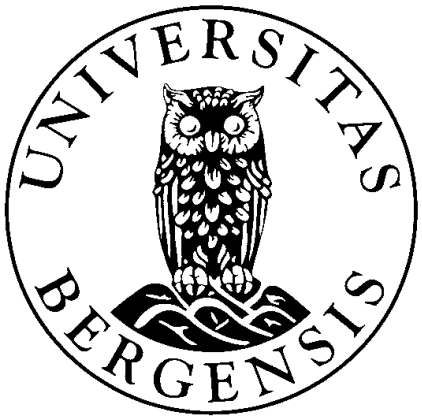

Improving usefulness and ease of use for a prototype tool for journalists

By: Sindre Moldeklev
Thesis advisor: Andreas L. Opdahl
DEPARTMENT OF INFORMATION SCIENCE AND MEDIA STUDIES AT THE UNIVERSITY OF BERGEN, NORWAY

May 24, 2018

Abstract

This thesis presents the work done on making a prototype tool for journalists more useful and easier to use through continuous improvements and evaluations. The work builds on the thesis News Hunter: a semantic news aggregator written by Ole Andreas Christensen and Kjetil Jacobsen Villanger (2017). This thesis focuses on the further development and addition of new functionality through iterative processes during development. As well as new functionality, the thesis focuses on making the project more sustainable for future use, with better documentation and refactoring of code. News Hunter is a collaborative project between the University of Bergen and Wolftech Broadcast Solutions. The goal of this thesis is to add new functionality to News Hunter and better document the project. This has been achieved by continuing working on Christensens and Villangers source code, and by adding new functionality in iterative agile steps throughout the development. The new functionality has then been evaluated by journalists and graphic reporters from TV 2 Norway, as well as journalism students from the University of Bergen.

Acknowledgments

I would like to thank my thesis advisor, Professor Andreas Lothe Opdahl. His professionalism and valuable input has been greatly appreciated.

I would also like to thank Wolftech for their collaboration. A lot good ideas has surfaced in meetings with them throughout the thesis.

Lastly I want to express my gratitude to my family for being supportive and encouraging through all five years of studying. It would be a lot harder to accomplish this degree without their ever loving support and motivational words along the way. Thank you.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Introduction

As more and more people are online, and with a vast majority of people in the developed world owning a smart phone, the easy access of news, and the ease of reporting on news have created a whole new daily life of media. News agencies and news desks around the world have access to a tremendous load of new information hour by hour, minute by minute, but it can be hard to distinguish between what is important and what is noise. With such amounts of data being generated everyday, there is a need for better solutions to navigate in the stream of information. Journalists that does not have technological tools to help them may fall behind and not keep up with how fast information changes. Tools that aid the journalist does already exists, such as *Hermes: a semantic web-based news decision support system* (2008), but continuing to improve the tools and explore what can be done is nevertheless still important. News Hunter is also one of such tools, utilizing the semantic web to support the journalists in their work.

This project carries on the work done by Kjetil Villanger Jacobsen and Ole Andreas Christensen in their master thesis *News Hunter: a semantic news aggregator* (2017). Their work has implemented clustering, event detection and entity- and keyword extraction from different news sources. Letting the journalists at work retrieve up to date, relevant information in the sea of both valuable and non-valuable data.

The goal and motivation of this enhanced version of News Hunter is to improve on the overall experience of the application, as well as introduce new functionality that is missing from the previous version. The overall experience should be visible, from both an end user perspective, as well as a developer perspective. The thesis is a cooperation between the author, the University of Bergen and Wolftech Broadcast Solutions AS[1] to help Wolftech explore what is possible with semantic technologies and to accommodate any new features they require. The thesis does not include any user experience aspects, and during development UX regards has not been considered more than absolutely necessary.

[1] `http://wolftech.no/news.php`

This improvement of News Hunter will introduce new features, such as automatic analysis of stories written, overview over politicians, overview over last news written by other news outlets and visualization of who knows who. This enhanced version will aim at becoming more useful and with better ease of use than its predecessor, both in terms of day to day use, but also for the developers working on future requests from users and Wolftech. The application will continue to use semantic technologies at its core to make reasoning about the data easy, while still maintaining the flexibility to add new incoming data sources in the future. The boundaries of the research will be limited to developing a minimum viable product each iteration, while always focusing on making News Hunter more useful and easier to use.

1.2 Research Questions

Research Question 1 - How to improve usefulness and ease of use in tools for journalists?

The research question is motivated by how software can be made more useful and easier to use for journalists, by enhancing an existing prototype tool for journalists. The tools in question is the second version of News Hunter, a semantic web application developed by Christensen and Villanger (2017).

Research Question 2 - How to organize projects so that ease of use and usability can be maintained?

This research question acts as a subquestion of the main research question in RQ 1. The motivation for this research question is to think about how projects of a certain length and size can be maintained when there are different roles present. For this project, the collaboration goes between the university of Bergen which is in the scientific and social science domain, Wolftech, which is in the business domain and students. Keeping the project organized so that new actors can contribute in the future is therefore very important.

2 Theory and Technologies

This section gives a brief overview of the different technologies that are used to create a more useful and usable version of News Hunter, as well as technologies and implementations that has been developed during the previous iteration of News Hunter done by Christensen and Villanger.

2.1 Programming Languages

This project uses an array of different programming languages to accomplish its task. There are two separate backend modules at work, one written in Python and one written in C#. The frontend is based on AngularJS, the first version of Angular.

2.1.1 Python

There are three main reasons for using Python in this project. One is ease of use in regards to downloading and parsing news sources from the web, the second is the vast majority of different text analysis tools used for entity extraction, sentiment analysis, keyword extraction, classification and connecting to other API's, and the third is as an easy way to run a backend server. The first use of Python is based on scripts that downloads articles from all the major news outlets in Norway, Great Britain and USA. It parses the articles and stores them as JSON. Another script can then push the JSON files to the C# backend which in turn adds them to a background processing job ran by Hangfire.io[2]. The background processing processes each JSON and send them along the pipeline for semantic annotation and processing. A third script sends the JSON files to the Elasticsearch database which is used in conjunction with text analysis.

The second use of Python is as a text analysis tool. The tools chosen do their job well and makes development easy.

