevant usability measures could be assessed; usefulness, effectiveness, and satisfaction.

The goals of the usability testing were thus to:

- Get an indication of the usefulness of the elderly quiz for Norwegian elderly people (the target audience), with the help of participants.
- Get an indication of the effectiveness of the elderly quiz, by letting the participants play the quiz themselves (while the test moderator is controlling the interaction with the quiz and acting as a "quiz master").
- Get an indication of the satisfaction with the elderly quiz, based on factors such as the participants' perceptions, feelings, and opinions.

4.3 Participants

As stated in Section 4.1, one should use target users when conducting evaluations with usability testing. The target audience for the elderly quiz are Norwegian elderly people above the age of 60. Rubin and Chisnell (2008) state that the best sources for recruiting participants over the age of 50 to a usability study is to use personal networks, as this population may be more cautious and sceptical than people of a younger age.

Four participants were recruited for testing, birth years ranging from 1939 to 1952. The participants had varying experience with playing quizzes, and different levels of affinity towards this type of activity. Two participants (P1 and P2) are a couple. One participant (P4) had a visual impairment, although she had normal eyesight for her age. Two participants were recruited by me through my personal network, the other two were recruited by Weiqin Chen. When selecting the amount of participants for a usability test the general guideline is that "more is better". However, for a less formal usability test, research has shown that having four to five participants will expose about 80 percent of the product's usability deficiencies, for a given type of audience (Rubin & Chisnell, 2008). This would indicate that a substantial amount of usability deficiencies in the elderly quiz (for the specific usability measures that were the focus of the tests) could be detected during testing.

Three usability sessions were conducted. The first session, with two participants, was conducted by me. The next two sessions, each having one participant, were conducted by Weiqin Chen. Table 4.1 below contains basic information for the individual participants, including which usability session they attended.

Identifier	**Gender**	**Birth year**	**Test #**
P1	Female	1939	1
P2	Male	1942	1
P3	Male	1948	2
P4	Female	1952	3

Table 4.1: Participants

4.4 Procedure

The usability testing was aimed to get an indication of the usability of the quiz, according to the goals in Section 4.2. I planned the test procedure, and documented it in a test plan adapted from Rubin and Chisnell (2008). To identify and correct potential weaknesses with the procedure, a pilot test was conducted. After the pilot test, the procedure was finalized. Some minor changes were also made to the elderly quiz, after which all development was suspended until the evaluation period had reached its end.

4.4.1 Data collection

Data collection methods were selected that would be appropriate for the usability testing goals. In addition to observation during the test of the quiz, it was decided to conduct interviews (pre-test and debriefing interview). When two participants participated in the same usability session, they performed the test and did the interviews together. The usability test was designed as an assessment test, where participants were asked to perform representative tasks (that is, to actually play the quiz). To collect data related to the thought process of participants when they were performing tasks, the thinking aloud technique, or its variation "co-discovery", was employed. The co-discovery technique involves having two participants communicate while performing a task together, and was used when two people participated in the same usability session. While a participant was performing tasks, the test moderator could also ask unscripted follow-up questions regarding the thought process of a participant, for instance if the person did not remember to think aloud. Both the pre-test interview and the debriefing interview were semi-structured. To guide the data collection, some test objectives (in the form of questions) were defined. These objectives were used because answering these specific questions would enable us to get an indication of the usability of the quiz, based on the goals of the evaluation (see Section 4.2).

- Objective 1: Is a computer-based quiz that asks questions about people that were well-known and works (such as movies and literature) that were created during their past (i.e., the elderly quiz), interesting for Norwegian elderly people? Can it help players to learn new things?
- Objective 2: Are the questions in the elderly quiz well-formulated? Do the answer alternatives seem appropriate for such a quiz?
- Objective 3: Do the question categories and time periods to choose from for each question category in the elderly quiz seem appropriate?
- Objective 4: Do the participants have any preference between being asked questions about a specific picture and being asked a purely textual question? Do they find one to be more easy than the other?
- Objective 5: How challenging is it for the elderly to answer random questions, from a specific question category, and within a chosen time period (decade to decade), from the elderly quiz? Does the difficulty level for the elderly quiz seem appropriate, or should the questions be easier or harder?

- Objective 6: Would the participants play the elderly quiz again, if it was made available to them? If so, would they prefer to play alone or with other people?

To collect data, the usability sessions were audio recorded. In addition, computer-generated log files were created when participants played the quiz. These automated recording methods ensured that the test moderator could focus on conducting the tests, rather than having to record data manually. If it was necessary to supplement the automatically recorded data, the moderator could take notes. The data collected were performance data from the test of the quiz; and preference data from interviews, thinking aloud technique during the test of the quiz, and (if applicable) answers to unscripted follow-up questions when participants were playing the quiz.

4.4.2 Usability session

Before conducting the tests, some materials were developed that would facilitate the test procedure. These materials were: 2-part session introduction script (see Appendix E), consent form (see Appendix F), pre-test interview guide, and debriefing guide. These materials will be presented in the following text, based on where they appear in the test procedure.

A session began with the test moderator reading the first part of the session introduction script. This part of the script gave the participant(s) some general information about the quiz that would be evaluated, which data would be collected and what would happen to the data after the user study was over. During the session introduction script each participant received a consent form that was orally explained to the participant(s) by the test moderator. This form was adapted from a consent form template by the Norwegian Centre for Research Data (NSD)[63]. Each participant needed to sign his or her form before the procedure could continue.

When the consent form(s) had been filled out the pre-test interview began. The pre-test interview guide listed themes and relevant questions for this interview. The interview was used to establish what previous experiences the participants had with playing quizzes, and if they enjoyed this type of activity. The information from the interview could later shed light on other information gained through the test of the quiz. Additionally, the pre-test interview was used to examine what attitudes the participants had toward the elderly quiz in particular (after a short introduction to it), if they could see the elderly quiz being valuable (for example by being

[63] http://www.nsd.uib.no/personvern/meldeplikt/samtykke.html

interesting and helping them to learn), and what experience they had using computers (the assumption being that little or no experience with computers could possibly be related to scepticism toward a computer-based system). After the pre-test interview, the second part of the session introduction script was read. This part explained how the quiz itself would be tested, the protocol for the rest of the session, and how the test moderator would interact with the quiz for the participant(s). The script encouraged the participants to think aloud during the test of the quiz; and emphasized that it was the quiz that was being tested, not the participants.

Now the test of the quiz could start. The test moderator controlled the interaction with the quiz prototype and did not show the computer screen to the participants at this point. The reason for this was to ensure that the focus of the evaluation would be on the quiz itself, rather than user interface development. The first window opened by the test moderator was the quiz configuration window (see Figure 3.2), where preferred settings could be selected.

As the usability (assessment) test tasks, the participants would play two separate quiz games of the elderly quiz (each containing ten questions). The type of quiz was set according to the amount of participants. If one participant was involved in the usability testing the game type would be "1 player" for both quiz games. If two participants were involved in testing then in one of the games played the game type would be set to "2 player (co-operative)", while in the other game played the game type would be set to "2 player (competition)" (the order of playing co-operative versus competition was decided by the participants). Two different question categories would be chosen by the participants during the test; one in the first game, the other in the second game. One question category should contain picture questions (about actors or musicians) and the other should contain textual questions (about movies or literary works). The participants could decide on the time period they wanted to be asked about (within available decade ranges, as described in section Section 3.3.3) for each of the two question categories, the exception being the musicians question category where this was not possible. The minimum decade range from year to year was 20 years. When two participants were involved in testing they would have to decide on the question categories and time periods together.

After the test moderator had input the configuration options to the quiz prototype, the first quiz game could start. For each question, the test moderator would read the question with its answer alternatives to the participant(s), and input the chosen alternative into the quiz. The participant(s) would only be shown the computer screen if it was necessary for answering a question. This was only necessary when they were asked a picture question. In that case the test moderator

would click on the picture to display a higher-resolution version of it in a maximized window (not containing any other information), before showing the screen. Figure 3.3 shows a screenshot of the quiz GUI for a picture quiz, before the picture is clicked on to display it in a maximized window. After playing two games with 10 questions each, the test of the quiz was finished. One log file for each game documented information from, and the results of, playing the quiz.

