Node search - a graph approach to navigating linked data

Frederik Bo Albeck

Kongens Lyngby 2014
IMM-M.Sc.-2014

Technical University of Denmark
DTU Compute
Department of Applied Mathematics and Computer Science
Matematiktorvet, building 303B,
2800 Kongens Lyngby, Denmark
Phone +45 4525 3351
compute@compute.dtu.dk
www.compute.dtu.dk IMM-M.Sc.-2014

Summary (English)

Finding information on the web is easy thanks to search engines. However there is a growing realisation that new methodology is needed in terms of combining information from different sources (semantic web) and navigation of relational data. Web data usually comes with linked information either as hyper-links or from links that can be inferred from data. In this project we aim to explore navigation of web data by means of graphs. We take Wikipedia as an example because it is an authoritative source with a well-defined link structure (internal hyper-links or DBpedia). The concepts, if successful, can straightforwardly be applied to other types of media such as new paper, dating and media sites. Although these examples come with the added challenge of inferring good link structure, we need to build algorithms to determine what is similar to build the graph. But that will not be part of this project.

Summary (Danish)

At finde information på internettet er let på grund af søgemaskiner. Imidlertid er der en voksende opfattelse af, at nye metoder er nødvendige, når det drejer sig om at kombinere information fra forskellige kilder (Semantic Web) og navigering i relaterede data. Web data kommer sædvanligvis med linket information enten som hyperlinks eller som links, der kan blive udledt fra data. I dette projekt er det vort mål at undersøge navigationen af Web data ved hjælp af grafer. Vi bruger Wikipedia som eksempel, på grundt af dets velansette betydning og med en veldefineret linkstruktur (interne hyperlinks eller DBpedia). Metoderne kan, hvis det lykkes, umiddelbart blive overført til andre typer af medier så som aviser, dating og medie sites. Selv om disse eksempler kommer med yderligere udfordringer i form af udlede passende linkstruktur, er det nødvendigt at bygge algoritmer for at bestemme hvad det kræver så det svarer til at bygge grafen. Men det vil ikke være en del af dette projekt.

Preface

This thesis was prepared at the department of Informatics and Mathematical Modelling at the Technical University of Denmark in fulfilment of the requirements for acquiring an M.Sc. in Digital Media Engineering.

The thesis deals with linked data navigation.

The thesis consists of design and implementation of an iPad application including backend.

Lyngby, 15-August-2014

Frederik Bo Albeck

Acknowledgements

I would like to thank my supervisor Ole Winther for giving me the opportunity to use the WilkyMay application as project for my master thesis project and for the help and guidance throughout the process.

In addition I would like to thank the projects co-supervisors Anders Borum, Dan Svenstrup and Mads Hallas for their help. Anders Borum for his guidance with the developing of the iPad application. Dan Svenstrup for the feature ideas of the app and help with data mining for the help with the database design.

At last I would like to thank the test subjects helping me with testing of the application and my family for help and support during the project.

Contents

Chapter 1

Introduction

The internet it to day a big part of many peoples lives and each day it grows and becomes more complicated. This will mean that more and more data is accessible on the internet. But how is it possible to get the desired answers to the questions that you might have? This is where the semantic web comes in. The semantic web builds on the idea to make the internet we have today, based on unstructured and semi-structured documents, to be a "web of data": a connected web where there are no differences between data sources to make the use of linkage possible. Search engines are a good tool for finding data and can often return the desired results. But search engines are limited in the way, that they can search in a lot of data from a lot of sources, but they have difficulties comparing the sources and thereby present an even more accurate result. If the data on the web was uniformed, the search engines could be even better than they are today and the user could query the search engines with more advanced inputs. The idea of using linked data, is also to make it possible explorer and discover new data. This is where the need for tools to explore the "web of data" is essential.

1.1 Project description

The idea for this project was inspired by the WilkyMay application from Milky-Way Labs [Mil] for visualizing linked data. The extensions compared to the Milkyway application is an implementation of PageRank scores to handle sorting and of methods to navigate through the layers of linked data in the graph visualization.

1.2 Project delimination

The main focus is to review suitable visualization tools for mobile applications and deliver a functioning iPad application. This implies development of an iPad application with backend, that can create interactive graph visualizations of linked data. The use of linked data in this project is constrained to use the Wikipedia link structure from the DBpedia project. For evaluation of the design and implementation an usability test will be conducted.

Chapter 2

Analysis and Theory

This chapter describes the analysis of subjects related to the project and presents the basic theory. This implies a description of Linked data and how it is used, as well of visualization techniques and tools, furthermore the use of graph theory and PageRank algorithm.

2.1 Linked Data

Linked data is a term used in the semantic web, describing relations between different datasets. Biezer *et al.:* [BHBL09] describe linked data as *"...Linked Data is simply about using the Web to create typed links between data from different sources."*
The idea of linked data is to make the data on the web accessible to everyone by making links between structured data. This is done by making the data machine-readable, which includes setting some usable standards for linking the data using the RDF framework (Resource Description Framework) and the HTTP protocol (Hypertext Transfer Protocol). By doing so, the user can easily perform advanced queries on a collection of structured data from different resources at once.

Tim Berners-Lee, founder of the World Wide Web, has described a set of rules for publishing data on the web as linked data. [w3cb]

1. Use URIs as names for things
2. Use HTTP URIs, so that people can look up those names.
3. When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL)
4. Include links to other URIs, so that they can discover more things.

For naming linked data objects it is advisable to use URI (Uniform Resource Identifier) and publish the linked data objects using the HTTP protocol which enables everyone to access. URI functions as the identifier of the resource, which can be used to look for similar things. Furthermore it is advised to provide useful information and include URIs to discover more linked data. [HB11]

These rules should be used as a guideline every time data is published for the public.

Linked data is presented in the RDF format (Resource Description Framework) with URI as identifier for the objects. Linked data is normally stored as RDF triple's.[W3Ca] A triple consists of a subject, a predicate and object. **Subjects** are the "thing" the RDF triple discribes. **Predicate** represents the type of relationship between the subject and the object. **Objects** are the "target" in the triple. An object can be another resource or a value describing the subject.

So linked data can for example be a way of describing the relationship between objects in a data set. Below is an example of linked data, where the objects are the city "Copenhagen" and the country "Denmark":

Copenhagen is a city in Denmark

If we are to denote this in a RDF triple then Copenhagen would be the subject, Denmark the object and "is a city in" the predicate. The predicate would normally have a single word describing the relationship such as "City" or "Capital".

There is a lot of different projects working with making data available for everyone. These range from geographical database as **Geonames**, a geographical database, with linked data from more than 10 million geographical places to a

movie database as **Linkedmdb** with information about more than 80000 movies and 50000 actors. One of the larger projects is the DBpedia project.

DBpedia is an open source project started in 2007 by a group of people from the Free University of Berlin and University of Leipzig. [Pro] The goal with DBpedia described in *DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia* by *Lehmann et al.*:[LIJ+14] is to extract structured, multilingual data from Wikipedia and to make it available for everyone. DBpedia has a list of extractors describing the type of data they acquire from Wikipedia. This includes information such as abstracts, article categories, infoboxes and page links. Abstracts and article categories contain the short description of the page. The DBpedia extraction focuses a lot on extracting data from the infoboxes on Wikipedia pages. These boxes contain valuable data such as dates, coordinates, numbers, links etc. The page links contain information of links to other Wikipedia articles.

The two tables below in Figure 2.1 show the relationship between two pages in the DBpedia dataset "Copenhagen" and "Denmark". Each page has a title, this title is used as a name for the object. Each page has information about the given object and this can be used as a set of properties for the object and each property has a link to the corresponding page. The arrow represents the link between the two pages of Copenhagen and Denmark.

Title	**Copenhagen**
Country	Denmark
Region	Hovedstaden

Title	**Denmark**
Capital	Copenhagen
Language	Danish

Figure 2.1: Tables showing relationship between Wikipedia pages.

2.2 Graph theory

Graph theory is widely used in areas from computer science to topics of economics and social networks. An example of where graph theory is used is Facebook. They use network theory for their Facebook Graph API, which allow users to discover and work with the available social network data. [Fac]

Basically a graph $G = (V, E)$ is a set of V vertices (nodes) connected by a set of E edges. The nodes and edges describe the relationship between objects in a

network. [DJ10] The nodes in a network represent each item in a data set and are normally visualized as circles. The edges in a network represent the type and how nodes in a network are connected. These are normally presented as lines.

Normally a connection between nodes in a network is symmetric, which means that there exists a connection from node A to node B and vice versa. If the connection only is in one direction, from node A to node B, the connection is asymmetric. This can be visualized as undirected and directed graphs. See Figure 2.2 and Figure 2.3.

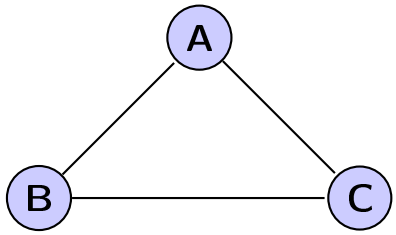


Figure 2.2: Undirected Network

Figure 2.2 shows a network with a set of nodes A, B and C, with symmetric connectivity.

Figure 2.3 shows a network with a set of nodes D, E and F, with asymmetric connectivity due to the one-way direction of the edges.

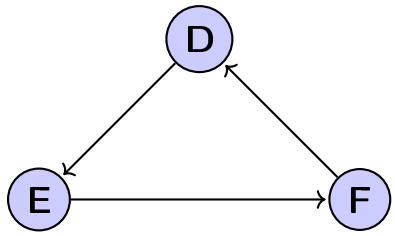


Figure 2.3: Directed Network

For both undirected and directed graphs it applies that they can be weighted. Weighted graphs mean that every edge in the graph has a value indicating the edge state. This can for example be used for deciding traversal path in a network with geographic coordinates.

In graph theory the measuring of objects centrality in a graph is essential to identify important nodes. There exists different algorithms for measuring the

centrality of objects or nodes. *Brandes et al.:* describes some of these graph centralities. [BE05]

Degree centrality is the most simple version. This centrality is calculated by the degree $d(v)$ of v, which means how many edges are connected to the given vertex. When working with directed graphs there are two centrality values, the in-degree and out-degree. Basically degree centrality counts the number of neighbours and the vertex with most neighbours are the most important in the graph.

Closeness centrality uses the distance between vertices (nodes) in a graph to calculate the closeness of vertex. For each node in a network a distance sum is calculated to every other node. The node with the smallest distance to every other node is the most central node.

Betweenness centrality in a graph is calculated by how often a vertex (node) has been used as a connection point. The calculation counts how many shortest paths between every node in a network that passes through a given node.