[2] `https://www.hangfire.io/`

The third use of Python is to run a Flask App[3] which acts as a REST endpoint for connecting C# code with the analysis tools of Python and returning the results after analysis.

2.1.2 C#

C# is used as the main language for the semantic annotation and lifting pipeline. This is where the data is semantically annotated using the BrightstarDB[4] framework. C# was chosen because it is what Wolftech uses in their daily development and therefore the project originated from C#.

2.1.3 AngularJS

AngularJS[5] is used as the Javascript framework for the frontend of the application. This version of Angular is about to be deprecated, but because of lack of time, I chose not to rewrite the application to a newer version of Angular.

2.2 Databases

The project uses two different databases. A semantic graph which stores the various triples generated via the C# code, and a text database used for text analysis.

2.2.1 BrightstarDB

BrightstarDB has been used as the graph store of choice since the beginning of this project. I did not want to mess with the database, so I too have used BrighstarDB when enhancing and extending the usefulness of News Hunter. BrightstarDB works very well with C# and the .NET platform, so there is no need to change to something else. BrightstarDB is a an RDF triple store. The positive sides of such database is that it does not require

[3] http://flask.pocoo.org/
[4] http://brightstardb.com/
[5] https://angularjs.org/

any database schema definition, and therefore it supports data of different shapes. If specifications changes during development or after the system has been set to production, it is rather easy for developers to add new data to the database, without the need to remodel the database schema. BrighstarDB is also licensed with the MIT license, which means that it can be used in both commercial and non-commercial applications. This is important if Wolftech is to use Newshunter in the future, for their clients meaning it will be commercialized.

2.2.2 Elasticsearch

Elasticsearch[6] is a RESTful and distributed search and analytics engine, mainly used for digesting and searching through text. It is very versatile and can be used in many different scenarios. For this project it is used to retrieve all news articles stored in the Elasticsearch instance, which mentions a person either in the title or in the body of the article. The data is stored in Elasticsearch, after it is downloaded from RSS feeds, using Python.

2.2.3 DBpedia Spotlight

DBpedia Spotlight[7] is a tool used to automatically annotate entities in a text document. It provides an API endpoint which lets you pass it some text data and retrieve a JSON object containing the entities and DBpedia URL's for said entity. It has four main features; spotting, candidate selection, disambiguation and filtering. This project uses spotting to retrieve entities when a user writes an article in the editor. DBpedia Spotlight allows the user to configure the annotations specified for their needs, through the use of the DBpedia Ontology (Mendes et al., 2011).

[6] `https://www.elastic.co/`
[7] `http://www.dbpedia-spotlight.org/`

2.3 Semantic Technologies

2.3.1 RDF

This project is using Resource Description Framework (RDF)[8] as a means to semantically annotate the news articles into a standard format that can later be queried with the SPARQL Query Language. RDF is a standard model for data interchange on the Web. RDF has features that facilitate data merging even if the underlying schemas differ, and it specifically supports the evolution of schemas over time without requiring all the data consumers to be changed (Group, 2014).

2.3.2 OWL - Web Ontology Language

OWL[9] is a semantic web language used to describe a domain of knowledge. It can describe things, people, relationships and so on, in a machine processable manner that can later be verified. By using OWL, the system can derive either implicit or explicit knowledge (W3C, 2012)

2.3.3 SPARQL

SPARQL[10] is a query language for RDF. It has since the first working draft in 2004 been widely adopted, and in January 2008 it became a W3C Recommendation (Pérez et al., 2009). SPARQL provides tools for accessing information in an RDF graph through the use of a structured query language. This project uses SPARQL for retrieval of the semantic data stored in BrightstarDB.

[8] https://www.w3.org/RDF/
[9] https://www.w3.org/OWL/
[10] https://www.w3.org/TR/rdf-sparql-query/

2.4 Systems for News Desks

A news desk is a busy place, with tight deadlines and an ever increasing demand to publish the story first online. Journalists wants to write informative and good articles, backed by facts, and News Hunters aim is to help journalists achieve this. For a lot of news desks around the world, a good portion of how ratings are measured is trough clicks online. The more clicks a story gets, the better. Making sure that journalists have the best tools available for their tasks are important. Wolftech, which was founded in 2011 in cooperation with the news and TV broadcaster TV 2 Norway, creates software aimed at aiding the journalists and editors in their daily workflow.

2.4.1 Wolftech and Wolftech News

Wolftech has since the start in 2011 worked on developing tools for the modern journalist. Wolftech News is one of those products. News aims to stimulate creativity and collaboration. It provides journalists and editorial executives good solutions for working efficiently and to create, manage and publish media to a variety of publishing platforms (Wolftech, 2016). The features of Wolftech News are visually summed up in figure 1.

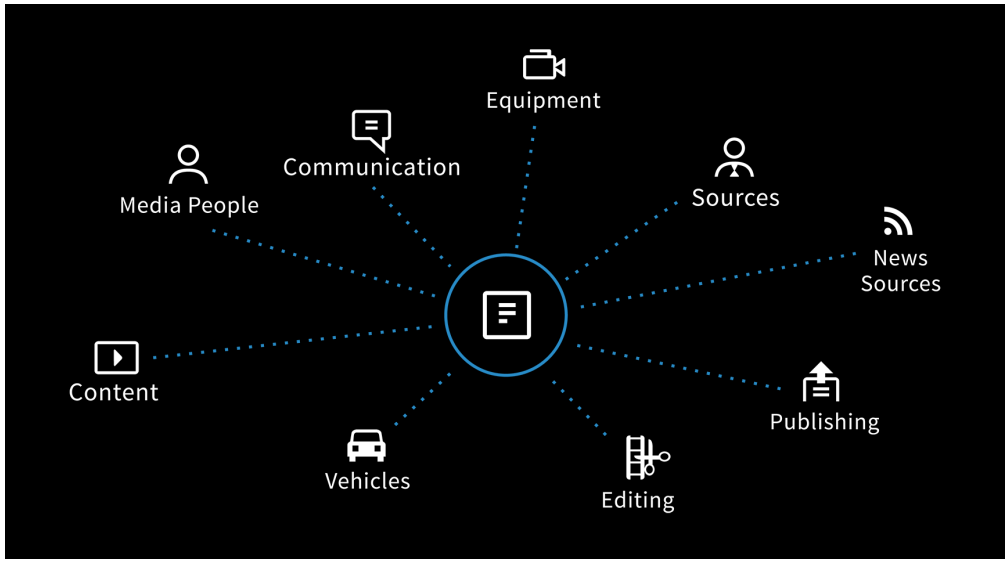

Figure 1: Main features of Wolftech news (Wolftech, 2016)