After testing the system, post-test debriefing interviews were conducted. The debriefing guide listed themes and relevant questions for the interview. The debriefing interviews inquired about experiences of having played the quiz. This was also an opportunity to ask about particular problems that came up during the session. In addition, the answers to the debriefing questions could be used to know more about the satisfaction with the quiz, and to obtain some suggestions for improvements.

4.5 Data analysis and results

After conducting each usability session, the collected data was collated and examined, and the test moderator wrote a summary which included both preference and performance data. Based on the summaries, supported by information from the test log files, data were analyzed. The method I used was a type of qualitative data analysis, which involves searching for themes (or patterns) in the data. The identified themes can be categorized into non-overlapping categories (Rogers et al., 2011).

For analyzing data, I laid data out in a spreadsheet to easier see patterns between the tests and individual participants. When analyzing the usability sessions, I mainly used a task-oriented approach, as recommended by Rubin and Chisnell (2008). The task-oriented approach involved focusing on tasks that could be perceived as difficult to perform by the users due to the design of the quiz itself; in other words, if the representation of questions and answer alternatives made it difficult to use the quiz effectively.

The results from the evaluation will be presented in the following sub-sections. To provide relevant background information about the participants and their attitudes, results from the pre-test interview will first be presented. This is followed by a summary from the tests of the quiz. Thereafter results will be presented, categorized according to the test objectives.

4.5.1 Participant background information (from pre-test interview)

The pre-test interview showed that all of the participants had experience with quizzes. P1 had only played one time a few years ago, in a pub quiz. P2 had played pub quizzes several times in later years, and had also answered quizzes from newspapers and the radio. P3 and P4 had both played quizzes in social settings, such as at home with family, and at work with colleagues. None of the participants had played quizzes from computers or other electronic mediums before. In the pre-test interview P2 and P4 stated that they liked quizzes. P2 said he liked quizzes because they are fun and help him to remember things. P4 stated that she liked quizzes because it is a social activity. P1 and P3 expressed that they did not like quizzes. P1 stated that this was mainly because she thought it could be difficult to memorize and remember facts, especially as she has gotten older. When asked about something she did not know, she would rather read to find out the correct answer than make a guess. P3 said he prefered reasoning and thinking instead of quizzes.

When asked about what question categories they would prefer to be asked about from a quiz, P1 and P2 stated they would both like to be asked about different categories of "general knowledge". P3 and P4 both expressed they prefer to be asked about geography and literature. P3 also liked to answer questions about history. When asked about categories they did not like to be asked about, the participants had varying preferences. P1 did not like geography and nature. P2 would rather not be asked about literature. P3 did not like religion, politics, and sports. P4 did not like to be asked about celebrities. The participants had varying experience with computers. P2 and P4 said they used computers on a daily basis, P3 said he seldom used computers, while P1 expressed she never used computers. When given a description of the elderly quiz, all participants stated that such a quiz sounded interesting. P1 and P2 also thought it could help them to remember, and P4 said that she thought it had potential for learning.

4.5.2 Tests of the quiz

Table 4.2 below shows a summary from the tests of the quiz. P1 and P2, who did the testing together, played one game co-operatively and the other game competitively. The other two participants tested the quiz alone. As mentioned in Section 4.4.2, the participants chose the question category and time period (within available decade ranges) that was played for each game; the constraint being that one of the games played must be a picture quiz, and the other game must have textual questions.

Test # (participant)	Game type	Quiz category	Decade range	Points (out of 10 possible)
Test 1 (P1,P2)	2 player collaborative	Actors (picture)	1920-1950	7
	2 player competitive	Movie	1940-1980	6 (P1: 3/5, P2: 3/5)
Test 2 (P3)	1 player (both games)	Actors (picture)	1890-1950	5
		Movie	1940-1980	4
Test 3 (P4)	1 player (both games)	Musician (picture)	1890-1950	7
		Literature	1940-1980	3

Table 4.2: Tests of the quiz

Table 4.2 shows that all of the question categories in the quiz were tested by one or more of the participants. For the most part, the participants chose the full decade ranges that were available for the question categories. One exception was P1 and P2, who collectively chose to narrow the range for actors to 1920 to 1950, making an informed decision because "1890 to 1910 was too long ago". Initially, P3 also wanted to chose this range for actors, but after a few questions the quiz had to be restarted due to Internet connectivity issues; he then chose the full available decade range, from 1890 to 1950.

The number of correct answers for a game varied from 3 out of 10, to 7 out of 10. The average amount of correct answers for the six games played was 5 (5.33).

4.5.3 Categories

The following section will list results, categorized by test objectives (see section 4.4.1). The results presented under these categories mainly originated from the tests of the quiz and from post-test debriefings, although data from the pre-test interviews were also used to provide background information and put the results into context.

Objective 1: Is a computer-based quiz that asks questions about people that were well-known and works (such as movies and literature) that were created during their past (i.e., the elderly quiz), interesting for Norwegian elderly people? Can it help players to learn new things?

Based on the quiz being described in the pre-test interview, all participants indicated that such a quiz sounded interesting (as mentioned in section 4.5.1). In the debriefing P1, P2, and P4

expressed that they thought the elderly quiz was interesting. P3 now disagreed, stating that the quiz was boring. In the pre-test interview, only P4 indicated that she thought the quiz had potential for learning. After having played the quiz, none of the participants said that they had learned anything new. P1 and P2 did, however, express that the quiz had helped them to remember. They mentioned that the multiple choice alternatives had been particularly helpful in this regard.

Objective 2: Are the questions in the elderly quiz well-formulated? Do the answer alternatives seem appropriate for such a quiz?

During the debriefing, the participants generally indicated that the questions in the quiz were well-formulated. P1 thought it was an alright way to formulate the questions. P2 could not really think of any other ways the questions could be formulated for such a quiz. P4 thought the questions were understandable and made sense. However, P3 expressed that some of the movie questions were too long. This was confirmed by a log file from his test, which showed that one question listed 10 actors starring, while another listed 11 actors starring (see Section 3.3.4 for an example movie question). Out of the 10 movie questions asked to the participant, the median number of actors listed were 2, so the two long questions were outliers in this regard.

Several participants mentioned that when testing the quiz, they could often identify some alternatives that were obviously not the answer, which could then be eliminated. This was not considered as negative. Sometimes participants would also look for special patterns for the answers, for instance P1 and P2 said that "Leif Juster only makes comedies" (therefore they would look for a movie title that seemed like it could be a comedy), and P4 stated that "Jon Bing always uses strange short titles for his books". When the participants did not know the answer, the most common strategy for answering was to eliminate answers that did not seem correct, and then make guesses on the remaining. P3 brought up that some alternatives (distracters) re-appeared for several questions in the same game, something he considered as negative. During the debriefing, he also commented that for one question the alternative Pinchcliffe Grand Prix had appeared, instead of the Norwegian title "Flåklypa Grand Prix".

Objective 3: Do the question categories and time periods to choose from for each question category in the elderly quiz seem appropriate?

The question categories available for questions seemed to be appropriate with regards to the question categories that the participants liked to be asked about (for instance P1 and P2 liked to be asked about general knowledge, P3 and P4 both liked to be asked about literature, and P3 also liked to be asked about history). Several participants said they would try with other categories if possible.

Regarding time periods, P1 and P2, who narrowed the time period for actors (birth year) from 1920 to 1950, said this was an appropriate time period for them. They also indicated that being asked about movies released from 1940 to 1980 was appropriate for them, because they knew these movies better than newer movies. P3 thought that the questions about older actors and movies made the quiz more difficult. He suggested that the time periods could be extended to the year 2000. P4 also expressed that it would be easier to answer questions about literature from a newer time period. She was given a chance to play a game with time period from 1960 to 1980 instead of 1940 to 1980, and then got a substantially better score (8 out of 10 correct answers, versus 3 out of 10). However, some questions reappeared between the games, thus the much better score could be due to learning the answers in the first of the literature games.

Objective 4: Do the participants have any preference between being asked questions about a specific picture and being asked a purely textual question? Do they find one to be more easy than the other?