Eigenvector centrality calculates a vertex (node) influence in a graph. The calculation is based on, that each node has a score and is weighted according to its neighbours score. [Ruh00]

Pagerank centrality is an algorithm that was created by the founders of Google and is a key part of the Google search engine. The Pagerank algorithm is based on the Eigenvector centrality and is used to calculate the centrality of websites in a network. [PBMW99] This is among other things how Google calculates their search engine results. The idea with PageRank is that every page in a network that links to another page is recommending the page and thereby gives the linked page higher importance and rank. Every link to a given page has different weight according to the importance of the source page. Furthermore the page that links to other pages can not give the full weight of its own value to other pages. This means that if an important page links to other pages it gives them a higher importance, but if it links to a lot of other pages, fx. thousands of pages, then the weight of its out-links is diminished. If a page only has a few in-links it is most likely that it has less importance than a page with a lot of in-links, but that depends due to fact of the importance of the in-links. *Anderson and Ekström, 2004* [AE04] describes in *Investigating Google's PageRank algorithm* PageRank as:
This idea can be seen as a way of calculating the importance of pages by voting for them. Each link is viewed as a vote - a de facto recommendation for the importance of a page - whatever reasons the page has for linking to a specific page.

The equation presented below shows how to calculate the PageRank for given web page.

$$R(u) = c \sum_{v \in B_u} \frac{R(v)}{N(v)} \tag{2.1}$$

The PageRank algorithm works by calculating the rank for object compared to others - for example a web page in a network. Before the calculation is performed none of the objects in a network have a PageRank score, this is therefore necessary to assign every object with an estimate of its PageRank. This means that a initial PageRank doesn't match to the actual rank of the page. By using a iterative computation algorithm it is possible to give an approximation of the PageRank for each page. [eFa]

A simple example of calculating Pagerank can be visualized with the use of a limit amount of pages in a networks as the following.

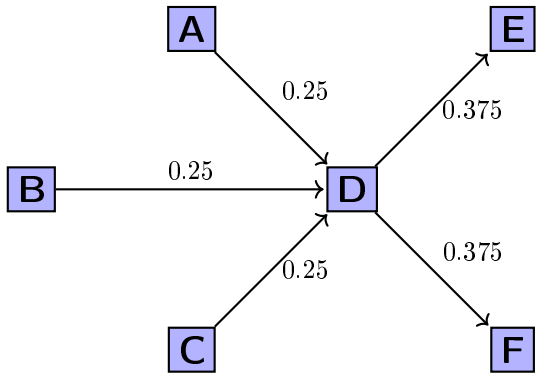


Figure 2.4: PageRank computation example

Figure 2.4 presents a very simple example of the three pages, A,B and C, that links to page D. Each page has a PageRank of 0.25 and with only one out-link they weight the importance of D with 0.25 each. This means that D's has three in-links and its PageRank can be calculated as $PR(D) = PR(A) + PR(B) + PR(C)$. Thereby D's PageRank is $PR(D) = 0.25 + 0.25 + 0.25 = 0.75$. Page D has two out-links to page E and F and it weights each page with its PageRank divided by the number of out-links. $PR(E) = PR(F) = PR(D)/N = 0.75/2 = 0.375$.

2.3 Visualization

For the normal user a huge amount of data presented as a big pile of unsorted objects with a lot of unnecessary information, can be very difficult to understand. This can be long lists with a lot of attributes and numbers describing each item. This can be poorly presented data with odd colour, positioning, font size etc. All of this apply when working with linked data. A problem with graph visualization and linked data is the amount of data that is to be handled. The problems with navigating in datasets of big sizes are discussed by *Dokulil and Katrenikova RDF Visualization - Thinking Big* [DK09]
Dokulil and Katrenikova describe the problems of selecting start node for their application. Among other things the problem with limited resources of the hardware that computes the visualization. They discuss this problem in their description of selecting the starting node in a large data set. *One important aspect to mind is the size of the displayed graph – it is often not possible or viable to load and analyze the whole data when the visualizer starts. The size of the data may exceed the available memory or loading might take too long.* This is written in accordance to processing large graphs at once, but it can still be assigned to problems working with smaller graphs and the response of the application.

2.3.1 Tools

For visualization of data, there already exists a great variety of available codes and libraries. Some of them are professional tools that require some kind of payment to use, but there also exist free alternatives.

The quality and available features in the visualizations tools, ranging from very simple to very complex and expensive ones. For this project it was necessary to study suitable tools for the visualization. The tools were required to handle interactive graphs visualization which includes visualizing nodes, edges and labels. Furthermore it was necessary to handle zooming features or the possibility to implement this.

Visualizing linked data on websites or mobile entities requires a code that can visualize data. This can be obtained with the use of visualization libraries which in this case can visualize graph networks. There exist some graph visualization libraries, but for the mobile platforms it is quite limited with options. Especially when it comes to graph networks, the majority of the libraries can only handle normal graphs as bar or line charts [Goob]

Visualization wise, there exist no proper native visualization libraries for developing visualizations of linked data on mobile platforms such as Android or iOS. However this can be accomplished with the use of some alternative JavaScript libraries such as Infoviz JavaScript [Inf], Prefuse [Pre], Sigma JavaScript [Jac] and D3 JavaScript [Bos].

2.4 Related work

In the following some of the related work that features visualization and/or navigation of graph networks and linked data are described.

Lodlive Project
The LodLive project [CMA] aims to promote the linked-open-data philosophy. They have created a tool that can visualize and browse available RDF resources. The LodLive website consists of an interactive graph visualization with a lot of features. LodLive features an extensive use of the DBpedia source, with several language and furthermore a map and image tab for the presented nodes. Connected nodes are visualized in a semi-circle around the center node and when clicking on one it will expand. Navigating in connected nodes are performed by clicking on little arrows at the end of the semi-circle. This does that the semi-circle changes content to other linked nodes. The project and how to use it, is further described in *LodLive, exploring the Web of Data* [CMA12].

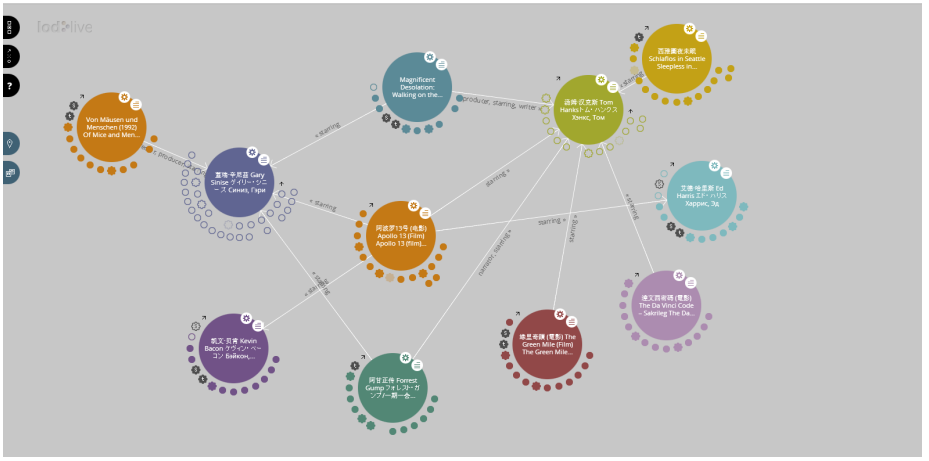


Figure 2.5: Screenshot of the Lodlive project

TouchGraph Navigator

TouchGraph navigator is a tool for visualizing and navigate linked data. [Tou] This is a professional tool that you have to pay for to use. They have a demonstration web application that shows the tools potentials. The demo application lets the user explore a network of connected websites. The feeling with the visualization is smooth and quick. They have tried to make it more vivid by implementing a gravity behaviour between the nodes which is visible when moving the nodes around. With use of the gravity behaviour the nodes keep a calculated distance between its neighbours and doesn't get placed on top of each other.

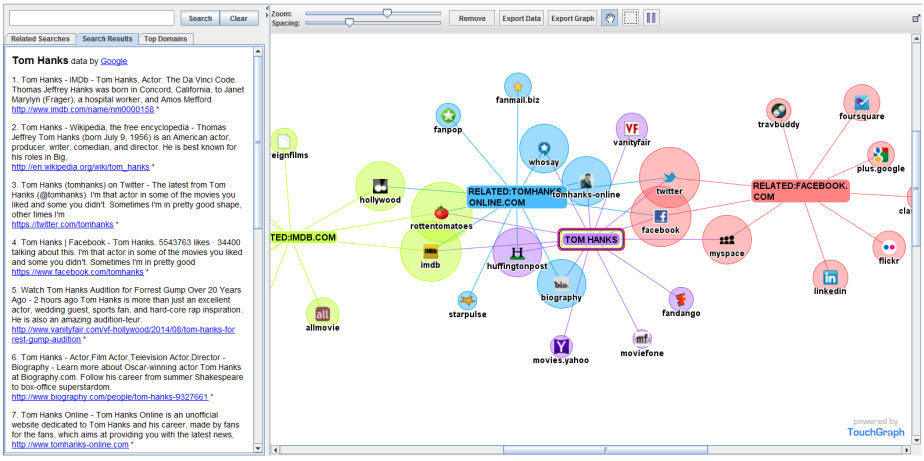


Figure 2.6: Screenshot of the Touchgraph web application

Linked Jazz

Linked Jazz is a project exploring the potential of linked open data with the use of relationship between jazz musicians.[lin] The interactive visualization shows the social networks between jazz musicians throughout the history. It lets the user explore the network by clicking on nodes to see connections and get more information. This project use a very nice graph visualization with different colour edge and presents a small info box with Wikipedia link and options to listen to the selected musician. Furthermore it highlights the connected musician by making the edges wider and hide not connected nodes and edges by making them transparent.

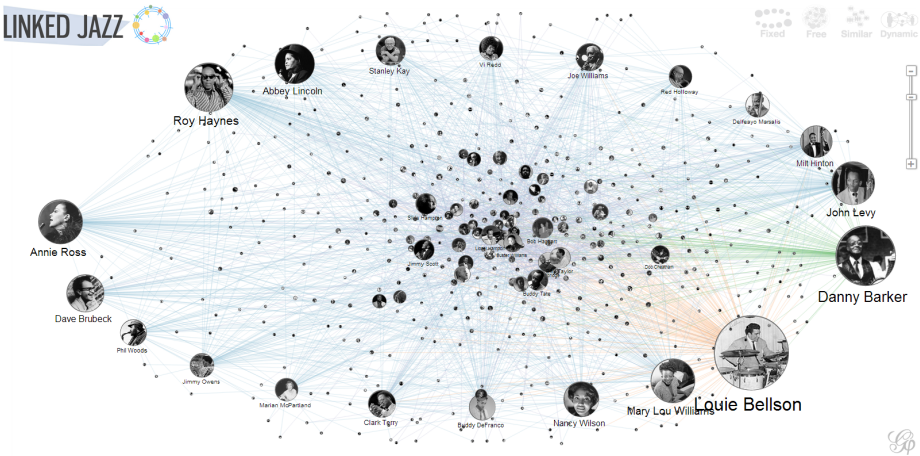


Figure 2.7: Screenshot of the Linked Jazz project

These related works have all worked with visualization of graphs and linked data, but are limited compared to the goal in this project: namely combining the visualization and navigation of linked data and furthermore work properly on tablet devices. The Lodlive project is the most advanced tool that offers a huge amount of options and lets the user interacts with the graphs. Unfortunately it feels a bit crowded with options and it is developed for desktop browser and thereby isn't optimized for use with a tablet.
TouchGraph is more simple and is quite limited compared to the features of LodLive, however, it contains of a visualization and a side panel that can give an overview of the selected objects and furthermore let the user search for other topics. The demo application is developed as a Java applet and it is therefore not possible to use with an iPad. The linked Jazz project offers a nice visualization, instead of just displaying a selection of links, it displays the whole graph. This looks good but it is very difficult to find specific nodes. Linked Jazz is developed for desktop browsers and unfortunately it works very poorly on a tablet.