2.5 News Hunter

News Hunter started as a collaboration between the University of Bergen (UiB) and Wolftech. The main goal was to use semantic technologies and natural language processing to dissect and analyze a stream of news articles written by in-house journalists, as well as news articles from other sources. The system was then to semantically annotate the various entities and extract keywords for the story. After the system dissects the messages and categorizes them, it can then be able to deliver up to date and relevant information from a variety of sources to the journalists in real time, providing them with a better tool for writing their own stories (Christensen and Villanger, 2017).

2.5.1 Previous Work

System - 1. generation

As Christensen and Villanger writes in their thesis, Wolftech had previously made some efforts to semantically annotate content. The first prototype of News Hunter focused on gathering and marking up posts from Facebook. The prototype was called WT Semantic Prototype and it gathered Facebook Posts, sent them to Google's API for language translation of posts not already in english. The translated text was than parsed through an online analyzer for additional metadata extraction. The data was stored as triples in a semantic graph database, with all the entities, keywords and sentiment score connected to the original post. This first prototype was written in C# with the .NET framework, and the semantic graph database used was BrightstarDB (2017).

System - 2. generation

The second generation of News Hunter, developed by Christensen and Villanger featured a fully working web application. The application lets the user browse through different news stories and get information about which entities and keywords the articles contain. The user can also write their story and get additional information about their entities returned. This version however does not support an automatic analysis while writing.

That is one of the main contributions done in this more useful and enhanced version of News Hunter.

Christensen and Villanger had most of their focus on working on the backend in the 2. generation of News Hunter. A lot of the more computational and backend heavy programming had therefore already been implemented by them.

For analysis of categorizations, sentiments and entities, a variety of machine learning algorithms were implemented by Christensen and Villanger. They worked with machine learning libraries written for Python, such as:

- AFINN[11]
- Spacy[12]
- Textacy[13]
- Scikit-learn[14]

AFINN (2011) was implemented for sentiment analysis, and has been used further on when displaying sentiment of news articles written in the editor. Spacy is a natural language processing library for Python and it has been and is still being used to extract entities from the editor. Textacy is a wrapper library for Spacy, and it has been used by Christensen and Villanger to extract keywords and key phrases from content gathered through the RSS feeds. Scikit-learn was chosen as the preferred machine learning library for Python. It is used in News Hunter to classify the news articles, as well as automatic classifications when a user writes a new story in the editor.

2.5.2 Related Work

UiB and Wolftech developing News Hunter is not the only work that have been done on semantic news aggregation. There has also been other projects worth mentioning to get

[11] https://github.com/fnielsen/afinn
[12] https://spacy.io/
[13] https://pypi.python.org/pypi/textacy
[14] http://scikit-learn.org/

an idea of the related work in the field.

BBC

BBC was one of the adopters of linked data. Already since 2010 have the sports pages of BBC.com used semantically linked data. BBC also provides their up to date-ontology to the public, which lets anyone else describe their data with the same ontology as BBC uses (Angeletou, 2014).

Kobilarov et al. (2009) has also researched how the BBC has been actively working on integrating their different categorization systems by integrating their data with links to DBpedia (Auer et al. (2007), as cited in Kobilarov et al. 2009) and Musicbrainz (Swartz (2002), as cited in Kobilarov et al. 2009).

NRK

NRK (Norwegian National Broadcaster / Norsk rikskringkasting AS) is also one of the contributors to the field. They use semantically linked data stored in a graph database to keep track of every program produced for both radio and TV. They use OpenLink Virtuoso, which contains over 2.8 petabyte with sound files. By semantically linking the metadata of each production, they are able to create tools for finding and gathering old productions easily. This has opened up a lot of new possibilities for the journalists in their daily work (Børdalen, 2017).

NASS - News Annotation Semantic System

Garrido et al. (2011) has developed a system called NASS, seen in figure 2, which attempts to classify news documents faster and more reliable, than their human counterpart. They have worked with real world news paper agencies in Spain to train a model which can than classify a number of news documents automatically, obtaining really promising results.