Regarding questions with pictures versus purely textual questions, I could not really find any preference. P1 and P2 commented that both picture questions and textual questions have their purpose, and that they complement each other in a quiz. With regard to which is more easy, it seems to depend on the information given in the question string, and the quality and context for the picture taken (when asking a picture question). As the tests indicated, which type of question is perceived as the most difficult will also vary among participants. For example P3 thought that there were more clues in the actor questions than the movie questions, making them easier to answer; while P1 and P2 had the opposite opinion.

Objective 5: How challenging is it for the elderly to answer random questions, from a specific question category, and within a chosen time period (decade to decade), from the elderly quiz? Does the difficulty level for the elderly quiz seem appropriate, or should the questions be easier or harder?

The results from the tests of the quiz showed that the amount of correct answers for a game varied according to the participant(s), game type, question categories, and time periods chosen. The highest amount of correct answers (7 out of 10) was obtained by P1 and P2, when they collaboratively answered questions about actors born between 1920 and 1950; and by P4 when she answered questions about musicians born between 1890 and 1950 (7 out of 10 correct). The lowest amount of correct answers (3 out of 10) was obtained by P4 when she answered questions about literature published from 1940 to 1980. As mentioned in Section 4.5.2, the average amount of correct answers for all of the played games was 5.

The participants generally indicated that the quiz was a bit difficult. P1 and P2 said that some of the actor questions were difficult. Sometimes they did not know the picture and alternatives. At other times the picture was taken from afar, in an unfamiliar context, or the actor was very young at the time; this made it difficult to recognize some actors even though P1 and P2 stated that they knew what the actors looked like. P1 and P2 mostly thought that the movie questions (released from 1940 to 1980) had an appropriate difficulty level, although there was one question both considered too easy. P1, who during the pre-test interview expressed that she did not like quizzes because it is difficult to remember, stated that the quiz was easier than she had imagined before playing.

P3, who chose questions about movies from the same period as P1 and P2, and about actors born from 1890 to 1950, expressed that the quiz was difficult. He expressed that the questions about the older movies, and about actors that were born a long time ago, were particularly difficult. Sometimes he did not know any of the alternatives. P3 thought that the actor questions were easier than the movie questions, because they gave him more clues to the correct answer (as mentioned in the results related to objective 4). During the debriefing P3 expressed disappointment with his amount of correct answers. P4 thought that musician questions were difficult due to her visual impairment; she had troubles seeing the pictures. She suggested that the quiz could have zoom functionality for images to alleviate this problem. The literature questions (from 1940 to 1980) were difficult for her because of the lack of hints (she would have liked to see more information about the literature in the questions), and sometimes all alternatives were unknown to her. P4 expressed that it would probably be easier for her to answer literature questions from a more recent time period. She was given a chance to test this assumption (as was explained in the results related to objective 3).

The tests showed that one measure that made the quiz easier was the use of multiple choice alternatives. This made it possible to eliminate wrong alternatives, and (as stated by P1 and P2) it made it easier to remember the correct alternative. P1 and P2 agreed that in several instances they would probably not have been able to answer correctly, if they did not have the multiple choice alternatives.

Objective 6: Would the participants play the elderly quiz again, if it was made available to them? If so, would they prefer to play alone or with other people?

P1, P2 and P4 indicated that they would play the elderly quiz again. P3 expressed that he might play it again. All participants stated that they prefered to play with other people. Only P2 said explicitly that he could play the quiz alone. P1 and P2, who played both co-operatively and competitively said that they preferred to play competitively, because it is more exciting. They seemed to be enjoying themselves while playing, something that was confirmed during the debriefing. P3 and P4, who played the quiz alone during the test, expressed a preference for playing such a quiz co-operatively. P4 said that if she was playing co-operatively with others, it would be possible to help each other and discuss the questions and answer alternatives.

4.6 Discussion and suggested improvements

Three out of four participants mainly expressed positive attitudes toward the quiz. The results showed that the quiz was interesting for most participants, that it could help them to remember, and that most participants would play the quiz again. The participant preference is to play such a quiz with others. The questions were mostly found to be well-formulated, the answer alternatives were for the most parts found to be appropriate, and having multiple choice alternatives were suggested as a positive feature. There is some room for improvement for questions and answers, which will be discussed later in this section.

The time periods available for the question categories were generally found to be appropriate, although two participants, P3 and P4, expressed that it would probably be easier for them if they could also choose newer time periods. It should be noted that P3 and P4, who were positive towards adding newer time periods to the quiz, are approximately 10 years younger than P1 and P2. Overall, the participants expressed that the quiz was a bit difficult. P3 was disappointed that he could not answer more questions correctly. The question categories in the quiz were found to

be suitable for the participants, although more could be added (based on the question categories the participants stated that they like to be asked about).

If we return to the goals of the usability testing (see Section 4.2), we can now attempt to get an indication of the usability of the quiz according to the following usability criteria: usefulness, effectiveness, and satisfaction. Regarding usefulness, we see that three out of four participants would play the quiz again, and one participant might play it again. This is an indication that the elderly quiz lets participants achieve their goal (of playing quizzes). Regarding effectiveness, the overall results indicate that the quiz behaves like the participants expect, and that they can achieve their goals (of playing quizzes) without much difficulty. Some issues were, however, noted by the participants during the tests; and some suggestions of improvements were given (as will be discussed later in this section). By correcting identified issues and implementing suggestions, the effectiveness of the quiz could be improved further. Regarding satisfaction, three out of the four participants expressed positive attitudes toward the elderly quiz. They overall expressed satisfaction with the product. The last participant did not seem satisfied with the quiz, he said the quiz was boring, and expressed disappointment that he could not answer more questions correctly during the two games. If identified issues were corrected and suggestions were implemented, the satisfaction with the quiz could increase further.

As mentioned above, the participants suggested some improvements for the quiz during the usability testing:

- Expand the available time periods to choose from for the question categories, enabling the choice of newer time periods (P3 suggested to expand all time periods to year 2000).
- Add zoom functionality for pictures.
- Add larger font size (suggestion made by P4; she was shown the quiz user interface when playing the second game of literature questions, to compare playing the quiz with two different time periods).
- Questions should not re-appear between games (suggestion made by P4; after she played the second game of literature).
- Add extra hints to questions, for instance a picture question about a musician could also contain information about a song that the musician had sung.

Some issues also appeared during the tests:

- Two movie questions listed 10 or 11 actors starring, making them unnecessarily long.
- The quiz had to be restarted one time during a test because of Internet connectivity issues.
- In some picture questions it was difficult to recognize the person, even though participants stated that they knew what the person looked like.

Based on the suggestions and issues listed above, I would finally like to suggest some functionality that could be added to further improve the quiz and its usability:

Configurable font size

Although the graphical user interface was generally not the focus of the evaluation, a non-functional requirement was to use a font size larger than average, to make text more readable for people with reduced vision. The two font sizes used were 18 point (used for most text), and 30 point (used for some headings); as stated in Section 3.3.3. P4, who was the only participant that was shown the font size, suggested that the font size should be increased. As noted earlier, she is visually impaired. Regarding which font sizes to use when displaying body text (standard text, not including headings) to elderly people, "numerous studies show that font size should be 12-point or larger to accommodate aging vision" (Becker, 2004, p. 394). Hodes and Lindberg (2002), at the United States' National Institute on Aging and the National Library of Medicine, has recommended using either 12 point or 14 point font size for body text. These recommendations were made specifically for web pages, but I assume they are also relevant for a computer program (such as the elderly quiz).

Regarding using 12 or 14 point font size, a study by Bernard, Liao and Mills (2001) showed that elderly people overall found 14 point font size to be more legible than 12 point font size. The study also found that the elderly read faster with, and generally had a preference for 14 point font size. The 18 point font size used for most of the elderly quiz is already four points larger than the recommendation of 14 point. Instead of further increasing the font size, I would rather suggest that the elderly quiz could have a font size that is configurable by its users. One solution could be that when the quiz starts, the participants are shown a configuration window with text in some different font sizes, and that they are asked to click on the one that they prefer. The font sizes (that would be exemplified with the text) to choose from could include 14 point, 18 point, and a few larger font sizes. Which larger font sizes to include would need further examination.

The information in the quiz would need to be organized differently according to the different font sizes.