Chapter 3

Design

This chapter describes the design process of the application and backend. Furthermore a walkthrough of the necessary requirements for the application and backend is described.

As mentioned in section 1.1 the goal for the project was to explore the possibilities in visualizing and navigating in linked data. The idea was to develop an application as an extension to the WilkyMay application, but with the feature of visualizing and navigating in more linked data. The main features were discussed in the first meetings with the project supervisor Ole Winther, who is a part of the MilkyWay project. Later on meetings with assistant supervisors and interviews were used to find and discuss feature changes.

The design of the application is primarily based on the existing project by MilkyWay Labs. The data used for the visualization was internal Wikipedia links from DBpedia. For presenting more data it was discussed that it should be prioritized by using PageRank.

3.1 Application

The ideas for the application were at the beginning to be a multiplatform development for both Android [Gooa] and iOS [App]. Later in the development process, it was changed to only be developed for iOS iPad devices due to the delimination of the project.

As mentioned in section 1.1 the application is based on the MilkyWay Labs application for visualizing and interacting with the data. The main goal was to explore the possibilities for navigating in data with many connected links.

3.1.1 Visualization

It was a problem to find suitable visualizations tools that met the requirements of this project with interactive graphs working on iPad devices. After discussions with supervisor Ole Winther and co-supervisor Anders Borum it was decided that the visualization was to be developed from scratch. This means that it was required to develop an application that could create interactive graph visualization. The visualization should be able to visualize graphs with nodes, edges and labels.

3.1.2 Navigation

For the navigation of the linked data the visualization needed to be interactive. The application had to fulfil the features of the WilkyMay application such as moving and selecting nodes. Furthermore it had to be able to move the entire view by panning. At last but not at least, it was required to let the user be able to navigate among the linked data by using the zoom functionality.

Figure 3.1 shows the problem we are trying to solve with navigation. Links between objects in linked data can exceed a count that is possible to display at once. The figure is created to show that if an object has more than for example 8 links to other nodes it would be difficult to display the nodes without the view getting to crowded. You could discuss that the amount of 8 nodes were to many or to few, but this was chosen based on our initial discussions and due to delimitation of the project.

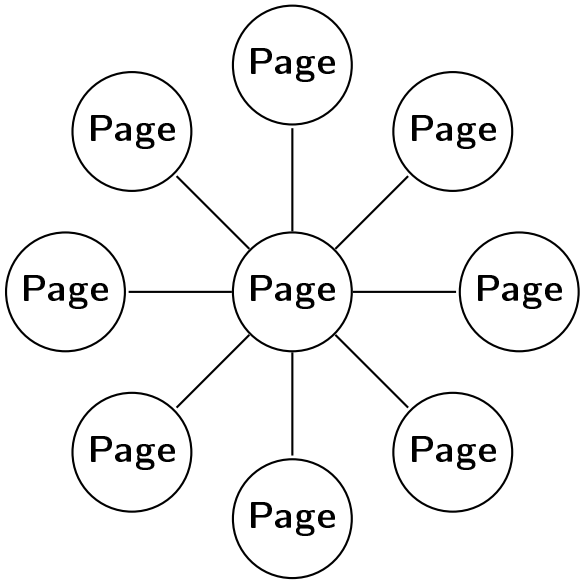


Figure 3.1: Navigating linked data

3.2 Backend

The design of the backend was discussed in the beginning of the development process to create a base for the project. The initial design of the database was discussed in startup meetings with the co-supervisor Dan Svenstrup.

Dan Svenstrup proposed some ideas on how the database could be implemented using the datasets from DBpedia. Initial setup were implemented using a small Windows server with SQL queries using varchar searches. These setups performed very slowly and gave poor results.

For the project the data from DBpedia was limited by only using internal Wikipedia links between pages. DBpedia contains a lot of other data such as population size, geographical coordinates and external links to other interesting datasets such as IMDB, ect. But this limit was chosen as the DBpedia dataset for the Wikipedia structure in it self offers a huge amount of data and would be more than enough to present the project application potential and features. It was further chosen to sort the linked data with the use PageRank. This would be better for the use of the application when the most popular nodes was presented first. This means that the database had to be implemented in such that it could store the internal Wikipedia link from DBpedia and a PageRank score

for each of Wikipedia pages.

3.3 Requirements

Based on the project goals and talks with Ole Winther and Dan Svenstrup a set of requirements was clarified. Table 3.1 shows the requirement of the application with description and priority.

ID	Name	Description	Priority
1	Basic visualization	The application shall present a visualization of the graph with nodes, edges and labels.	High
2	Data	The application shall retrieve and visualize data from the web service.	High
3	Interaction	The user should be able to move nodes and panning view from user interaction.	High
4	Zoom navigation	The user should be able to navigate through the linked data.	High
5	Web view	The application shall present a view with Wikipedia information for the selected object.	Medium
6	Animation	The application shall animate moving objects. Feedback animation.	Low
7	Advanced visualization	The application shall only show closest visisted nodes.	Low
8	Search	The user should be able to search for other subjects.	Low

Table 3.1: Prioritized requirements

Chapter 4

Implementation

In this chapter the implementation of the application, database and web service is describe. The development process was performed as an iterative process, which means that several first drafts were implemented to get acquainted with the technology and frameworks.
Figure 4.1 displays a diagram of the communication between the client and the server. The client is in this project the iPad application, but it can actually be any device that can send http (Hyper Text Transfer Protocol) requests and receive JSON objects. JSON (JavaScript Object Notation) is an open standard format widely used for parsing data between servers and clients. JSON is a lightweight format and language independent, thus making it simple to work with.

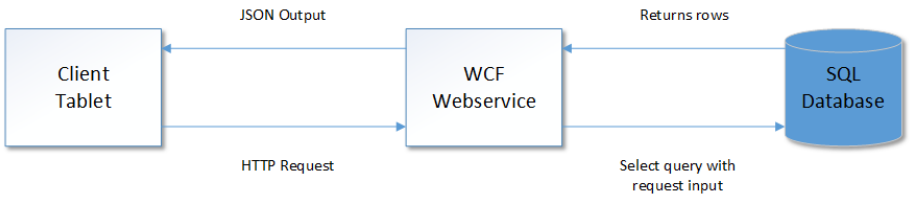


Figure 4.1: Client Web service diagram

The client application implementation is explained in the following section, fol-

lowed by a section describing the server.

4.1 Application

The application developed for this thesis project, is developed as an application for Apple's iPad. This was accomplished by using the Apple's iOS developer resources and the Stackoverflow community which both contains huge amount of information about the core foundation framework used in iOS development.

The application is developed as an "singleview" application which implies that there are no other pages the user have to navigate between. The application consist of main view for the interactive graph visualization and a sub view for the presentation of the Wikipedia page.

4.1.1 Main application features

The following describes the main features from the design requirements that was implemented in the iPad app.

4.1.2 Main View

The main view of the applications presented in Figure 4.2 consists of a scroll view that enables the user to scroll vertically and horizontally in a large view. This view contains of the graph visualization with nodes and edges, representing the Wikipedia pages as nodes and links between them as edges. The application uses an UIScrollView to handle pan and pinch zoom interaction from the user. On top of the scroll view the nodes and edges are located. The nodes are implemented using the standard UIView with touch handlers for dragging them around. The edges are implemented using the iOS framework CaShapeLayer, which is a lightweight layer that doesn't use a lot of memory and is very useful for drawing.

4.1.2.1 Nodes

To represent a page node in the linked data network, a custom UIView class called NodeView was implemented. With this custom UIView it is possible

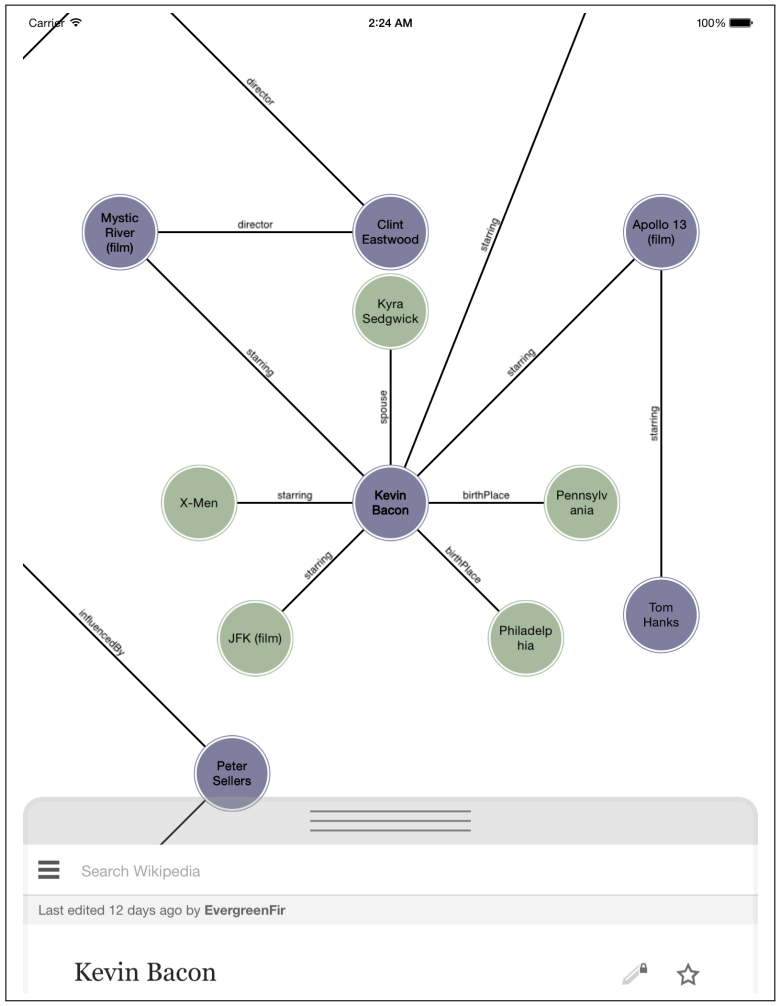


Figure 4.2: Screenshot of the iPad application

to easily change the appearance of each node indiviually and implement touch interaction. The touch interaction is used when the user wants to move or tap on a node. For intercept tap on a node an UITapRecognizer was implemented, which only reacts when a node is tapped, not dragged around.

For intercepting dragging of a node an UIResponder was implemented. This was used to calculate the movement of the node along with the interaction of the user and furthermore to calculate the animation when the interaction was stopped to give the "elastic" effect of an object that had been moved.

4.1.2.2 Edges

To represent the edges in the network a CaShapeLayer was implemented. The interaction between the nodes and edges is handled by observers. For each edge there is added an observer which monitor the position of an edge end nodes. When one of the end nodes has changed position the observer notifies the edge to update its path.