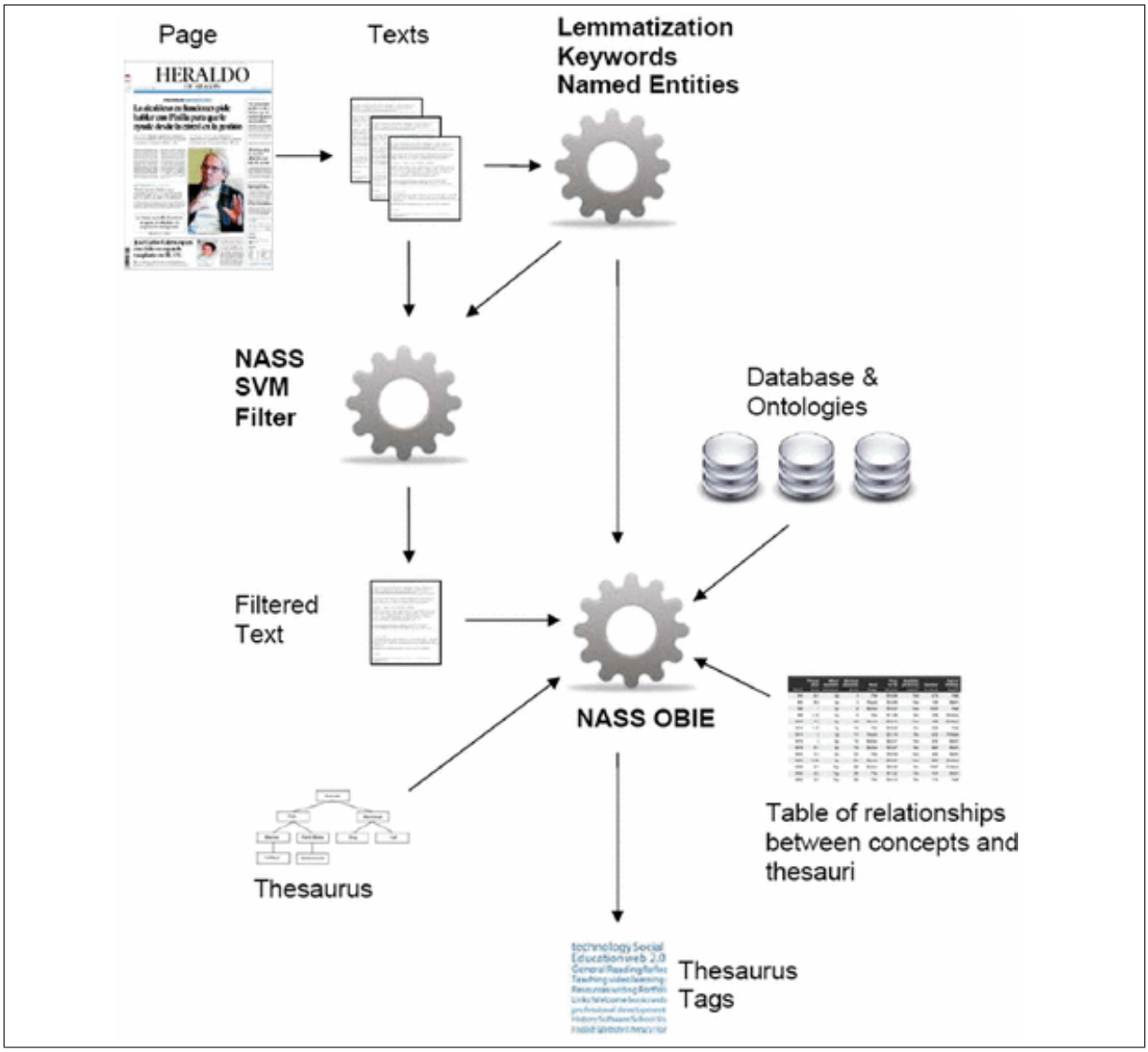

Figure 2: Overview of NASS

Aggregation of news

Since more and more news articles are published online, the border between each new publication vanishes. News readers are now able to get a constant feed of news delivered to their phone, desktop or tablet. With such an ever increasing degree of news it becomes impossible to filter through it all. Semantic news aggregation tries to cope with that problem, by aggregating news based on keywords, categorization and entity extraction.

Krstajic et al. uses in their research, data from the Europe Media Monitor (EMM) to analyze and visualize news from 2500 sources in 42 languages (Krstajić et al., 2010). The data is fetched at the instant it is available on EMM's servers due to their architecture, which lets their tools analyze the data in near real time.

Bergamaschi et al. has extended the tool *RELEVANT* to automatically group news related to the same topic, but from different sources into a web feed reader. *RELEVANT* is a tool that computes the relevant values based on strings. RELEVANT is then used to cluster and group news articles based on their title, both syntactically and by lexical similarity (Bergamaschi et al., 2007).

3 Research Methods

For this project, I have chosen to use qualitative methods and design science research. I have chosen these two research methods, as a means to get valuable feedback from users of News Hunter. The data gathered from this type of research can be seen as connected to ideas and concepts developed in News Hunter, as stated by Lawrence Neuman (2013). I chose qualitative methods because the main goal is to make News Hunter more useful and easier to use, and I think that by doing focus groups, listening to what the users are saying about the new features of News Hunter is a very valuable asset for further development. Since this thesis does not cover the user experience aspect, quantiative and measurable methods has been omitted.

3.1 Qualitative methods

Qualitative methods lets us reflect about the data before and after the data collection process. Qualitative data are often obtained through interviews, and the method will produce more complementary data than just quantitative data. This complementary data will let us reflect on it simultaneously and generate new ideas for News Hunter at the same time (Neuman, 2013). Ideas build upon existing ideas, and qualitative methods will be a well suited choice for the type of evaluation planned at the end of development.

3.2 Design Science Research

This project has used the design science research approach. Information systems (IS) is an interdisciplinary field, where a lot of different processes intertwines. IS is therefore composed of a mix of both hardware, software, different design processes and humans all trying to make a viable product that can help solve real world problems (Hevner and Chatterjee, 2010).

Hevner and Chatterjee says that the design science research paradigm is highly relevant to information systems because it directly addresses two of the key challenges of the dis-

cipline: the central, but somewhat controversial role of the IT artifact itself in IS research, and the lack of professional relevance of research (Hevner and Chatterjee, 2010).

Herbert Simon has in his book *The Sciences of the Artificial* conceptualized design science, and he supports a pragmatic research paradigm that calls for creating innovative artifacts that solves real-world problems (Simon, 1996). Design science research is therefore a process that combines a focus on what is being made, with a high relevance of the application domain and why it should be made (Hevner and Chatterjee, 2010). It is important to note that in design science research in the information systems field, there is a loop going on, depicted in figure 3, between design science research and behavioral science research. If we do not understand how our artifacts are being utilized we are in a weak spot to make any assumptions about what systems or artifacts we should create.