Expand available time periods for questions

Overall, it seems advantageous to let users choose from newer time periods than those that are currently available in the quiz. However, if the time periods are too new, it would not be a quiz asking users about their past anymore. Expanding the available time period for actor and musician birth years to 1970, and movie and literature release or publication years to 2000, could be a way to make the quiz easier and more appropriate for the elderly that prefer to be asked about newer time periods than those that are already implemented in the quiz.

Other functionality that could be added

- Allow the users to zoom while seeing pictures in picture quizzes.
- Create a heuristic to filter out pictures that are taken far from the person in question. It would be interesting to research the use of Computer Vision algorithms to achieve this.
- Set a limit for the maximum amount of actors that can be listed as starring in a movie question (currently all relevant actors found in DBpedia are added to the question string). I would suggest setting the maximum amount to four; as this gives the users some important hints, while avoiding the questions being very long, difficult to read and hard to remember.
- Make the quiz more tolerable to Internet connectivity issues. If the computer that has the quiz installed loses its Internet connection while the quiz is running, the quiz should always be able to resume execution when the Internet connection is re-established.
- The quiz could have the possibility of registering and accessing the quiz with user profiles. When a participant accesses the quiz with their user profile, the quiz could keep track of the questions they had been asked, to avoid repeating questions within short time intervals between games. With user profiles, one could also keep track of individual users' previous amount of correct answers for the question categories.
- Avoid asking about English titles for Norwegian movies instead of Norwegian titles. The problem is that a few of the Norwegian movies obtained from DBpedia do not contain a Norwegian title at all, as described in Section 3.3.7. Re-examining the problem at a later time has indicated that the issue would be fixed if updating to the DBpedia 2015-10 dataset[64]. This would naturally need to be confirmed after updating.

[64] http://wiki.dbpedia.org/Downloads2015-10

Chapter 5 - Conclusion and Future Work

Studies have indicated that games can have benefits for the elderly, in areas such as cognitive functioning and well-being. Participating in social activities, such as playing a quiz with others, could also have health benefits for some. During this master project, I have developed a quiz application (high-fidelity prototype) for Norwegian elderly people, where the questions and answers are automatically generated using information from Wikipedia (also referred to as the elderly quiz). The quiz can ask factual questions about people (actors and musicians) that were well-known, and works (movies and literature) that were created, during the past of its target users (elderly people above the age of 60). For each question, the quiz has four multiple choice answer alternatives to choose from, where one is the correct answer. The quiz can be played alone or with another player (competitively or co-operatively).

The elderly quiz has been developed using the incremental model, combined with iterative development. Eight development iterations were conducted, each aiming to deliver a new increment of functionality. The first requirements for the quiz were decided on in the beginning of the project, but more requirements were added and existing requirements were refined as the quiz prototype evolved. The emphasis for this project was placed on implementing functional requirements. Through the incremental and iterative development of the quiz, I have been able to answer the research question about how quizzes for Norwegian elderly can be automatically generated using online resources. Different tasks, techniques and technologies have been researched and applied to answer the research question; largely within the field of NLP (IE) and the Semantic Web. For representation and storage of quiz data, Semantic Web technologies, such as RDF, OWL (ontology made by DBpedia), and a RDF database have been employed. To generate questions and answers, RDF APIs are used to retrieve data from the quiz database.

To obtain data for generating questions and answers, datasets from the DBpedia project were first used. In this case, the DBpedia project had already extracted (mostly) structured information from Wikipedia, for example from infoboxes, which was represented in RDF. Two of the DBpedia datasets that were used combined data extracted from DBpedia with data that had been extracted by the YAGO project. The YAGO project also extracts structured data from Wikipedia, but they have a more intricate system for inferring the type of a resource. I made the DBpedia data obtainer, which could obtain smaller databases for specific entity types and time periods, from a local DBpedia database where DBpedia datasets had been loaded. These smaller

databases could then be combined into a quiz database. Some heuristics were also implemented, that would allow more resources to be selected from the local DBpedia database for storage in the smaller databases. Currently, the DBpedia data obtainer can obtain databases for Norwegian actors, musicians, and movies, but it could easily be extended to obtain databases for other entity types.

After finishing the DBpedia data obtainer, NLP supported automatic question generation was researched. During this process I searched for and applied NLP (IE) tasks and techniques that could be used to extract quiz data. The plan was initially to extract quiz relevant information from unstructured text in Norwegian documents, with the help of NER and POS tagging (as inspired by Kim and Kim, 2008). Unfortunately, this plan had to be rejected, as the tool I planned to use (Oslo-Bergen tagger) did not support NER at the time. From there on I focused on extraction of quiz data from English text (about Norwegian topics). Different tools and libraries were tested, related to tasks such as Web IE, NER, and relation detection and classification. Some of the tools and libraries proved valuable in the search for an appropriate solution for generating questions, while others did not seem to fit the use-case of generating questions for Norwegian elderly people.

Throughout the process of researching NLP supported automatic question generation, two central quiz components were developed. The first is the Scrape utility, which is a Web IE extractor that applies different DOM extraction rules to DOM parse trees that are constructed from semi-structured HTML documents from Wikipedia. The Wikipedia HTML documents are all related to literature, and the extraction happens from semi-structured elements (mainly lists and tables). After applying the extraction rules, the extracted data usually needs to be post-processed. After post-processing, the data are represented in RDF and stored in a literature database that is combined with the quiz database, meaning the data can be used for question generation. The data are extracted from HTML elements that are not covered by DBpedia extractors (as the DBpedia project has it focus on extraction of more structured data). Literature data was chosen as there was generally not a lot of information about this available in the DBpedia datasets. The Scrape utility also creates gazetteers for rule-based NER for Norwegian authors, literary works and last names.

The second central quiz component that was developed when researching NLP supported question generation was the RDC module. This module works with unstructured text and can detect and classify entities and their relationships from sentences (that is, binary relationships

between authors and their literary works). The current document collection is author HTML documents from Wikipedia, which are also used by the Scrape utility. The RDC module applies the task of relation detection and classification; but before this task can be achieved, unstructured text must be run through a NLP pipeline, to process the text before it is ready for relation detection and classification. The processing in the pipeline includes applying rule-based NER, with the help of the gazetteers made by the Scrape utility, to classify the authors and literary works in sentences.

The RDC module contains five patterns to identify relationships between authors and their (award-winning) literary works, thus it uses a rule-based approach for detection and classification. The detected and classified binary relationships between entities could have been represented as RDF triples, and used for question generation. However, when the RDC module was functional, the literature dataset created by the Scrape utility had already stored this type of information (the initial plan was to use the Scrape utility only for gazetteer extraction, but it evolved into a dataset extractor as well). Based on an observation that the literary works that are mentioned in unstructured text in author articles on Wikipedia are often the more well-known works of that author, it was decided that the RDC module could rather be used to distinguish literary works in the literature database. A RDF triple is added to the literature database for each literary work that is found to be mentioned together with its author in unstructured text. Even though Wikipedia documents is used as the document collection for the RDC module, the patterns are generic, meaning they could also be expected to work with documents from other data sources (this would need to be tested more thoroughly).

Regarding using Wikipedia HTML documents as a document collection; the RDC module works with unstructured text, the Scrape utility extracts information from more semi-structured elements (lists and tables), while DBpedia extracts data from more structured elements (such as infoboxes). In this way, the data that are identified or extracted by the RDC module, the DBpedia project, and the Scrape utility, all originate from different parts of a Wikipedia document.

Several projects have focused on automatic question generation (Zeng et al., 2013; Xu, Goldie & Seneff, 2009; Kim & Kim, 2008; Fjellanger & Armstrong, 2013). The elderly quiz is a multiple choice quiz. This is something that it has in common with several of the projects (Zeng et al., 2013; Fjellanger & Armstrong, 2013). We wanted to make a quiz for elderly people which could help them to remember, by answering questions about well-known people and works from their past. Therefore, the questions were not meant to be very difficult. The intended difficulty level

has commonality with the automatic multiple choice quiz generation system using tourism blogs (Zeng et al., 2013), where the aim was to make the users remember information more easily. The questions in the elderly quiz were generated using extracted data that was stored in a RDF database. Fjellanger and Armstrong (2013) also used extracted data stored in a RDF database. During the master project all data used for question generation in the elderly quiz came from Wikipedia articles. Kim and Kim (2008) also used Wikipedia as a data source. What differentiates the elderly quiz from the other works is that it is a Norwegian language quiz (however, methods used are generic, meaning it would be easy to generate questions in other languages as well). The other quiz generation projects use either English, Japanese or Chinese as their language. Another difference is that this quiz has elderly people as its target audience, which is not the case for any of the other mentioned works. The elderly quiz thus has commonality with the tangible tabletop quizzes that Chen (2013) designed specifically for elderly people. However, the questions in the tabletop quizzes were created manually.