CATextLayer was used to present the edge labels. Initial attempts with "normal" UILabels couldn't handle ongoing updates when the edges moved according to the nodes. This caused the visualization to be out of synchronization and it didn't looked good.

4.1.2.3 Navigation

For navigating and interacting with graph in the main view, tap and drag features is implemented in the nodes and panning and pinch zoom features is implemented in the scroll view. When a node is selected with a tap, the application is adding new nodes to the view. It is therefore necessary to calculate the position of the nodes or else they will most likely be placed on top of existing nodes and then be an inconvenience to the user. Therefore when new nodes need to be added, it would be calculated where the nearest rectangle with size of the new nodes was placed in a circle so it wouldn't overlap existing nodes. This algorithm works by creating a rectangle and takes the nodes closest to the selected node. The algorithm then checks if there are any nodes inside this rectangle and if so the rectangle is moved and the nodes in the surrounded areas are investigated until an empty space is found.

When the pinch zoom feature is used the application removes the neighbour nodes for the selected node and requests a new layer of linked nodes from the web service. For visualizing changes in neighbour nodes, the positioning of the nodes rotated with 15 degrees in the circle.

Some animations were implemented to make the interaction with application more vivid. This is shown with bouncing animation when a node is dragged and released and furthermore a bouncing animation of the web view when selected.

4.1.3 Data Request

Connection to the web service was handled in separate thread. It is important not to cause the application to hold up and make the user wait. Therefore an implementation of an asynchronous NSURLreguest was used. The asynchronous NSURLreguest can handle web service call without the main thread having to handle how long a call takes. When a reply is received from the service, a NSNotification is used to call the main view controller in the main thread.

4.1.4 Sub View

The sub view consists of a web view and is presented in Figure 4.3. The web view is implemented to show the actual content of the selected Wikipedia page in the network.

The web view is implemented as sub view of the main view and it is placed on top of the main view and can be dragged up from the bottom. Each time a node is tapped the graph visualizes the new node and its connected nodes in the main view, the web view loads the content for the tapped node. The web view is implemented inside an UIView container, so the user can swipe it up from below.

The web view was implemented with a feature that intercepts when a link is tapped in the web view. If the links url points to another Wikipedia page, the title of the link is used to make a call to the web service. If the link exists in the database, the main view is cleared and the node and the linked nodes for the Wikipedia page matching the clicked link are visualized in the main view.

4.1.5 Implementation Improvements

After the usability test of the application a few features were added or improved in the application.

A slider and a label were implemented to improve the zoom navigation and furthermore notify the user about which layer of linked nodes that was visualized. This should furthermore engage the users to use the slider and thereby access more linked nodes. The old zoom feature were not removed and when used the slider updates according to the selected layer.

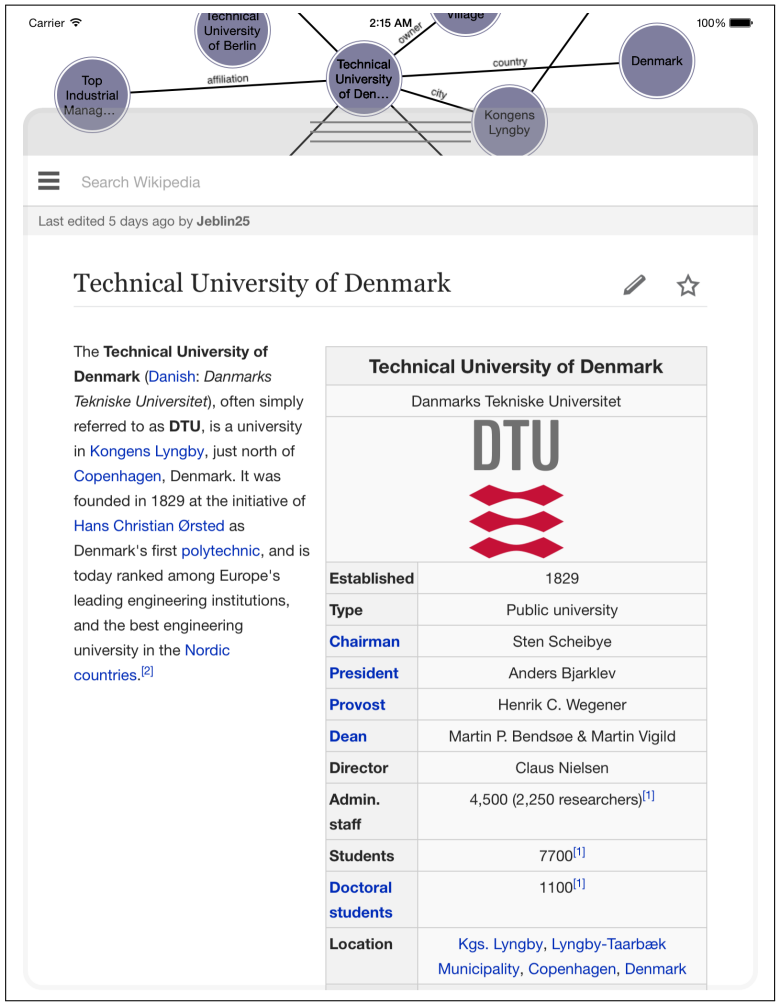


Figure 4.3: Screenshot of the subview containing the Wikipedia page

Interaction with the web view caused some usability problems when only some part of the view could intercept the user interaction. Therefore this feature was further improved to handle interaction in the search field and selecting pages in a drop down list.

4.2 Backend

The backend consists of and server with a database for storing the linked data and a web service for handling requests from client devices. The database is an SQL database running on a data center with Microsoft Server 2012 [Mica] and Microsoft SQL Server 2012 [Micb].

The design of the database was changed several times throughout the process. This was due to changes of the idea for the project, optimization of the query request time and the use of available data from DBpedia.

The implemented database consists of three tables and the structure of the database is presented in the following tables.

WikiPageId	
PageId	bigint
PageName	varchar

Table 4.1: WikiPageId

WikiRank	
PageId	bigint
Links	bigint
Score	deciaml

Table 4.2: WikiRanks

WikiLinks	
Id	bigint
SubjectId	bigint
Predicate	varchar
ObjectId	bigint

Table 4.3: WikiLink

This structure was chosen to ease the import of data and prepare for possible changes.

Implementation of the SQL queries caused some difficulties and had to be optimized multiple times. Optimization of an sql query is an important part of the development and was very applicable for this project. If the SQL query is wrong or poorly implemented the service either has nothing to return or has very poor performance. Because of a specific term in the DBpedia data structure either

can be a subject or an object, it was necessary for this query to lookup the table with the Wikipedia links two times. This was first implemented with the result that each query execution was more than a minute to complete, which was only due to use of the "or" keyword when combining lookup in multiple tables with inner join.

The SQL query used in the web service to find Wikipedia links for a selected node is presented in Appendix A and has an execution time in a matter of seconds. The SQL snippet shows how the query was implemented to look both subjects and objects in the DBpedia data structure.

The web service application was implemented using the WCF framework (Windows Communication Framework). This web service handles the asynchronous NSURL requests made from the iPad application and returns the result from the SQL database in a JSON format.

The web service takes two parameters, a page id and a level. The page id parameter specifies which Wikipedia page the service call uses to find Wikipedia internal links. The level parameter specifies on which level the service should return the result. This parameter is used for the pinch zoom navigation to get the more layer of the links.

The web service outputs a JSON object which is shown in Appendix B. This shows an example with the actor Tom Hanks and the top 8 most popular Wikipedia paged linked with Tom Hanks based on PageRank score.

Chapter 5

Evaluation

In this chapter the evaluation of the project and the developed iPad application is described. This includes an usability test and a performance test of the application and backend. Furthermore it includes a presentation of the alternative prototypes and how the prototype testing and interviews were carried out. The result of the test will follow in the next chapter.

The testing of the iPad application, was carried out throughout the development process, to secure that the application features were implemented correctly. That implies ongoing tests and error correction.

A couple of usability test was carried out, to discover usability problems and to compare the initial design for navigating and visualizing with other prototypes presenting the data.

The test subjects for the usability test were as far as possible chosen in such to cover a wide spectrum of possible users. The test subject were chosen based from different age groups, gender and use of technology such as search engines and tablet use.

Prototypes to alternative ways of presenting the linked data visualization and navigation were created to show the user different ways of presenting the linked data. They were developed in such to present the test subjects with different

approaches in order to get them to comment on the application features. They can't comment on things they don't know about.

5.1 Prototypes

There are many ways to develop and present prototypes for testing a product. This depends on the type of product and the data one would like to obtain. Prototypes can be in different stages of complexity from low-fidelity to high-fidelity prototypes. This is described in 1996 by Rudd *et al: Low vs. High-fidelity Prototyping Debate* [RSI96]. Each has advantages over the other. Low-fidelity prototypes are fast and easy to develop and are good for creating a proof-of-concept. The drawbacks of a low-fidelity prototype are poor detail of the product and limit user interaction. High-fidelity prototypes are fully interactive and can give the user a feeling of the final product with all functionalities implemented. However, this takes a lot time to develop compared to the low-fidelity prototype.

Normally prototypes are used in the beginning of a development process. In this project three interactive low-fidelity prototypes were developed with use of a prototype tool. At first a test with a low-fidelity prototype, created with pen and paper, was conducted. The result of this was not desirable and the test subject was clearly confused about the prototypes appearances and the intended features. This initial test showed that it was necessary to develop a more advanced prototype. There exists a wide range of tools for prototyping mobile applications with different qualities and features. For this project the tool Flinto [Fli] was chosen. Flinto is a prototyping tool, that enables the user to create interactive prototypes from images and presenting them on smart phones or tablets. However, the tool is limited regarding the interaction such as panning and zooming in visualization where it isn't possible. In that case it would demand more advanced tools or to develop an actual application as the prototype.

The three alternative prototypes developed for the usability testing is presented in Figure 5.1 and in Appendix C.

Figure 5.1: Alternative prototypes

The prototypes are developed with the idea to present the user with different ways of presenting and interacting with linked data.

Prototype 1 is the "normal" way of presenting the data in an application with lists grouped after the types of links for example actor, director, writer ect.

Prototype 2 is an updated version of prototype 1 showing one combined list for all the connections sorted by the PageRank score.

Prototype 3 is a further updated version, but with a combination of a list and visualization of the connections. The visualization shows the previous visited page in the application and lets the user easily jump right back to a previous node.

5.2 Test subjects

For the usability test of the application and prototypes a few test subjects were chosen. The application was developed for everyone to use and thereby no specific target audience. The test subjects were chosen from a wide range as possible regarding to age, gender, occupation and use of tablet devices.

Table 5.1 shows the subjects used for the test and interview.

Test subjects		
Test subject	Profession	Use of tablet and online media.
Male 25	construction engineer	Owns an iPad and uses it daily at work. Uses Google daily and Wikipedia a few times a week.
Male 27	business consultant	Owns an iPad and uses it daily. Uses Wikipedia regularly.
Male 27	developer	Has owned an iPad. Use Google at least once a day. Wikipedia only a few times a week.
Female 35	teacher/student	Doesn't own an iPad. Use Google regular. Has used Wikipedia a lot during projects.
Male 69	pensioner	Owns an iPad and uses it regularly. Rarely uses Google. Knows about Wikipedia, but have only used it a couple of times total.
Female 65	psychologist	Owns an iPad and uses it sometimes at work. Uses Google and Wikipedia few times a month.