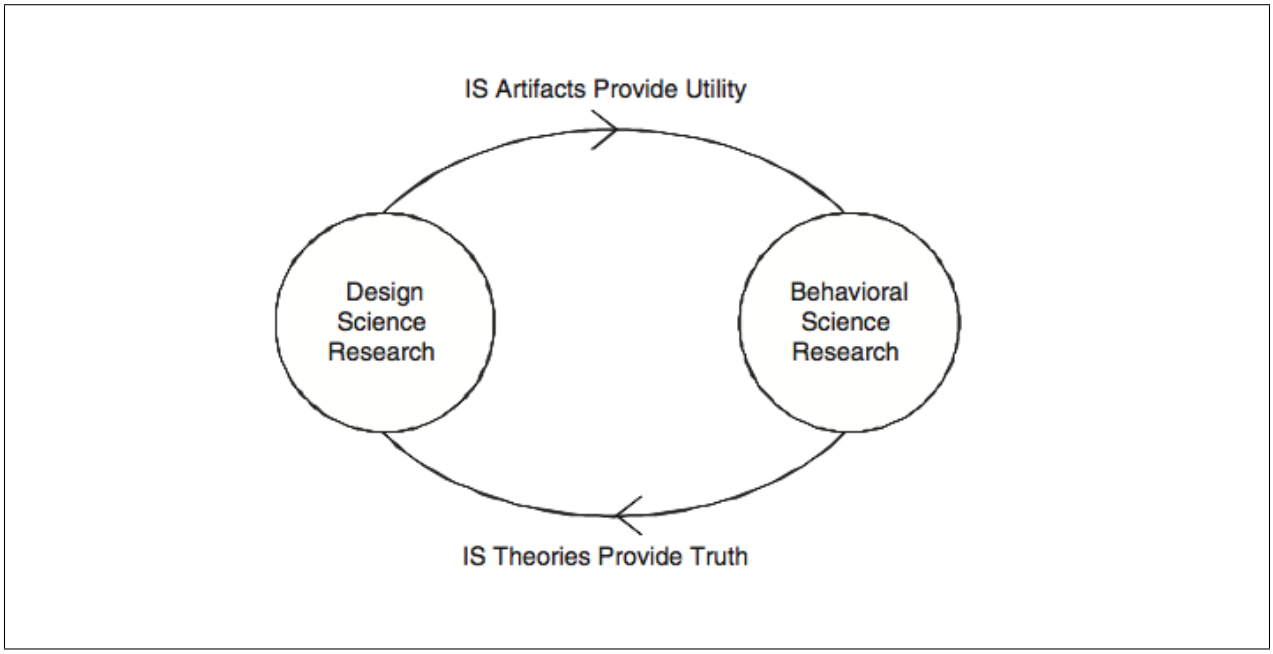

Figure 3: The interconnected loop between design science research and behavioral science (Hevner and Chatterjee, 2010)

Hevner et al. (2004) provides a table of guidelines for how to conduct, evaluate and present design science research to information systems researchers and business managers. The table can be seen in table 1.

Table 1: Guidelines for Design Science Research (Hevner et al., 2004)

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

3.2.1 Guideline 1: Design as an artifact

As the table states, the first guideline is to produce a viable artifact. In this case, it is the web application which lets users interact with the data analyzed and gathered from News Hunter. It is important to note that an viable artifact does not have to be a fully operating and working system. A viable artifact is something that can be tried out and is something that lets users interact with it in some way. This project already had a bigger surrounding system built around it, but for this third iteration of News Hunter, the viable artifact is new functionality the user can try out and test.

3.2.2 Guideline 2: Problem relevance

The second guideline tells us to focus on solutions that are based on relevant and important business problems. News Hunter focuses on the problem of time and information

gathering when journalists and graphic reporters need to get a hold of different sources of information instantaneously. Instead of going to lots of different websites for information, News Hunter's relevance is that it tries to combine various information sources into one.

3.2.3 Guideline 3: Design evaluation

Hevner et al. writes that the utility, quality and efficacy of a design artifact must be rigorously demonstrated via well executed evaluation methods (2004). In this thesis, the design evaluation has been done with end users of the finished product and the product owners represented by management for Wolftech Broadcast Solutions.

The evaluation was conducted in smaller focus groups, where the participants got an in-depth demonstration of all the features of News Hunter, with the opportunity to come with their honest feedback throughout the session.

3.2.4 Guideline 4: Research contributions

The fourth guidelines says that we must provide clear and verifiable contributions. These contributions are done by making this thesis public and letting other researchers know what technologies and systems have been developed to get News Hunter up and running. This project also has documentation in regards to how to run the system locally and what prerequisites needs to be in place for successfully further enhance the system.

3.2.5 Guideline 5: Research rigor

This guideline talks about how we need to be working in a strict and consistent manner to perform the best possible design science research. In this case, the construction of the artifact has been done using a lean and agile method and the evaluation of the artifact has been conducted with qualitative research methods. With only one person developing, it is hard to be rigorous about the lean methodologies.

3.2.6 Guideline 6: Design as a search process

The sixth guideline is all about how we gather information during development within the bounds of the laws of the environment. For this project, Wolftech has been a valuable source of information. They have provided a lot of constructive feedback about what their customers want and ideas on how they can achieve the desired outcome.

Both journalist- and graphic reporter colleagues at TV 2 Norway provided a lot of good ideas before development. They have also been the main source of motivation for this work, by supplying the thesis with observations through their daily use of their current tools at work. I did also gather empirical data from an earlier survey for the potential end users/colleagues before development started. I will talk more about this survey later on.

My fellow students have been part of the process by giving positive feedback in the process of formulating what the thesis is trying to achieve. The students has also been a good arena for discussion without limitations, to spark the creative process before and during the software development.

During the work on the thesis, my thesis advisor, Andreas Lothe Opdal have provided both good ideas and constructive criticism during development. Being able to discuss with Andreas during development has been an invaluable source of new input and feedback on the current state of development.

The whole search process has not only been a process to find out what features should be included in this enhanced version of News Hunter, but it has also served as a purpose to test and prototype with new technologies. Wolftech wanted to test a new WYSIWYG-editor, Froala, which got implemented. Another new introduction of technology to News Hunter was the addition of Elasticsearch, which has a good API for searching through a corpora of text.

3.2.7 Guideline 7: Communication of research

Lastly, the project must communicate both to technology-oriented and management-oriented individuals. This is achieved by both explaining what has been used of tech-

nologies and how they have been used, as well as writing more descriptive about why it has been done the way it has been done. We can separate the two distinctions into a white-box tech view, focusing on the system internals, and a black-box view, focusing on the usage context. The white-box view lets technically skilled users with a technology and programmer background understand the internals of News Hunter and how it all connects together. The black-box view lets management and organizational subjects focus on how News Hunter may aid their news journalism development in the context of a news desk.

3.3 The Technology Acceptance Model

This sections talks about the different technology acceptance models used in the research. This project follows the same path as the previous iteration of News Hunter, mainly by focusing on TAM and TAM2. By using the same models it will be easier to to compare our results with earlier results.