The elderly quiz was evaluated by means of usability testing. Four people above the age of 60 participated in the usability testing, which involved observing participants while they were playing the quiz (the quiz was operated by the test moderator), and interviews to complement the data gathered during the observation. The results indicated that the quiz could help the participants to remember. The participants all preferred to play such a quiz as a social activity, rather than alone. Since the aim was to make a quiz that could help participants to remember, we did not want to make a quiz where the questions were very difficult to answer. In this regard, the tests indicated that the questions were somewhat more difficult than we had aimed for. Three usability measures were assessed during the evaluation; usefulness, effectiveness, and satisfaction. The majority of the participants thought the quiz was useful, effective to use (with the constraint that the test moderator operated the quiz for the participants), and expressed satisfaction with the quiz.

5.1 Reflection

Developing the elderly quiz has been an interesting learning experience. The incremental model, combined with iterative development, was very suitable for a project where the requirements evolved over time. As the requirements evolved, I could research and apply different tasks, techniques and technologies to find suitable ways to implement the requirements in the quiz prototype. The incremental model, combined with iterative development, was also very fitting

since I did not know in advance exactly how much functionality I would be able to implement in the elderly quiz.

Semantic Web technologies were found to be very suitable for storage and representation of data used for automatic question generation. These technologies turned out to be reliable, and through the Jena framework they were easy to work with. In general, Wikipedia turned out to be a good source for quiz data; the data used for question generation was accurate, and it was a plus that Wikipedia is cross-domain, meaning it was possible to use Wikipedia for generating different types of questions. Ideally, other online sources would also have been used; however, this was not possible during the master project, as explained in Section 3.3.1. Using datasets from the DBpedia project to obtain data from Wikipedia proved to be a good solution, at least after I found the tool tdbloader2 which could efficiently populate a local TDB database with specific DBpedia datasets. Extracting quiz data from semi-structured elements (lists and tables) in Wikipedia HTML documents was an alright way to obtain quiz data. For Norwegian literature, it turned out that more data could be obtained with this approach, than was available in DBpedia. However, it was time-consuming to extract and post-process the data, and this type of solution would probably have to be altered quite frequently to accommodate changes in HTML pages.

Identification (or extraction) of data from unstructured text, by means of relation detection and classification, seemed like a good idea. When developing the RDC module, the thought was that distinguishing literary works that appeared in a binary relationship with their author in unstructured Wikipedia text, could for instance be used to make the quiz easier and more relevant for the users. As mentioned, five patterns were created for the RDC module. More literary works would need to be distinguished through use of patterns before the RDC module could be used to make the quiz easier or more relevant for the users. Distinguishing more works could be done through adding of more patterns, or adding more documents to the document collection. The RDC module uses a (hand-written) rule-based technique, with patterns that should be matched against (patterns are written in the TokensRegex syntax, which extends ordinary regular expression syntax). When such rules are written to have a higher precision, they will match less text. In hindsight, it would probably be better to attempt to use a semi-structured machine learning approach (see Section 2.2.2.3, paragraph: Semi-supervised learning) for relation detection and classification. In this way more patterns could be added automatically, and more text could be matched. It would be interesting to try out such an approach with TokensRegex patterns.

The evaluation had some limitations, due to time and resource constraints. Four participants evaluated the elderly quiz. While the number of participants would indicate that a substantial number of usability deficiencies could be detected, having more participants could have exposed further issues. Usability testing was conducted at one specific stage of development, after all of the development iterations had finished. Ideally, it would have been preferable to test the quiz several times and at different stages of development. If this type of iterative testing had been employed, the requirements (or functionality) of the quiz could have been changed after each type of test had been conducted, based on the results from the previous test.

5.2 Future Work

The evaluation revealed some issues and possible improvements for the quiz prototype. Based on this, some suggestions have been made for functionality that could be added to further improve the quiz and its usability, as stated in Section 4.6. Adding of the suggested functionality could be done as future work on the quiz prototype.

Another change that could be made concerns the quiz difficulty level. In general, it seems that it could be positive to use some measures to make the quiz easier (in addition to expanding available time periods, as suggested in Section 4.6, paragraph: Expand available time periods for questions). This is because during the evaluation, the participants overall expressed that the quiz was a bit difficult. To make sure that the questions are suitable for participants with various amounts of pre-existing knowledge about a question category, it could be differentiated between difficulty levels for each question category. The difficulty levels could for instance inform the type and amount of hints given in individual questions, and the similarity between the correct answer and distracters (where more similarity leads to questions that are more difficult to answer). For the literature category, distinguishing more literary works with the RDC module could also be used to differentiate between the difficulty of questions (as mentioned in Section 5.1, it could be interesting to try to use semi-structured machine learning to add more patterns and distinguish more literary works). It would be possible to let users decide on the difficulty level for individual question categories by themselves; but if user profiles were implemented (see suggestion in Section 4.6, paragraph: Other functionality that could be added), the quiz could dynamically decide on appropriate difficulty levels, based on individual users' previous amount of correct answers for the question categories.

Although it worked well to use DBpedia with English datasets for automatic question generation, it could also be interesting to add Norwegian datasets to the local DBpedia database in the future. The Norwegian datasets may not have the same level of data quality (as mentioned in Section 2.3.3), but they could still be good enough for question generation. Information about a resource from English DBpedia could be related to the same resource in Norwegian DBpedia. In this way, more data would be made available to the quiz, and Norwegian culture could have better coverage.

As mentioned in Section 2.3, most of the text and images in Wikipedia are licensed under a Creative Commons license (CC-BY-SA 3.0), which requires attribution of the author (if one exists) or of the licensor (Wikipedia). The quiz prototype does not include attribution. The final product of the elderly quiz would therefore need to include the appropriate copyright information, both for textual information and pictures.

References

Allemang, D. & Hendler, J. (2011). *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL* (2nd ed.). Morgan Kaufmann.

Apache. (2015). Apache Jena. Retrieved from http://jena.apache.org/

Bach, N. & Badaskar, S. (2007). A review of relation extraction. *Literature review for Language and Statistics II.*

Baptist, L. & Seneff, S. (2000). GENESIS-II: a versatile system for language generation in conversational system applications. In *INTERSPEECH* (pp. 271-274).

Basak, C., Boot, W. R., Voss, M. W. & Kramer, A. F. (2008). Can training in a real-time strategy video game attenuate cognitive decline in older adults?. *Psychology and aging*, *23*(4), 765.

Baeza-Yates, R. & Rebeiro-Neto, B. (2011). *Modern Information Retrieval. The concepts and technologies behind the search* (2nd ed.). Addison-Wesley.

Becker, S. A. (2004). A study of web usability for older adults seeking online health resources. *ACM Transactions on Computer-Human Interaction (TOCHI)*, *11*(4), 387-406.

Bernard, M., Liao, C. H. & Mills, M. (2001). The effects of font type and size on the legibility and reading time of online text by older adults. In *CHI'01 extended abstracts on Human factors in computing systems* (pp. 175-176). ACM.

Bizer, C., Heath, T. & Berners-Lee, T. (2009). Linked data - the story so far. *International Journal on Semantic Web and Information Systems*, *5*(3), 1-22.

Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R. & Hellmann, S. (2009). DBpedia-A crystallization point for the Web of Data. *Web Semantics: science, services and agents on the world wide web*, *7*(3), 154-165.

Chandrasekaran, B., Josephson, J.R. & Benjamins, V.R. (1999). What are ontologies, and why do we need them? *IEEE Intelligent Systems, 14*(1), 20-26.

Chang, A. & Manning, C. (2014). TOKENS REGEX: Defining cascaded regular expressions over tokens.