Table 5.1: Test subjects

5.3 Interview

When developing an interview plan it is important to think about what you want to obtain with the interview. There are four types of interview forms in general. [SRP07] Each interview type has different strengths and weaknesses based on the given tasks and furthermore lets the user define how much control they want for the interview.

5.3.1 Interview types

Unstructured interview is close to be a casual conversation between the interviewer and interviewee. This type is useful when the goal is to obtain as much data as possible and to let the interviewee contribute with his own opinions and thoughts. The questions are open and lets the interviewee answer more detailed. A problem with this type can be that the interview derails and the "conversation" changes subject and thereby the time is not used best and furthermore the data obtained can contain "bad" data that isn't useful for the

intended purpose of the interview.

Structured interview is in contrast to the unstructured used for obtaining quick and precise answers. This is done by presenting closed questions, which can be as questionnaires with a selection of predefined answers. When conducting structured interviews the same questions are used and in the same order every time. By doing so the interviewer can easily obtain comparable data from many interviewees.

Semi-structured interview is a combination of unstructured and structured interview. It can contain both open and closed questions usually start with a closed question for precise yes/no answer and then an open question to let the interviewee elaborate on his answer.

Group interview or focus groups are an interview form with groups of 3-10 people. This lets the interviewees talk with each other and therefore give feedback to the interviewer. A problem with this is that some people may be influenced or convinced by other views of the subject which can lead to that they doesn't contributes with their initial points of view.

5.4 Usability Test

The usability test consists of a test with the test subjects use of the alternative prototype and application and long with a semi-structured interview. The test included a lists of tasks the test subjects were to complete for each prototype. These tasks were created in a way that the user experiences the basic features of the prototypes and gets an understanding of the application, both visualization and navigation wise.

After the tasks were performed for the prototype the test subjects were presented for the final application. The user was given a list of tasks similar to them in the prototype so the subjects could compare the experiences with prototypes contra the final application.

After the test the subjects were given a couple of open questions regarding the prototypes and the application. The ideas with the test and the interview questions were to get feedback from the subjects about where the applications were helpful or where they got lost. What they thought of the application compared to the prototype, the visualization, the navigation and what features they would suggest to improve the application.

5.5 Performance Test

The application is developed as a single view application and has no problems performance wise it self. Early builds of the application did suffer from slow animation.

These problems have been eliminated and the animation of particular the simulations runs smoothly and doesn't let application hang for a long time which will disallow user interaction.

The performance problem of the application rely in the communication with the server and the capability of the server to handle the requests. A performance test was conducted to find the response times when the application was to communicate with the server. 8 Wikipedia pages with different amount of links were tested by logging response time. The duration of the response times was logged from the web service call and until the graph was drawn in the application.

The internet connection may have a influence on this, but the backend was implemented to only transfer data for 8 nodes per request and therefore this hasn't been tested in this evaluation.

Chapter 6

Results

6.1 Usability test

The results of the usability test with the selected test subjects are presented in section D.1. This includes the tasks for the prototype and the application with comments during the test. Furthermore are listed the interview answers from the test subjects see section D.2.

6.2 Interview

The results of the interview with subjects show that there is a general good understanding of the applications features and of how it works. There were only a few occasions where problems appeared.

The analysis of the interview with the test subjects is divided into the following sub sections **Visualization**, **Navigation** and **Other**.

6.2.1 Visualization

The visualization of the graph had both positive and negative impacts on the test subjects (TS). The majority of the test subjects expressed that they liked the visualization and that the application was interesting to use, however, it was not flawless.

6.2.1.1 General

First of all the visualization gave the TS a good impression of the application and several expressed their opinion.

> TS1: - *It's cool that you can see the connection visualized instead of lists.*
> TS3: - *It is much more fun to see something visualized than the previous prototypes, which was a bit boring.*
> TS5: - *It looks exciting, but its not something that I would normally use, to complicated for a quick information search.*

There where some minor but noticeable problems with the visualization among other things the labels used and the colouring of the nodes in the graph. TS2 didn't like the colours of the nodes:

> TS2: - *I'm not happy with the colours, it looks like something from the 80's...*

TS3 had similar problems with visualization of the nodes and proposed that there should be nodes with different colours:

> TS3: - *I don't like that the colours are the same for every node.*
> TS3: - *Should be different colours depending on the nodes connections or type.*

TS5 and TS6 both liked the visualization, but it an overview was missing and it could easily turn into a mess if there were to many nodes presented.

TS5: - *It looks better than the prototype, but I rather want the lists, they give a better overview.*

TS6: - *It looks really good, but it can be confusing if nodes are placed on top of each other.*

The labels used in the application were another problem that was commented.

TS1: - *Nice with the visualization, but the text could be a bit larger and more clear. Some of the text can look a little blurry.*

TS3: - *Text size on edges should be bigger.*

6.2.1.2 Double link

A question appeared with several subjects when they discovered the existence of two linked nodes with the same name and had to be explained why. This happens if an object has two types of connections to other objects. TS3 and TS4 commented on this:

TS3: - *It looks weird when the nodes exists more than once. Fx. when a node has two connections to the same neighbour node.*

TS4: - *It looks odd that it shows links to two nodes with the same name and sometimes doesn't show a full circle of nodes.*

6.2.1.3 New and old nodes

Visualizing the appearance of new nodes when selecting new layer was difficult for the subjects to notice. Two of the subjects commented on this.

TS2: - *It is difficult to see that new nodes have appeared, there should be more animation or a text showing it was changed.*

TS3: - *Show which layer that is visualized, it is difficult to see this change.*

The visualization of nodes were to uniform in its presentation and nodes that were not directly linked to the selected node should be visualized differently. TS4 thought it was not clear enough which node that was selected.

TS4: - *Nodes further away from the select node, should be smaller or less visible. There should be a visual difference.*
TS4: - *It is not clear which node that is selected. Tried to show Wikipedia page for a node, but other node was selected.*

One of the test subject found a problem with showing nodes between linked nodes to previous visited node.

TS2: - *Show links between nodes far away from each other, which are not directly connected.*

6.2.1.4 Layers

Visualization of the layer the user had to navigate through, needed to be changed. A few of the subjects wished that they wanted to see the previous and next layer of nodes.

TS2: - *Would be nice if the next layers was visible as laying in a bigger circle in the background.*

TS3: - *It should be possible to see next available layer in the background.*

Furthermore TS4 wished that the visited nodes further away was visualized differently.

TS3: - *Show previous visited layers blurred out.*

TS4: - *Nodes further away from the selected should be smaller or blurred out.*

Visualization issues:
The following lists some of the issues with the visualization.

1. Visualization colouring.
2. Visualization labels.
3. Visualize multiple links to the same node.
4. Show current layer.
5. Notification of new layer.
6. Positioning of nodes according to it neighbours.

6.2.2 Navigation

The navigation of the application arose some confusion and frustration, for one of the test subjects it was even too complicated. It was not intuitive for TS5 to use, he clearly stated that this demanded some more time to learn:

> TS5: - *It felt a bit too complicated to use. I had some problems when I had to go back to the Tom Hanks.*

TS4 expressed that she needed instruction in the application on how to use it.

> TS4: - *Instruction text should be visible all the time or another way of helping with the navigation.*

The big problem with the navigation for the majority of the test subjects was the feature for selecting more layer of nodes using the zoom functionality. Some had trouble understanding the meaning of the next layer and some intentionally used the zoom function with the expectation of an overview of the view.

TS1 was one that had problem with the selection of a another layer of nodes.

> TS1: *The navigation is ok, but it was a bit unclear how to "zoom to next layer"*

TS1 was confused when he wanted to zoom and then got a new layer of nodes and commented on what features he thought that the app was missing.

> TS1: - *Normal zoom function. I tried to zoom for a better view of the nodes, but i got a new layer, it wasn't what I expected.*

TS2 had similar difficulties with the selection of a new layer of nodes and expressed that it had to be improved.

> TS2: - *It is not clear that you should use the zooming function to get more linked nodes. There should be a marker or an indicator that showed zooming was possible.*
> TS2: - *The zooming feature should be the reversed, zooming out for next layer by pinching fingers.*

TS3 expressed his concern about this feature and had another suggestion for improvement of selecting layers.

> TS3: - *A normal zoom function to show an overview...*
> TS3: - *Use a slider to navigate to another layer.*

TS4 tried to use the zoom function as normally intended, but the result was more confusing than helping.

> TS4: - *It is odd that you can't zoom out. Confusing that the zooming function is used in another way. It is not what you would expect in a normal app. I clearly forgot about the instructions.*

The same problem occurred for TS6.

> TS6: - *It is a bit confusing that the normal way of zooming isn't working, have tried several times to zoom out.*
> TS6: - *I would really like the zoom function, I missed the overview.*

Two of the test subjects expressed the need for more instructions. When asked what the app was missing and what they thought about the navigation, they answered:

> TS5: - *There should be more instructions while using the app. Couldn't remember the initial instructions*

> TS6: - *It would ben nice with more help for the navigation. Maybe more labels or icons indicating what to do.*

When asked for what the app was missing, TS1 and TS3 suggested that there should be a more visible loading indicator.

> TS1: - *It needs a loading indicator in the middle. I didn't saw the loading indicator at the top.*

> TS3: - *Show loading indicator in the node.*

Navigation issues:
The following lists some of the issues with the navigation of the application.

1. Selecting next layer.
2. Replacing the zoom functionality.
3. Indicate when more layers are available.
4. Loading indicator.

6.2.3 Other

Not everyone found features in the prototype that they thought was usable in the application, but two of the subjects pointed out that the sorting and grouping of pages used in the lists in the prototypes could be suitable for the application.

TS1: - *It was nice that the movies was sorted chronologically, so another way of sorting the nodes.*

TS6: - *Maybe something that grouped or categorized the nodes for example with different colours.*

6.2.3.1 Special improvements

TS2 suggested a way of searching by combining nodes and finding similarities between them.

TS2: - *Make advanced searching by drag and drop nodes on top of each other to find and visualize similarities between them.*

TS4 suggested that it should be possible to have more control over the visualization.

TS4: - *The option to remove separate nodes by yourself.*

6.2.3.2 Connected vs. disconnected graphs

When selecting new subjects in the web view, the visualization clears the previous to make room for the new graph. This feature arosed mixed feelings among the TS's, however, the majority wished that the nodes persisted.

TS1: - *It is nice that it clears the area when clicking on links in the web view. It would be difficult to see what happens if there are to many things on the screen.*

TS3: - *When a new node is selected from the Wikipedia page, the previous nodes should be kept in the view. Such that it would be easier to see previous nodes and go back.*

TS4: - *Would like to see all nodes, even when new nodes are added and have no links to existing nodes.*

6.2.3.3 Web view

The web view was developed for the user to easily access Wikipedia pages and for the user to navigate through this. The test subjects were happy with its existence, but did express some concern. TS1 was positive about the web view, but he experienced problems.