3.3.1 Technology Acceptance Model - TAM "1"

Fred Davis introduced TAM (Technology Acceptance Model) for the first time in 1989. TAM is a theory covering information systems. TAM models how users use and accept a new technology. Davis and the model suggests that when users are presented with a new technology, a number of factors influence their decision about how and when they will use it, notably:

Perceived usefulness (PU) - "The degree to which a person believes that using a particular system would enhance his or her job performance" (Davis, 1989).

Perceived ease-of-use (PEOU) - "The degree to which a person believes that using a particular system would be free from effort" (Davis, 1989).

TAM is an extension of theory of reasoned actions (TRA) developed by Icek Ajzen and Martin Fishbein in 1967. It is the most widely adopted and used models of users' acceptance for new technology usage of said technology (Venkatesh, 2000). TAM is more directed towards technology and replaced a lot of TRA's attitude measures with more technology related measures such as *ease of use* and *usefulness*. Where TRA is very general and was designed to explain virtually any human behavior (Ajzen and Fishbein, 1977), TAM is specifically designed to apply to computer usage behavior (Davis et al., 1989). TAM is a good tool to use when we want to predict future user behavior when the user has only tried the system for a small amount of time, such as with a trial of the program or a prototype still in development (Davis et al., 1989).

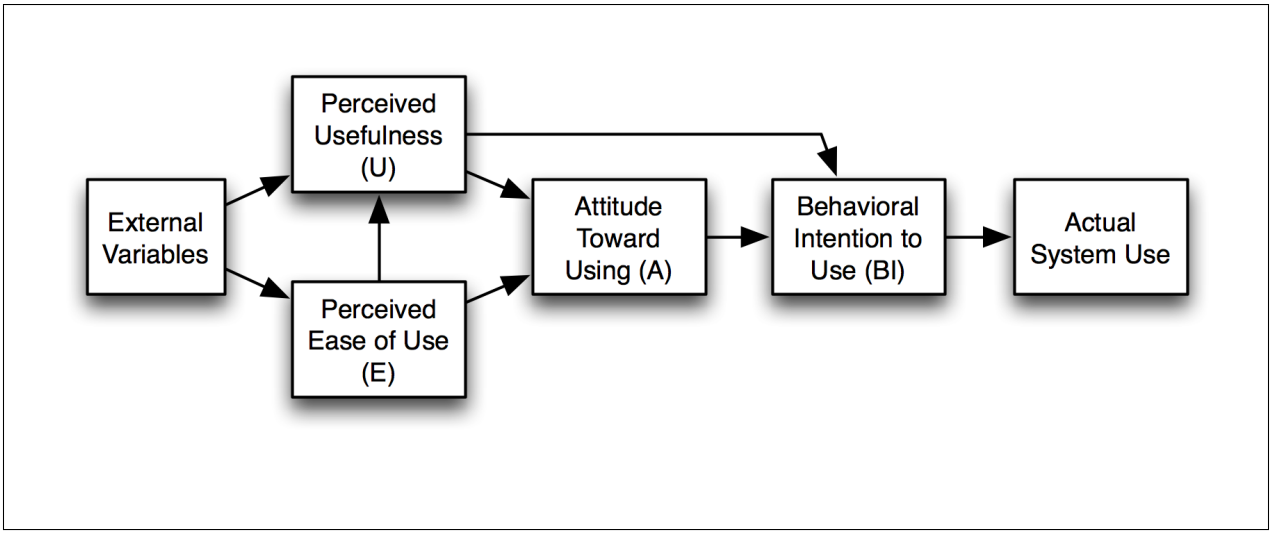

Figure 4: Technology Acceptance Model (Davis, 1989)

As figure 4 shows, TAM starts off with some external variables, which in turn are interpreted in the main features of the model, *perceived ease of use (PEOU)* and *perceived usefulness (U)*. Perceived usefulness is also influenced by perceived ease of use. If a user thinks that some program or feature will increase his or hers performance at work then perceived usefulness will be high. If the user thinks that the program or feature will work with little to no hassle and effort, then perceived ease of use will be high (Davis et al., 1989).

3.3.2 TAM2

TAM2 was developed by Venkatesh and Davis as an extension to TAM. TAM2 tries to explain perceived usefulness and usage intentions in terms of social influence and cognitive instrumental processes (Venkatesh and Davis, 2000).