Chang, C. H., Kayed, M., Girgis, M. R. & Shaalan, K. F. (2006). A Survey of Web Information Extraction Systems. *IEEE Transactions on Knowledge and Data Engineering, 18*(10), 1411-1428.

Chen, W. (2013) Remembering the Old Days - Designing Tangible Tabletop Games for the Elderly. In *Assistive Technology: From Research to Practice AAATE 2013* (pp. 1126-1132). IOS Press.

Cockburn, A. (2008). Using both incremental and iterative development. *STSC CrossTalk, 21*(5), 27-30.

Cognitivity [Def. 2]. (2016). *Dictionary.com Unabridged*. Retrieved May 14, 2016 from http://www.dictionary.com/browse/cognitivity

Collobert, R. & Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning* (pp. 160-167). ACM.

Elmasri, R. & Navathe, S. (2011). *Database systems: Models, languages, design, and application programming* (6th ed.). Pearson.

Farmakiotou, D., Karkaletsis, V., Koutsias, J., Sigletos, G., Spyropoulos, C.D. & Stamatopoulos, P. (2000). Rule-based named entity recognition for Greek financial texts. In *Proceedings of the Workshop on Computational lexicography and Multimedia Dictionaries* (pp. 75-78).

Fjellanger, M. & Armstrong, J. (2013). Semantic quiz [Software]. Retrieved from https://github.com/MaikenBeate/-semanticquiz

Giles, J. (2005). Internet encyclopaedias go head to head. *Nature, 438*(7070), 900-901.

Goldstein, J., Cajko, L., Oosterbroek, M., Michielsen, M., Van Houten, O. & Salverda, F. (1997). Video games and the elderly. *Social Behavior and Personality: an international journal*, *25*(4), 345-352.

Goyvaerts, J. & Levithan, S. (2012). *Regular Expressions Cookbook* (2nd ed.). O'Reilly.

Grav, S., Hellzèn, O., Romild, U., & Stordal, E. (2012). Association between social support and depression in the general population: the HUNT study, a cross-sectional survey. *Journal of clinical nursing*, *21*(1-2), 111-120.

Guarino, N., Oberle, D. & Staab, S. (2009). What is an ontology? In *Handbook on ontologies* (pp. 1-17). Springer. Retrieved from http://link.springer.com/10.1007/978-3-540-92673-3

Hansen, T. & Slagsvold, B. (2015). Late-life loneliness in 11 european countries: results from the generations and gender survey. *Social Indicators Research*, 1-20.

Heath, T. & Bizer, C. (2011). Linked data: Evolving the web into a global data space. *Synthesis lectures on the semantic web: theory and technology*, *1*(1), 1-136.

Hedden, T. & Gabrieli, J. D. (2004). Insights into the ageing mind: a view from cognitive neuroscience. *Nature reviews neuroscience*, *5*(2), 87-96.

Hebeler, J., Fisher, M., Blace, R. & Perez-Lopez, A. (2009). *Semantic web programming*. Wiley.

Hodes, R. J. & Lindberg, D. A. (2002). Making your website senior friendly. *National Institute on Aging and the National Library of Medicine*.

Hoffart, J., Suchanek, F. M., Berberich, K. & Weikum, G. (2013). YAGO2: A Spatially and Temporally Enhanced Knowledge Base from Wikipedia. *Artificial Intelligence, 194*, 28-61.

Johannessen, J. B., Hagen, K., Lynum, A. & Nøklestad, A. (2012). OBT+stat: A combined rule-based and statistical tagger. *Exploring Newspaper Language. Using the web to create and investigate a large corpus of Modern Norwegian,* 51-65. John Benjamins Publishing Company, Amsterdam/Philadelphia.

Jurafsky, D. & Martin, J.H. (2014). *Speech and Language Processing* (2nd ed.). Pearson.

Kesteren, A., Gregor, A., Ms2ger, Russell, A. & Berjon, R. (2015). W3C DOM4. Retrieved from http://www.w3.org/TR/dom/

Kim, M. K. & Kim, H. J. (2008). Design of Question Answering System with Automated Question Generation. In *Fourth International Conference on Networked Computing and Advanced Information Management* (pp.365-368). IEEE.

Kolomiyets, O. & Moens, M. F. (2011). A survey on question answering technology from an information retrieval perspective. *Information Sciences, 181*(24), 5412-5434.

Laender, A., Ribeiro-Neto, B., da Silva, A. & Teixeira, J. (2002). A Brief Survey of Web Data Extraction Tools. *ACM SIGMOD Record, 31*(2), 34-93.

Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., ... & Bizer, C. (2015). DBpedia-a large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web*, 6(2), 167-195.

Li, X., & Zhong, X. J. (2010). The source code plagiarism detection using AST. In *Intelligence Information Processing and Trusted Computing (IPTC), 2010 International Symposium on* (pp. 406-408). IEEE.

Ma, L., Mei, J., Pan, Y., Kulkarni, K., Fokoue, A. & Ranganathan, A. (2007). Semantic web technologies and data management. In *Proceedings of W3C Workshop on RDF Access to Relational Databases*.

Manning, C. D., Raghavan, P. & Schütze, H. (2008). *Introduction to Information Retrieval.* Cambridge University Press.

Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. & McClosky, D. (2014). The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations* (pp. 55-60).

Munoz, A. (2014). Machine Learning and Optimization.

Nadeau, D. & Sekine, S. (2007). A survey of named entity recognition and classification. *Linguisticae Investigationes 30*(1), 3-26.

Nadkarni, P. M., Ohno-Machado, L. & Chapman, W. W. (2011). Natural language processing: an introduction. *Journal of the American Medical Informatics Association*, *18*(5), 544-551.

Norwegian Institute of Public Health. (2015). Dementia in Norway - Public Health Report 2014. Retrieved from http://www.fhi.no/artikler/?id=114794

Oates, B. J. (2006). Researching Information Systems and Computing. SAGE publications.

Passer, M. W. & Smith, R. E. (2008). *Psychology: The science of mind and behavior* (4th ed.). McGraw-Hill.

Rogers, Y., Sharp, H. & Preece, J. (2011). *Interaction Design: beyond human-computer interaction* (3rd ed.). Wiley.

Reavley, N. J., Mackinnon, A. J., Morgan, A. J., Alvarez-Jimenez, M., Hetrick, S. E., Killackey, E., ... & Jorm, A. F. (2012). Quality of information sources about mental disorders: a comparison of Wikipedia with centrally controlled web and printed sources. *Psychological medicine*, *42*(08), 1753-1762.

Rector, L. H. (2008). Comparison of Wikipedia and other encyclopedias for accuracy, breadth, and depth in historical articles. *Reference services review*, *36*(1), 7-22.

Rosenzweig, R. (2006). Can history be open source? Wikipedia and the future of the past. *Journal of American History, 93*(1),117-146.

Rubin, J. & Chisnell, D. (2008). *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests* (2nd ed). Wiley Publishing, Inc.

Stanford Named Entity Recognizer (Version 3.5.0) [Software]. (2014). Retrieved October 26, 2014 from http://nlp.stanford.edu/software/CRF-NER.shtml

Statistics Norway. (2016). Key figures for the population. Retrieved May 19, from https://www.ssb.no/en/befolkning/nokkeltall

Steptoe, A., Shankar, A., Demakakos, P., & Wardle, J. (2013). Social isolation, loneliness, and all-cause mortality in older men and women. *Proceedings of the National Academy of Sciences*, *110*(15), 5797-5801.

W3C. (2011). Semantic Web Terminology. Retrieved from https://www.w3.org/2001/sw/wiki/Semantic_Web_terminology

W3C. (2015a). Semantic Web. Retrieved from https://www.w3.org/standards/semanticweb/

W3C. (2015b). Query - W3C. Retrieved from http://www.w3.org/standards/semanticweb/query

Weaver, P. (2004). *Success in your project - A guide to student system development projects.* Pearson Education Limited, England.

World Health Organization. (2016). Definition of an older or elderly person. Retrieved from http://www.who.int/healthinfo/survey/ageingdefnolder/en/

Wikipedia. (2016). Wikipedia: About. Retrieved from https://en.wikipedia.org/wiki/Wikipedia:About

Wilson, R. S., Krueger, K. R., Arnold, S. E., Schneider, J. A., Kelly, J. F., Barnes, L. L., ... & Bennett, D. A. (2007). Loneliness and risk of Alzheimer disease. *Archives of general psychiatry*, *64*(2), 234-240.