> TS1: - *It was difficult to figure out how to use, but it is smart feature. It would be nice if it pops up when touching the search field or the page its self.*

TS2 and TS3 were not satisfied with the current implementation

> TS3: - *I'm not fond of the web view. It looks like an easy way to show the information. However, it is nice that you can change to a new node from there.*

TS2 was not happy about the relation between the selected node and the web view and had difficulties to use it. He did have a suggestion to how this should be improved.

> TS2: - *It is not clear where to find the Wikipedia page. The bar should have another colour or a symbol indicating what it is.*
> TS2: - *Show Wikipedia page and visualization side by side.*

Other issues:

1. Complicated to use.
2. Instruction labels.
3. Connected vs. Disconnected graphs.

Improvements:

1. Change the sorting of linked nodes.
2. Web view: positioning, relation, visual.

3. Advanced search.
4. Hide/remove nodes.

6.3 Performance test

The following table shows the results of response times when communicating with server.

Web service response time		
Wikipedia page	Linked nodes	Avg. response time s.
DTU	22	0.489282
Kevin Bacon	61	0.3960184
Tom Hanks	89	0.4528436
Copenhagen	1440	0.5034282
Denmark	4174	0.8137042
London	15704	1.31386
Germany	34166	2.0476316
United States	211325	6,7832036

Table 6.1: Web service response times.

Table 6.1 presents the average response times for a 8 selected Wikipedia page. This shows that the more links there are between the pages the higher the response times gets. This is noticeable when looking at Wikipedia pages such as London, Germany or United States. Each of them has many thousands of links and the web service is slower to return the result. The majority of the Wikipedia pages stored in the database doesn't have more than 100 links and the response time for when the application is to visualize new nodes, is below 0.5 seconds.

Chapter 7

Conclusion

The use of linked data is evolving. This means that the need for applications to work with linked data is bigger and this is what we have tried to accomplish. The main goal of this project was to discover the possibilities in navigating and visualizing of the linked data. This was accomplished by developing an iPad application based on the Wilky May application and furthermore a back end that could process linked data.

The created application can present the relations of Wikipedia pages based on the internal Wikipedia links from DBpedia. For better presentation of the Wikipedia pages a PageRank sorting was used in an interactive graph visualization. For evaluating the application an usability and performance test was conducted. The findings of the evaluation and the usability test show that the application is usable and interests users with its use of visualization instead of lists and buttons as used in the prototypes. The navigation of the linked data caused some issues for the users.

The finding shows that the visualization in general was good, but that it could be improved with other colours, larger and more clear text and more detailed visualization with more links. Furthermore it shows that available layers should be visible in some way to be explored. The navigation caused a rather large problem for the users. The feature of visualizing and navigating to more linked nodes than was visualized was too difficult. The findings shows that the visu-

alization was insufficient in regarding to show available layers. The navigation using pinch zoom was too difficult to use and had to be improved. The performance test shows that the backend of the application could be a weak link, but in general it could return the results with an accepted response time.

Despite of the issues described in the findings, a working application was developed to present the possibilities of visualizing and navigating of linked data in a proper manner.

Chapter 8

Discussion

The project has introduced me to many new things among other things working with linked data and visualizations on tablet devices. This has let me experience a lot of new exciting areas of mobile development.

8.1 Implementation

The application developed in this project is an iPad application which can visualize interactive graphs of linked data. The design and implementation of the application are described in chapter 4. The backend developed to handle the requests for linked data by the iPad application is working well and return the desired results, but unfortunately suffers from high response times. The final product is a working application with backend and fulfils the most of the requirements.

8.2 Evaluation

The evaluation process of the project consisted of a usability test of the developed application and three alternative prototypes. The test should give an indication of how the user evaluated the potential and possibilities of application. The prototypes were developed to give the user ideas for other ways to navigate and visualize linked data. These prototype could have been developed in a lot of different ways and thereby influencing the test subjects differently. This also apply for the tasks and the interview questions that were created for the evaluation. In an usability study using an unstructured interview and open questions it is difficult to decide the outcome, but it can also be for the better when the test subjects give unexpected but usable feedback.
Optimal the usability test should have been conducted several times throughout the development process to give the best final result.

8.3 Further improvements of the application

There are almost always things you can point out after a finished project phase and this is not an exception. The following describes the findings in the usability test and our own thoughts for further development for the application.

8.3.1 Use of linked data

This application was limited to only use the data set from DBpedia containing data of Wikipedia to show our idea with visualization and navigation in linked data. The DBpedia data set consists of a lot of links to other data set, but in the project the use of DBpedia was limited to only show relations between Wikipedia pages. This was implemented with success, but the application would be more interesting if it was created to use the whole DBpedia dataset or use of other data sets. This could for example be geographical data which could be visualized on the map or data sources such as movies, music, books etc. With use of more data sets it would enrich the experience of the application and make it more useful.

8.3.2 Backend

The performance test showed that application works with an accepted response time in average under 1 second, however it would benefit the experience of the application if the backend was improved. This is especially when the application is querying linked data for a Wikipedia page with many thousands of linked pages. These are normally the popular ones that are often visited such as countries and big cities and it would damage the experience of the application if the response times were too high.

The result from the usability test presented in the chapter 6, revealed a lot of unknown issues and produced a lot of suggestions for changes in the application. These results are good guidelines for future work on this project and further development of the application.

8.3.3 Navigation

One of the main focuses of the application and the project was to find a way to navigate in huge amount of links in linked data. In this case internal Wikiepdia links from the DBpedia data set. The zoom navigation used to browse for more linked data by selecting layers, triggered a lot of problems for the test subjects in the testing. This feature was implemented such that it replaced the "normal" zoom function for scaling the view. This was confusing for the test subjects that even when they were told how the application worked, some tried to use zoom as they were used to. Other ways of selecting layers were discussed. The use of a slider was suggested.

8.3.4 Overview

The problem with selecting new layer in the navigation was among other things due to the test subjects expectation of getting an overview by zooming out. Several of the test subjects expressed that they did expect to get an overview when zooming out and it was confusing when the graph change appearance.

8.3.5 Visualization

The visualization of the graph in the application was working well and several of the subjects expressed that they liked it and was making the application

interesting. However, the visualization did have some features that needed to be added or improved.

8.3.5.1 Missing link

The visualization was well received as an exciting and more interesting use in tablet application. However it did had some flaws. When a few nodes have been visited, the linked nodes to the current selected node can have a direct relation to a previous visited node, but this isn't always visualized.

8.3.5.2 Visualization settings

It was needed to implement visualization settings for selecting preferences of the user. Some of the test subjects were not happy with the colour and the labels text size in the visualization. Furthermore it was discussed whether or not old nodes with no direction relation to new nodes should be preserved when selected from the web view.

8.3.5.3 Node layers

The navigation of the linked data was a key part of this project. Unfortunately this was made difficult by not visualizing the possibilities of the application entirely. The usability test shows that the need for better visualization of the new and old layers was highly wished.

8.3.5.4 Position of nodes

A general opinion is to keep the visualization simple and easily readable for the users. This means that the visualization should not present all the data as a big pile. One of test subject was worried about if it would be to crowded and thought that the positions of nodes according to the other should be improved.

8.3.6 Web view

The web view is a key part of the application and lets the user explore the nodes further and search for new topics. The web view was well received even though some thought it was difficult to operate and it wasn't particular unique. Others thought that it was easier to use. For a better use it was suggested that it should be presented simultaneously with visualization. For instance to show the visualization and the web view side by side.

Appendix A

SQL Query

```
select wl.Predicate, wpo.PageName,wps.PageName as Node,
    wps.PageId as NodeId, wr.Links, wr.Score
      from [DBWikiData].[dbo].WikiLinks wl
      inner join [DBWikiData].[dbo].WikiPageId wps
      on wl.SubjectId = wps.PageId
      inner join [DBWikiData].[dbo].WikiPageId wpo
      on wl.ObjectId = wpo.PageId
      inner join [DBWikiData].[dbo].WikiRanks wr
      on wl.SubjectId = wr.PageId
      where wl.ObjectId = @searchid
      union
      select wl.Predicate, wps.PageName,wpo.PageName as
          Node, wpo.PageId as NodeId,wr.Links, wr.Score
      from [DBWikiData].[dbo].WikiLinks wl
      inner join [DBWikiData].[dbo].WikiPageId wps
      on wl.SubjectId = wps.PageId
      inner join [DBWikiData].[dbo].WikiPageId wpo
      on wl.ObjectId = wpo.PageId
      inner join [DBWikiData].[dbo].WikiRanks wr
      on wl.ObjectId = wr.PageId
      where wl.SubjectId = @searchid

      order by score desc
```

Appendix B

Web service output

This shows a json output when quering the webservice for "Tom Hanks".

```
{
  GetDataResult: [
    {
      Links: 21551,
      Object: "Forrest_Gump",
      ObjectId: 41528,
      Predicate: "narrator",
      Score: 0.0000030143,
      Subject: "wiki:Tom_Hanks"
    },
    {
      Links: 21551,
      Object: "Forrest_Gump",
      ObjectId: 41528,
      Predicate: "starring",
      Score: 0.0000030143,
      Subject: "wiki:Tom_Hanks"
    },
    {
      Links: 244653,
```

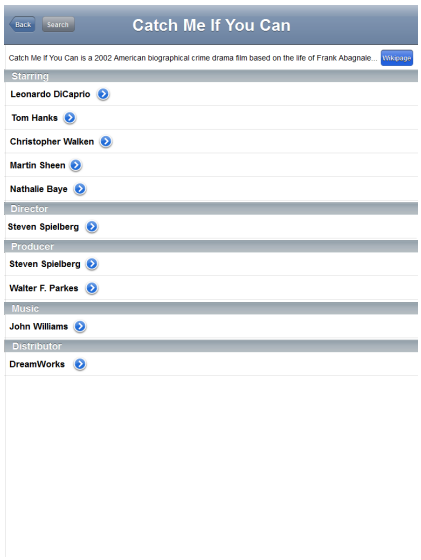
```
      Object: "California_State_University,
         _Sacramento",
      ObjectId: 637852,
      Predicate: "education",
      Score: 0.00000295318,
      Subject: "wiki:Tom_Hanks"
    },
    {
      Links: 27222,
      Object: "Toy_Story",
      ObjectId: 53085,
      Predicate: "starring",
      Score: 0.00000276214,
      Subject: "wiki:Tom_Hanks"
    },
    {
      Links: 52207,
      Object: "Concord,_California",
      ObjectId: 107395,
      Predicate: "birthPlace",
      Score: 0.00000254867,
      Subject: "wiki:Tom_Hanks"
    },
    {
      Links: 380204,
      Object: "Toy_Story_3",
      ObjectId: 1213838,
      Predicate: "starring",
      Score: 0.00000225291,
      Subject: "wiki:Tom_Hanks"
    },
    {
      Links: 84054,
      Object: "Apollo_13_(film)",
      ObjectId: 142417,
      Predicate: "starring",
      Score: 0.00000208045,
      Subject: "wiki:Tom_Hanks"
    },
    {
      Links: 117004,
      Object: "Sleepless_in_Seattle",
      ObjectId: 226198,
      Predicate: "starring",
```

```
64          Score: 0.00000156044,
65          Subject: "wiki:Tom_Hanks"
66        }
67    ]
68  }
```

Appendix C

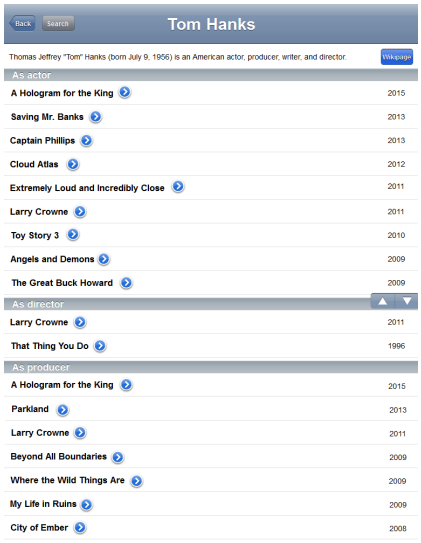
Prototypes

A selection of four images from the three prototype are presented in the following.