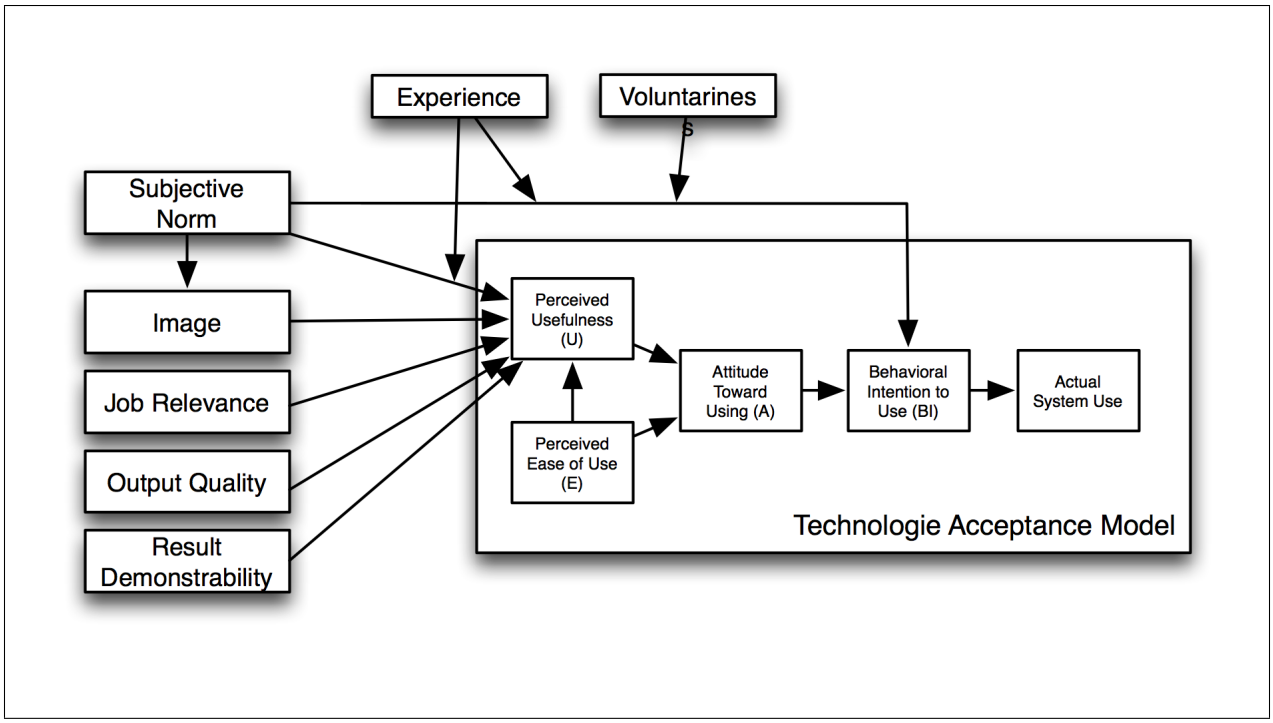

Figure 5: TAM2 - Overview of the extension of the Technology Acceptance Model (Venkatesh and Davis, 2000)

As figure 5 depicts, there are more external variables influencing the perceived usefulness of a system in TAM2. TAM2 categorizes between social influence processes such as

- Subjective norm
- Voluntariness
- Image

and cognitive instrumental processes such as

- Job relevance
- Output quality
- Result demonstrability
- Perceived ease of use

Subjective norm

Subjective norms says something about the intentions of the user, based on the users perception that those people that are important to him think he should or should not perform an action (Venkatesh and Davis, 2000).

Voluntariness

Venkatesh and Davis argues that there is a distinction between when a system is made mandatory by someone who can either reward or punish, versus when the system is made available at your own will and the user can use it voluntarily (Venkatesh and Davis, 2000). An example would be the workplace, where users must use a certain software or else they might get fired, or when a user chooses by their own free will to use a system at their own expense in their free time.

Image

When users adopt the same system, they form a group of end users, using the same system. This in turn, creates an image of each users, which is how the other users and adopters see and interpret each other. By using the system, the user is able to "achieve" membership with other colleagues in a positive way. This will in turn make every user more confident in their jobs, by being in the same group and being affiliated with the same use of the system.

Job relevance

Job relevance in TAM2 covers how well a system is at supporting the tasks a user have to do for a specific job. The more relevant a system will be for the tasks at hand, the more relevant will the user feel that the system is (Venkatesh and Davis, 2000).

Output quality

Output quality is a measure of how well a number of tasks are performed using a system. The user knows what tasks he or she needs to do, but if the system performs the tasks poorly, then the system will have low output quality for the user (Venkatesh and Davis, 2000).

Result demonstrability

If the user can see results immediately when using a new system, then the user is more likely to accept the system. This means that if a new system is being set into production, and the user can easily see how the system helps them get things done in an effective and safe way, then the users perceived usefulness will also be influenced (Venkatesh and Davis, 2000).

Perceive ease of use

Perceived ease of use is transferred into TAM2 as a direct criteria from TAM. The more easy and effortable it is to use a system, the more the system is being used will have an outcome of greater productivity (Venkatesh and Davis, 2000).

In 2003, Venkatesh et al. reviewed a set of different information technology acceptance research models. They analysed eight different models, compared them and then proposed a unified model based on the eight models studied. This new unified model was called UTAUT and is an acronym for *Unified theory of acceptance and use of technology* (2003). This thesis only focuses on TAM and TAM2, which are the same models used in the previous iteration of News Hunter. However, the reader should be aware that there is newer models also used in the research community.

4 Software Development

The development process of this project followed a lean and agile approach. It followed an iterative development cycle where features were developed throughout the process. This lead to better feedback from Wolftech early on, and a healthier discussion about the upcoming features all the way to the end of the development life cycle.

4.1 Project Inheritance

This project is the third edition of News Hunter. It was started by Wolftech in collaboration with the UiB as a means to discover how to use semantics technologies to provide better working tools for news journalists. It was picked up as a master thesis in 2016/2017 and then continued on in august 2017 to June 2018. As Christensen and Villanger says, the prototype they implemented was influenced by previous work done at Wolftech (2017). Even so, Wolftech as a company needs to renew its products to keep following the trends in the digital transformation. It has therefore been a process to develop News Hunter even further to accommodate the rising needs of digital journalists. The motivation and goal through this third iteration has therefore been to make News Hunter even more useful and easier to use.

When the project was continued, it lacked some documentation which made it more challenging to get started. I had some very constructive and informative discussions with both Christensen and Villanger after starting the project to get their feedback on what problems they knew they had when finishing up and handing over their work of the project. They were a great resource in the start to get the project working on my development machine, and to offer tips and tricks on how to get started. When development eventually started, I tried to have in mind the following quote:

> "Always leave the code you're editing a little better than you found it"
> - Robert C. Martin (2009)

When a project of this size is passed along to a new developer, it often takes some time

for the developer to get comfortable with the code written by others. This task was made especially hard because of the lack of documentation. No documentation meant that a lot of time was spend reading through all the files to get a grasp of the whole system and how each piece connects to create the application. There has been a great effort put into documenting and refactoring the existing codebase for further development in the future. The project and code repositories have far more documentation on how to get things started, which hopefully will make the next developer more productive earlier on.

4.2 Prerequisites

When starting this project Wolftech and I had a meeting where we discussed what I had in mind for this project, as well as what they were thinking. For Wolftech, this project is more an exploration of what is possible with semantic technologies, and therefore I as a developer stood somewhat freely to choose what features I wanted to work on and at what pace. However, that does not mean that Wolftech will just abandon the project after the project has ended. It is rather the contrary, where Wolftech hopes to be able to extract the features they seem most fitting to their own codebase in the future. So even thought I could choose the features I wanted to work on in my own pace, they still had some guidelines that I should follow. Those guidelines were to use the previous project delivered by Christensen and Villanger (2017) and to keep using C# as programming language.