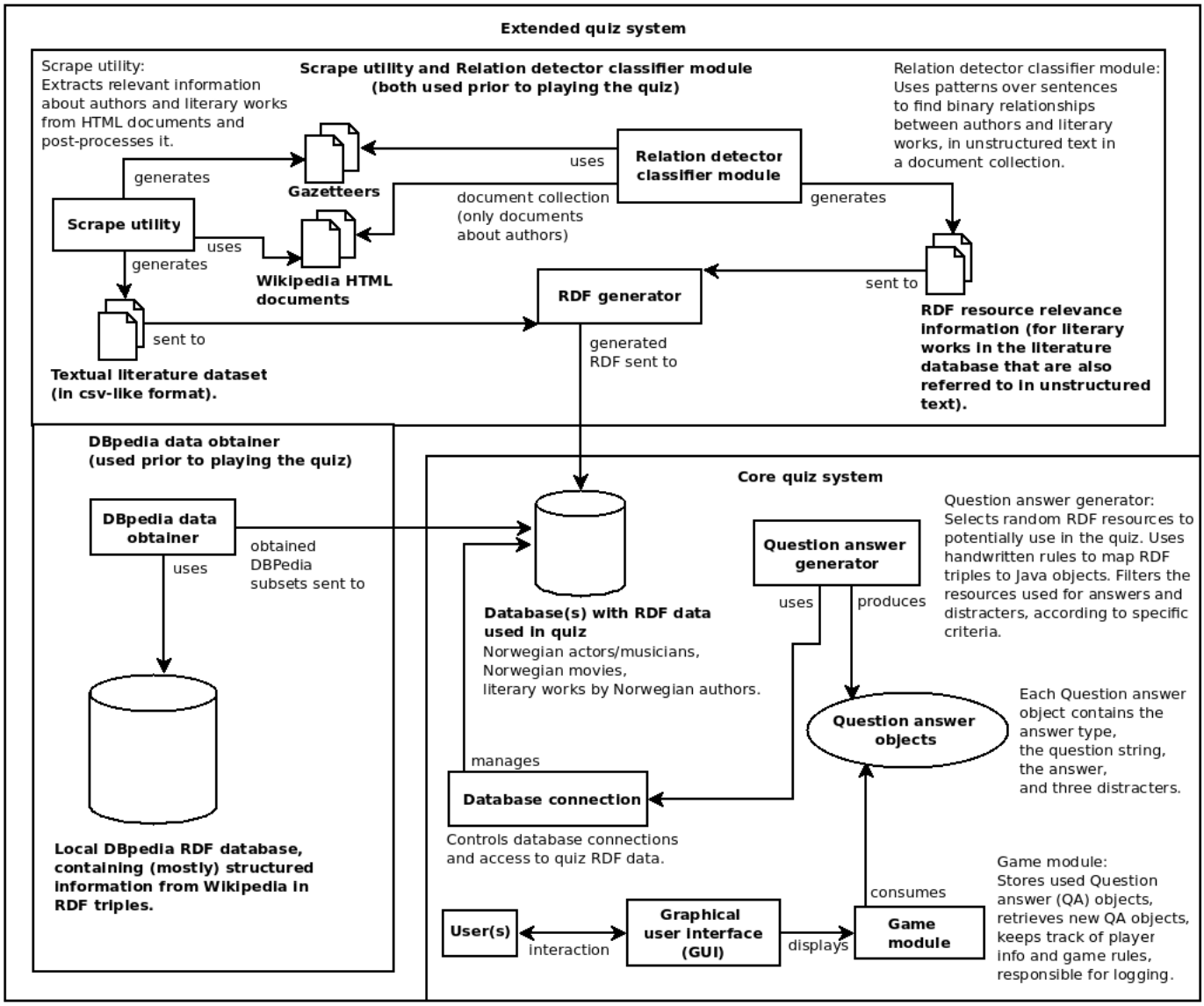
Wilson, R. S., Scherr, P. A., Schneider, J. A., Tang, Y., & Bennett, D. A. (2007). Relation of cognitive activity to risk of developing Alzheimer disease. *Neurology*, *69*(20), 1911-1920.

Wimalasuriya, D.C. & Dou, D. (2010). Ontology-based information extraction: An introduction and a survey of current approaches. *Journal of Information Science*.

Xu, Y., Goldie, A. & Seneff, S. (2009). Automatic Question Generation and Answer Judging: A Q&A Game for Language Learning.

Zeng, J., Sakai, T., Yin, C., Suzuki, T. & Hirokawa, S. (2013). Automatic generation of tourism quiz using blogs. *Artificial Life and Robotics, 17*(3-4), 412-416.

Appendix A - Quiz system architecture with explanations



Appendix B - Local DBpedia database overview

Local DBpedia database overview			
Database	**Description**	**Remarks**	
Local DBpedia database (20.2 GB)	Local database, loaded with datasets which are, or were assumed to be, relevant for the elderly quiz.	DBpedia version:	3.9 (extracted from Wikipedia dumps on 2013–04–03; source: http://wiki.dbpedia.org/services-resources/datasets/data-set-39/downloads-39)
		DBpedia language:	English
		Download links:	http://downloads.dbpedia.org/3.9/en/
			http://wiki.dbpedia.org/services-resources/datasets/data-set-39/downloads-39
Dataset name	**Dataset filename**	**Amount of triples (not inferred)**	**Amount of unique resources being described**
Images	images_en.nt	193,545,259	16,359,597
Mapping-based Types	instance_types_en.nt		
Titles	labels_en.nt		
Extended Abstracts	long_abstracts_en.nt		
Mapping-based Properties	mappingbased_properties_en.nt		
Inlinks	page_in_link_counts_en.nt		
Short abstracts	short_abstracts_en.nt		
Yago type hierarchy	yago_taxonomy.nt		
Yago types	yago_types.nt		
Articles Categories	article_categories_en.nt		
Categories (Skos)	skos_categories_en.nt		
Raw Infobox Properties	raw_infobox_properties_en.nt		
Raw Infobox Property Definitions	raw_infobox_property_definitions_en.nt		
DBpedia Ontology	dbpedia_3.9.owl		

Appendix C - Quiz databases overview

Quiz databases overview	
Database	**Description**
Actor database (201 MB)	Subset of local DBpedia database, containing information about Norwegian actors born between 1880 and 1955 (database contains triples with those actor resources as subjects).
Amount of triples	**Amount of unique resources being described**
8621	199

Remark	
Some resources are obtained through use of birth year and gender heuristics.	
Current entity type list	
http://dbpedia.org/class/yago/NorwegianActors	
http://dbpedia.org/class/yago/NorwegianFilmActors	
http://dbpedia.org/class/yago/NorwegianTelevisionActors	

Database	**Description**
Musician database (193 MB)	Subset of local DBpedia database, containing information about Norwegian musicians born between 1880 and 1955 (database contains triples with those musician resources as subjects).
Amount of triples	**Amount of unique resources being described**
4091	75

Remark	
Some resources are obtained through use of birth year and gender heuristics.	
Current entity type list	
http://dbpedia.org/class/yago/NorwegianSingers	
http://dbpedia.org/class/yago/NorwegianMaleSingers	
http://dbpedia.org/class/yago/NorwegianFemaleSingers	

Database	**Description**
Movie database (193 MB)	Subset of local DBpedia database, containing information about Norwegian movies released between 1940 and 1980 (database contains triples with those movie resources as subjects).
Amount of triples	**Amount of unique resources being described**
4540	97

Remark	
Some resources are obtained through use of release year heuristic.	
Current entity type list	
http://dbpedia.org/class/yago/NorwegianFilms	
http://dbpedia.org/class/yago/Norwegian-languageFilms	
http://dbpedia.org/resource/Category:Norwegian_films	
http://dbpedia.org/page/Category:Norwegian-language_films	

Database	**Description**
Literature database (209 MB)	Database containing scraped information from selected Wikipedia articles, about literary works written by Norwegian authors born between 1880 and 1955 (database contains triples with literary work resources as subjects).
Amount of triples	**Amount of unique resources being described**
2363	499

Data sources	
Names of relevant authors were found in: https://en.wikipedia.org/wiki/List_of_Norwegian_writers. Literary works for those authors were extracted from individual author articles. Individual author articles are obtained with the tool wget (see section 4.4, paragraph: wget), after extracting a text file with all relevant author links from the former website.	