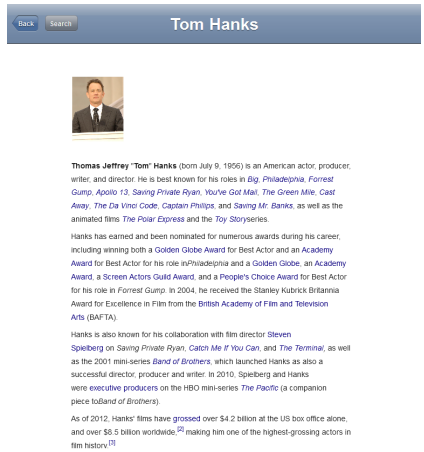


(a)

(b)

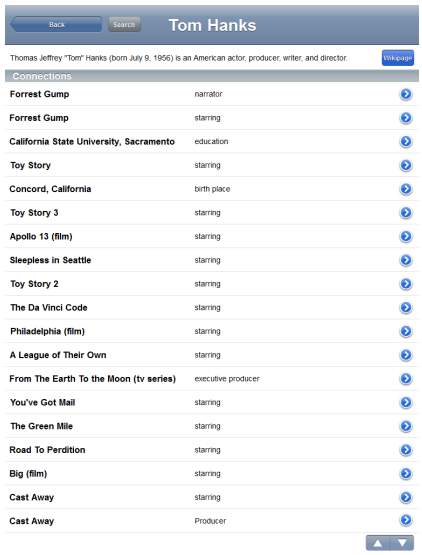


(c)



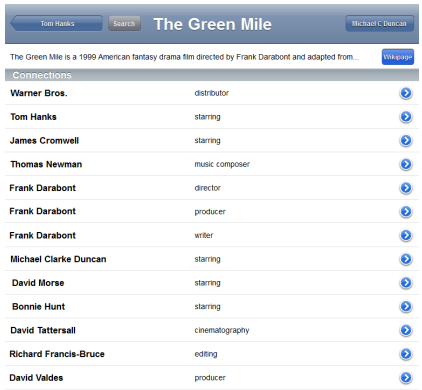
(d)

Figure C.1: Four images of the first prototype

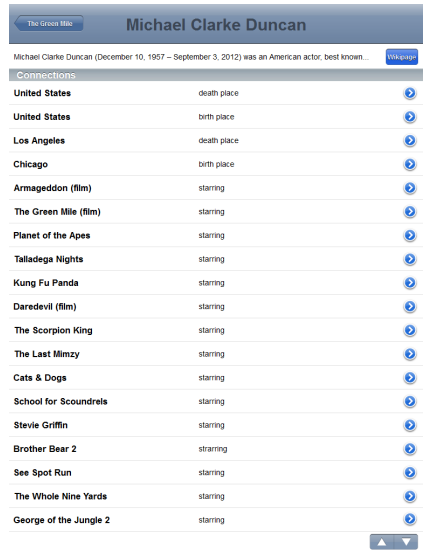


(a)

(b)



(c)



(d)

Figure C.2: Four images of the second prototype

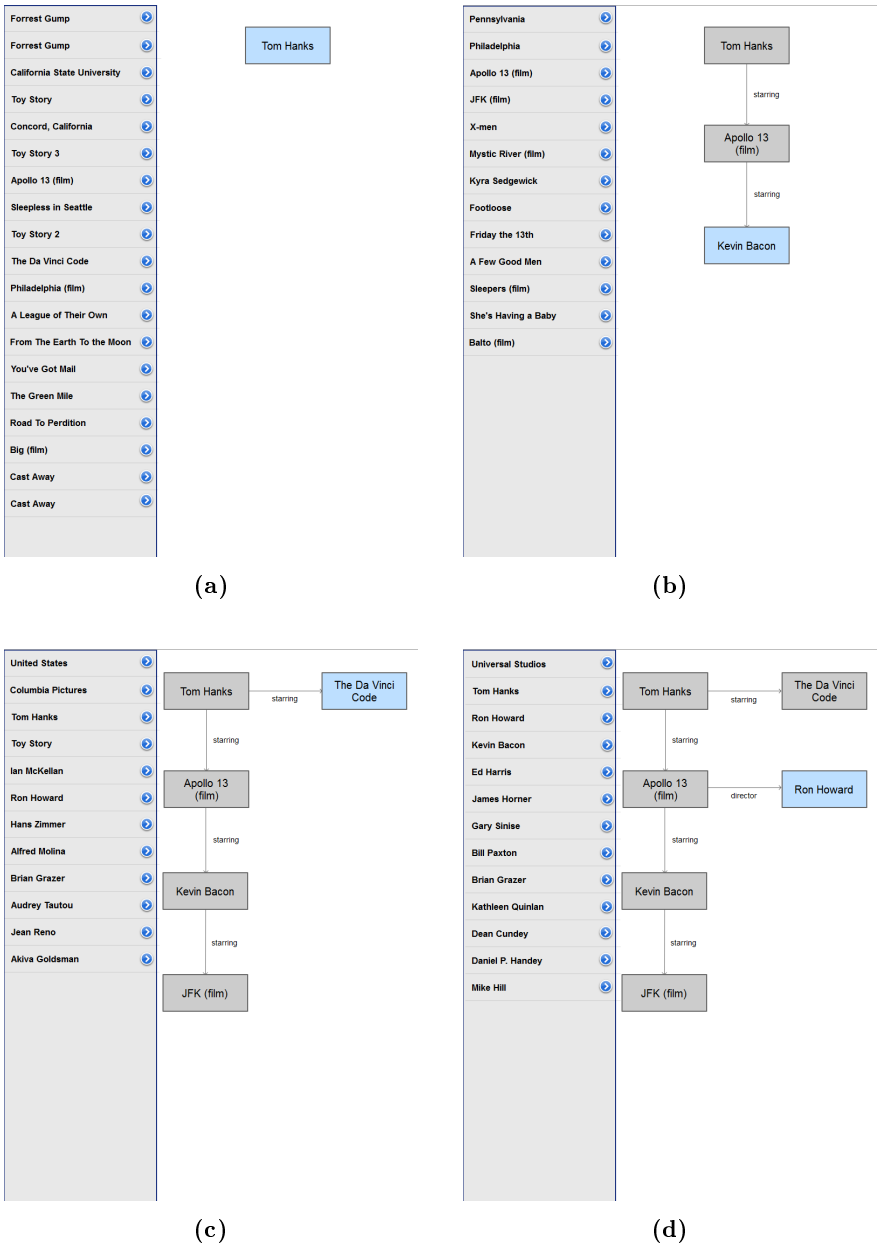


Figure C.3: Four images of the Third prototype

Appendix D

Interview

The interviews consisted of a test with three alternative prototypes and the final application. For each prototype the user was given a few tasks to complete. These tasks lets the user experience the navigation and different visualizations in the prototypes and in the final application. The following tables presents the tasks and the observations during the test.

D.1 Tasks

D.1.1 Prototype 1

Prototype Tasks		
#	Description	Observations
1	Search for "Tom"	No problem
2	Select "Tom Hanks"	No problem
3	Find more movies "Tom Hanks" have appeared in	Some confusion about scrolling. The prototype didn't support that feature.
4	Find the Wikipedia page for "Tom Hanks"	Some had trouble finding the wiki button
5	Find and select "The Simpsons Movie"	No problem
6	Go back and select "Catch Me If You Can"	No problem
7	Find the Wikipedia Page for "Tom Cruise"	It was a clear problem to find the search button.

Comments for the prototype:
- *I know its a prototype, but I need options to go back and fourth.*
- *It is not clear what the Wikipage button correspond to.*

D.1.2 Prototype 2

Prototype Tasks		
#	Description	Observations
1	Search for "Tom"	No problem
2	Select "Tom Hanks"	No problem
3	Find more data linked to "Tom Hanks"	It was not all who thought about scrolling. Comment: "The first prototype was better", "There should be a label for how the list are sorted"
4	Find and select "The Green Mile"	No problem
5	Find the Wikipedia page for "The Green Mile" actor "Michael Clarke Duncan"	No problem. Comment: "it is nice that the back button has changed to Tom Hanks". "Looks weird when the United States are listed twice"
6	Find the Wikipedia page for "The Green Mile"	Some had trouble finding back and tried to use the search function.

Comments for the prototype:
- It would be nice with an option to select how the lists is sorted for example alphabetically, type or chronological. - The first prototype had a better list with the categorization.

D.1.3 Prototype 3

Prototype Tasks		
#	Description	Observations
1	Search for "Tom"	No problem.
2	Select Tom Hanks on the list	No problem.
3	Select Apollo 13 on the list	No problem. Comment: "Nice to see it visualized, but the list needs the info link type from the previous prototypes.
4	Select Kevin Bacon on the list	No Problem.
5	Select JFK on the list	No Problem.
6	Select Tom Hanks without using the list	Some hesitated and didn't tried to click on the visualization.
7	Select The Da Vinci Code	No problem.
8	Select Apollo 13 without using the list	No problem.
9	Select "Ron Howard"	No problem. Comment: "Really nice to see the connections like that, and see the previous pages.

Comments for the prototype:
- Its a lot easier to use and to get an overview than the prototype 1 and 2.
- The Wikipedia information is missing compared to the other prototype. Could be a small button or info icon on the nodes.
- Include links between nodes with items in the list.
- Include categorization or sorting of the list.

D.1.4 Application

Application Tasks		
#	Description	Observations
1	Select Forrest Gump	No problem.
2	Select Gary Sinise	No problem.
3	Select CSI	No problem.
4	Select Tom Hanks again	One test subject had trouble selecting Tom Hanks again, didn't try to move the view by panning, but selecting all previous nodes.
5	Explorer next layer of nodes for Tom Hanks	Not all could remember the initial guide on how to do that and some tried double tapping or long press on the node. Almost everyone who used the pinch zoom tried zooming out before in. The reverse of the implementation.
6	Find more information about Tom Hanks by swiping the bottom bar upwards	The most didn't had any problem. One was frustrated that the view didn't pop up when touching inside the view and one was confused when a keyboard covered the view.
7	Search for Bruce Willis	No problem.
8	Select Bruce Willis Die Hard character John McClane	No problem. Comment: "I might used this instead of the visualization."
9	Show visualization by swiping the information view downwards	No problem. Comment: "Nice that it showed the page visited, but where are the previous nodes?"

D.2 Interviews

Test Subject 1	
Question	Answer
What do you think about the final app compared to the prototypes?	- It's cool that you can see the connection visualized instead of lists. - Cool way to use Wikipedia, but maybe a bit too much if you just want to find specific information about a single subject.
What do you think of the visualization of the results?	- Nice with the visualization, but the text could be a bit larger and more clear. Some of the text can look a little blurry.
What do you think about the navigation?	- The navigation is ok, but it was a bit unclear how to "zoom to next layer"
What do you think about the Wikipedia pop up?	- It was difficult to figure out how to use, but it is smart feature. It would be nice if it pops up when touching the search field or the page its self. The keyboard covers the view if it doesn't pop up.
What features from the prototypes are missing in the final app?	- It was nice that the movies was sorted chronologically, so another way of sorting the nodes.
What other features do you miss in the app?	- It needs a loading indicator in the middle. I didn't saw the loading indicator at the top. - Normal zoom function. I tried to zoom for a better view of the nodes, but i got a new layer, it wasn't what I expected.
Other comments?	- When moving nodes around, it would be nice if nodes connected follows. - It is nice that it clears the area when clicking on links in the web view. It would be difficult to see what happens if there are to many things on the screen.

Test Subject 2	
Question	Answer
What do you think about the final application compared to the prototypes?	- More fun than the "normal" way with lists and buttons.
What do you think of the visualization of the results?	- I'm not happy with the colours, it looks like something from the 80s. It could maybe be such that the user could decide and change. - Would be nice if the next layers was visible as laying in a bigger circle in the background. - The most popular should be closest and then less popular in wider and wider circles. - It is difficult to see that new nodes have appeared, there should be more animation or a text showing it was changed.
What do you think about the navigation?	- It is a bit complicated. - It is not clear that you should use the zooming function to get more linked nodes. There should be a marker or an indicator that showed zooming was possible. - The zooming feature should be the reversed, zooming out for next layer by pinching fingers.
What do you think about the Wikipedia pop up?	- It is not clear where to find the Wikipedia page. The bar should have another colour or a symbol indicating what it is.
What features from the prototypes are missing in the final app?	- Nothing.

Test Subject 2	
Question	Answer
What other features do you miss in the app?	- Could be nice if there be an image on the neighbour nodes. - Show links between nodes far away from each other, which are not directly connected. - Make advanced searching by drag and drop nodes on top of each other to find and visualize similarities between them. - Show Wikipedia page and visualization side by side. - Change the orientation of the view. I personally often use my iPad in horizontal view.
Other comments?	- When the Wikipedia page is in the foreground it may be easy to forget the visualization behind.

Test Subject 3	
Question	Answer
What do you think about the final app compared to the prototypes?	- It is much more fun to see something visualized than the previous prototypes, which was a bit boring.
What do you think of the visualization of the results?	- I don't like that the colours are the same for every node. - Should be different colours depending on the nodes connections or type. - It looks weird when the nodes exists more than once. Fx. when a node has two connections to the same neighbour node. - Show previous visited layers blurred out. - It should be possible to see next available layer in the background.
What do you think about the navigation?	- It is not clear that you have to use the zoom to get more nodes. - When a new node is selected from the Wikipedia page, the previous nodes should be kept in the view. Such that it would be easier to see previous nodes and go back.
What do you think about the Wikipedia pop up?	- I'm not fond of the web view. It looks like an easy way to show the information. However, it is nice that you can change to a new node from there.
What features from the prototypes are missing in the final app?	None
What other features do you miss in the app?	- A normal zoom function to show an overview or in the same time show an small overview in a window. - Use a slider to navigate to another layer. - Show which layer that is visualized, it is difficult to see this change. - Show loading indicator in the node.
Other comments?	- Text size on edges should be bigger.

Test Subject 4	
Question	Answer
What do you think about the final app compared to the prototypes?	- Graphs are new and unexpected in such kind of apps. - More interesting than traditional apps fx. like the first prototypes.
What do you think of the visualization of the results?	- Would like to see all nodes, even when new nodes are added and have no links to existing nodes. - It is not clear which node that is selected. Tried to show Wikipedia page for a node, but other node was selected. - Nodes further away from the select node, should be smaller or less visible. There should be a visual difference. - It looks odd that it shows links to two nodes with the same name and sometimes doesn't show a full circle of nodes.
What do you think about the navigation?	- It is odd that you can't zoom out. Confusing that the zooming function is used in another way. It is not what you would expect in normal app. I clearly forgot about the instructions. - It is nice that you can rearrange the nodes by dragging them around, but sometimes it detects that I taps instead and that is a bit frustrating.
What do you think about the Wikipedia pop up?	- It's ok. I think it may be easier just to click on the node to get more information.
What features from the prototypes are missing in the final app?	- No, I don't thinks so.
What other features do you miss in the app?	- The option to remove separate nodes by yourself. - Would like the normal zoom function. - Instruction text should be visible all the time or another way of helping with the navigation. - Info button on nodes for show more information. fx. open the Wikipedia page for the selected node.
Other comments?	- Nodes further away from the selected should be smaller or blurred out.

Test Subject 5	
Question	Answer
What do you think about the final app compared to the prototypes?	- It looks exciting, but its not something that I would normally use, to complicated for a quick information search.
What do you think of the visualization of the results?	- It looks better than the prototype, but I rather want the lists, they give a better overview.
What do you think about the navigation?	- It felt a bit too complicated to use. I had some problems when I had to go back to the Tom Hanks. Thought I had to go back by selecting every other node. - It was not clear that you have to zoom to get more node
What do you think about the Wikipedia pop up?	- It looks good when it appears like that, but I would probably be using that instead. - The other nodes should not be removed when I click on a link.
What features from the prototypes are missing in the final app?	A list overview would be nice.
What other features do you miss in the app?	- There should be more instructions while using the app. Couldn't remember the initial instructions.
Other comments?	- It looks good, but I don't think that it would be something for me.

Test Subject 6	
Question	Answer
What do you think about the final app compared to the prototypes?	- It is more fun to use than the previous prototypes. - This inspires you to explore more in the Wikipedia articles.
What do you think of the visualization of the results?	- It looks really good, but it can be confusing if nodes are placed on top of each other. - I like the design, clean and simple.
What do you think about the navigation?	- It is a bit confusing that the normal way of zooming isn't working, have tried several times to zoom out. - It would be nice with more help for the navigation. Maybe more labels or icons indicating what to do.
What do you think about the Wikipedia pop up?	- I like this function, it is good that it is so easy and quick to get in front and remove.
What features from the prototypes are missing in the final app?	- None of them.
What other features do you miss in the app?	- I would really like the zoom function, I missed the overview. - Maybe something that grouped or categorized the nodes for example with different colours.
Other comments?	- I'm not sure I understand the popularity of the nodes and the difference between layers.

Bibliography

[AE04] Erik Andersson and Per-Anders Ekström. Investigating google's pagerank algorithm. *Report in Scientifc Computing, advanced course-Spring*, 2004.

[App] Apple. ios development. `https://developer.apple.com/devcenter/ios/index.action`. Accessed: 01-08-2014.

[BE05] Ulrik Brandes and Thomas Erlebach, editors. *Network Analysis: Methodological Foundations (Lecture Notes in Computer Science / Theoretical Computer Science and General Issues)*. Springer, 2005 edition, 2 2005.

[BHBL09] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data - the story so far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.

[Bos] Mike Bostock. D3 javascript. `http://d3js.org/`.

[CMA] Diego Valerio Camarda, Silvia Mazzini, and Alessandro Antonuccio. Lodlive project. `http://en.lodlive.it`. Accessed: 06-07-2014.

[CMA12] Diego Valerio Camarda, Silvia Mazzini, and Alessandro Antonuccio. Lodlive, exploring the web of data. In *Proceedings of the 8th International Conference on Semantic Systems*, I-SEMANTICS '12, pages 197–200, New York, NY, USA, 2012. ACM.

[DJ10] Easley David and Kleinberg Jon. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, New York, NY, USA, 2010.

[DK09] Jiri Dokulil and Jana Katreniakova. Rdf visualization - thinking big. *PROCEEDINGS OF THE 20TH INTERNATIONAL WORKSHOP ON DATABASE AND EXPERT SYSTEMS APPLICATION*, pages 459–463, 2009.

[eFa] eFactory. Pagerank. `http://pr.efactory.de/e-pagerank-algorithm.shtml`. Accessed: 01-08-2014.

[Fac] Facebook. Facebook graph api. `https://developers.facebook.com/docs/graph-api/quickstart/v2.0`. Accessed: 06-07-2014.

[Fli] Flinto. Flinto. `https://www.flinto.com`. Accessed: 31-07-2014.

[Gooa] Google. Android development. `http://developer.android.com/index.html`. Accessed: 01-08-2014.

[Goob] Google. Google chart. `https://developers.google.com/chart/`. Accessed: 10-07-2014.

[HB11] Tom. Heath and Christian. Bizer. *Linked data : evolving the web into a global data space*. Morgan & Claypool, 2011.

[Inf] Infoviz. Infovis javascript. `http://philogb.github.io/jit/demos.html`. Accessed: 02-07-2014.

[Jac] Alexis Jacomy. Sigma javascript. `http://sigmajs.org/`. Accessed: 02-07-2014.

[LIJ+14] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, 2014.

[lin] Linked jazz. `http://linkedjazz.org/`. Accessed: 02-07-2014.

[Mica] Microsoft. Windows server 2012. `http://www.microsoft.com/en-us/server-cloud/products/windows-server-2012-r2/`. Accessed: 01-08-2014.

[Micb] Microsoft. Windows server 2012. `https://www.microsoft.com/en-us/server-cloud/products/sql-server/default.aspx`. Accessed: 01-08-2014.

[Mil] MilkyWay. Milkyway. `http://milkywaylabs.com/`. Accessed: 02-07-2014.

[PBMW99] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.

[Pre] Prefuse. prefuse. `http://prefuse.org/`.

[Pro] DBPedia Project. Dbpedia. `http://dbpedia.org/About`. Accessed: 01-07-2014.

[RSI96] Jim Rudd, Ken Stern, and Scott Isensee. Low vs. high-fidelity prototyping debate. *interactions*, 3(1):76–85, January 1996.

[Ruh00] Britta Ruhnau. Eigenvector-centrality — a node-centrality? *Social Networks*, 22(4):357 – 365, 2000.

[SRP07] Helen Sharp, Yvonne Rogers, and Jenny Preece. *Interaction Design: Beyond Human-Computer Interaction*. Wiley, 2 edition, 3 2007.

[Tou] TouchGraph. Touchgraph. `http://www.touchgraph.com/navigator`. Accessed: 10-07-2014.

[W3Ca] W3C. Rdf 1.1 concepts and abstract syntax. http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/. Accessed: 02-07-2014.

[w3cb] w3c. w3 linked data. `http://www.w3.org/DesignIssues/LinkedData.html`. Accessed: 06-07-2014.