Appendix D - Document collection and gazetteer overview

Document collections	
List of authors webpage	
Description:	Used by scrape utility to obtain relevant author names (for authors born between 1880 and 1955), which are added to the author gazetteer and literature dataset (as information about literary works), and to find links to individual webpages for those authors (used to download individual author webpages).
Downloaded from:	https://en.wikipedia.org/wiki/List_of_Norwegian_writers
Number of webpages:	1
Individual author webpages	
Description:	Used by scrape utility to obtain literary works for relevant authors (added to literary works gazetteer and literature dataset).
Downloaded from:	Downloaded from links at the list of authors webpage (as mentioned above).
Number of webpages:	147
Last name webpages	
Description:	Used by scrape utility to obtain last names that have been found to occur for more than 200 people in Norway (according to the individual webpages). The last names are added to the last name gazetteer.
Downloaded from:	Last name documents scraped are: https://no.wikipedia.org/wiki/Liste_over_norske_etternavn_på_A to https://no.wikipedia.org/wiki/Liste_over_norske_etternavn_på_Z.
Number of webpages:	29

Gazetteers	
Name	**Entries**
Author	147
Literary work	501
Last name	3443

Appendix E - 2-part session introduction script

Session introduction script part 1 (Before pre-test interview)

Takk for at du/dere har sagt ja til å delta i denne brukertesten. Mitt navn er [navn]. Her er samtykkeskjemaene ditt/deres. Jeg leser resten av denne informasjonen opp for deg/dere ordrett, for å være sikker på at informasjonen og instruksjonene til alle som deltar i testen er de samme.

Hensikten med samtykkeskjemaet er å la hver deltaker bekrefte at de ønsker å delta i denne forskningsstudien for å evaluere Jeanette sin/min masteroppgave, samt å gi litt mer opplysninger om hva denne evalueringen går ut på. Jeg vil nå gå gjennom det som står i skjemaet:

"Det som vil bli evaluert er en prototype av et spørrespill for eldre mennesker, hvor spørsmålene og svaralternativene har blitt laget ved hjelp av tekst fra internett. Meningen er at spørsmålene skal være relatert til personer og arbeider fra deltakernes tidligere liv. Denne quizen kan spilles sammen og/eller alene. For hvert spørsmål vil det være fire svaralternativer, hvor kun ett er det riktige svaret. Vi ønsker å finne ut mer om hvilke kvaliteter en slik quiz kan ha, for mennesker i målgruppen (norske statsborgere over 60 år).

En slik testsesjon som vi skal gjennom i dag består av et innledende og avsluttende intervju, samt test av spørrespillet. Den vil totalt ta maksimum 60 minutter. I intervjuene innhenter vi bakgrunnsopplysninger som kan knyttes til testen, samt informasjon om deltakernes preferanser knyttet til quizen (prototypen). Det som skjer når man spiller quizen vil lagres i en loggfil. Lyd fra testsesjonen vil taes opp og lagres på datamaskin (med passord). Jeg/vi vil ikke lagre personopplysninger som kan identifisere deltakere i denne studien. Masteroppgaven skal etter planen leveres 31. mai. Når den er levert vil loggfilene og lyd-filene destrueres. Det er frivillig å delta i denne evalueringen og du/dere kan når som helst trekke deg/dere dersom du/dere ønsker det."

Da har jeg forklart det som står i samtykkeskjema. Kan du/dere skrive under på samtykkeskjemaet/ene nå?

Så til det som skjer videre: Det jeg ønsker å lære mer om i dag er hva brukere synes om en norsk quiz, det vil si et spørrespill med spørsmål og svar, for eldre mennesker. Jeg ønsker først å stille deg/dere noen spørsmål om quiz-er/spørrespill generelt. Deretter skal jeg forklare deg/dere litt nærmere om hvordan vi skal teste quiz-en.

Har du/dere noen spørsmål til dette?

Slå på båndopptaker og start pre-test interview

Session introduction script part 2 (after pre-test interview)

I løpet av testen vil jeg be deg/dere om å spille 2 runder med quizen. Jeg vil da lese opp spørsmål og svaralternativer til deg/dere. Du/dere skal ikke se på skjermen hvis ikke jeg ber deg/dere om det. Jeg vil kun be deg/dere om å se på skjermen hvis spørsmålet er relatert til et bilde, da kommer jeg til å vise deg/dere bildet. Det er ikke satt noen tidsbegrensning på å svare på spørsmålene, så svar når du/dere er klare for det.

Jeg vil gjerne at du/dere skal tenke høyt når du/dere prøver å finne ut hvilket svar du/dere vil avgi. Bare fortell meg hvilke tanker som dukker opp. Datamaskinen vil også lage en logg over spørsmål og poeng-giving under spillingen.

Vær klar over at det ikke er deg/dere vi tester, men at det er selve quizen. Det gjør ingenting om du/dere svarer feil. Dette er noe som hjelper oss å forstå hvilken vanskelighetsgrad quizen har og hvor godt den fungerer.

Jeg vil at du/dere skal være helt ærlig i dine/deres tilbakemeldinger om quizen du/dere nå skal prøve ut.

Hele testingen vil som nevnt maksimalt ta 60 minutter. Det inkluderer det korte intervjuet vi hadde i sted, og et lite intervju etter at du/dere har prøvd ut quiz-en, hvor jeg vil stille deg/dere noen spørsmål om dine/deres opplevelser med den.

Har du/dere noen spørsmål?

Da kan vi prøve ut quizen.

Start quizen og begynn å spille

Appendix F - Consent form

Forespørsel om deltakelse i forskningsstudie:

"Quiz (spørrespill) for eldre"

Bakgrunn og formål

"Quiz for eldre" er en prototype som er utviklet i tilknytning til masteroppgaven til Jeanette Samuelsen ved Universitetet i Bergen. Oppgaven er veiledet av Weiqin Chen. Prototypen er av et datamaskin-basert spørrespill for eldre mennesker - spørsmål og svaralternativer er generert ved hjelp av tekst på Internett. Tanken er at de som spiller quiz-en skal svare på spørsmål om mennesker og arbeider som kan ha vært betydningsfulle i Norge tidligere i livene deres. For hvert spørsmål vil det være fire svaralternativer, hvor kun ett alternativ er riktig. Quizen kan spilles alene eller sammen med andre.

Denne studien søker å avdekke ulike kvaliteter ved prototypen, slik som hva slags verdi en slik quiz eventuelt kan ha, vurdert av brukere i dens målgruppe (norske statsborgere over 60 år, som helst bør ha bodd i Norge hele livet).

Hva innebærer deltakelse i studien?

Hver testsesjon består av et innledende og et avsluttende intervju, samt brukertesting av den aktuelle prototypen. En sesjon vil totalt ta maksimalt 60 minutter. Intervjuene vil innhente bakgrunnsopplysninger som kan knyttes til selve brukertesten, samt informasjon om deltakernes meninger og preferanser knyttet til prototypen. Under brukertesten vil relevante data om interaksjon med prototypen lagres i en loggfil. Hele sesjonen vil bli tatt opp (kun lyd) og lagret i digital form. Det vil ikke lagres personopplysninger knyttet til deltakerne.

Hva skjer med informasjonen om deg?

Det vil som nevnt ikke lagres personopplysninger knyttet til deltakerne.
Rådataene som samles inn under testsesjonen vil lagres på et passordbeskyttet filområde.
Deltakere i studien vil ikke kunne gjenkjennes i masteroppgaven.
Prosjektet skal etter planen avsluttes 31. mai 2015, da vil rå-dataene som er samlet inn til masteroppgaven destrueres.

Frivillig deltakelse

Det er frivillig å delta i studien, og du kan når som helst trekke ditt samtykke uten å oppgi noen grunn.

Samtykke til deltakelse i studien

Jeg har mottatt informasjon om studien, og er villig til å delta.

(Signert av prosjektdeltaker, dato)