As I wrote in the research methods chapter, I conducted a survey as a step in the *design as a search process*-phase. The survey helped to get an overview of how the end users felt about certain aspects of their daily tasks at work. I showed Wolftech the results from this survey and together we came to a conclusion about what features they thought would be good to work on first, before meeting again later during the development.

The survey among potential end users were a means as how I could get their opinion on what they felt about the current system and how things were being done. I asked them five questions:

1. How often do you feel that you are doing repetitive tasks at work?

2. Du you often feel that the tasks you do are without meaning? E.g create graphics which will never be used
3. Do you think it is hard to reuse data from previous productions?
4. Du you feel that the way you as a graphic reporter works leads to errors on air? E.g reuse of data you thought was updated, but was not?
5. Have you ever wanted a better and more separate system for reuse and update of system? E.g a database independently of what you use on air?

The questions were formulated to get an initial feeling about how colleagues and potential end users experienced their workflow at the moment in TV 2. They were conducted at a preliminary stage of this research and were used as a main selling point for what direction this thesis should continue on, and what features we wanted to explore and develop. From figure 6, the questions are positioned from left to right, top left being the first question in the survey.

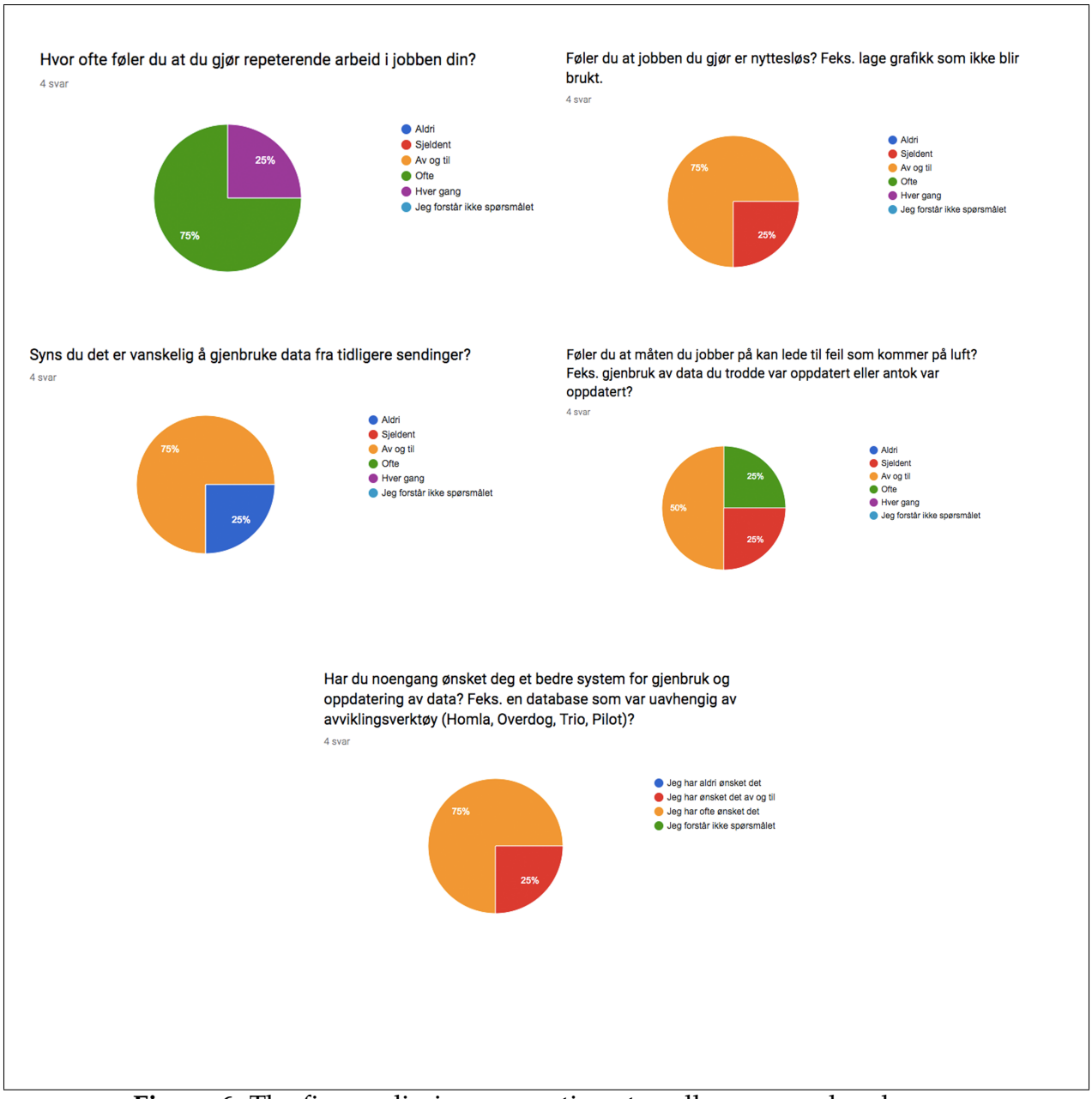

Figure 6: The five preliminary questions to colleagues and end users

The questions were asked in Norwegian, and they translate to the following:

How often do you feel that you are doing repetitive tasks at work?

For this question, 3 out of 4 answered that they often felt they did repetitive tasks that they thought could be automated while one felt he did repetitive work every day at work.

Du you often feel that the tasks you do are without meaning? E.g create graphics which will never be used

3/4 felt that they sometimes felt that what they did at work was without meaning.

Do you think it is hard to reuse data from previous productions?

3/4 felt that it sometimes was hard to reuse data from previous productions.

Du you feel that the way you as a graphic reporter works leads to errors on air? E.g reuse of data you thought was updated, but was not?

This question lead to more diversity in the answers, which I think is related to how long my colleagues have been working as graphics operators. One person answered that he often felt that the way he worked lead to errors, while another answered on the opposite side of the scale, answering that the way he worked would not lead to any errors.

Have you ever wanted a better and more separate system for reuse and update of system? E.g a database independently of what you use on air?

3/4 answered that they have often wanted a more robust and better system for storing data which is used during the productions.

The survey helped both Wolftech and I to get a grasp for what needed to be done first in the development phase.

4.3 Handover

Starting fresh on a project someone else has worked on can be a daunting task. With tight deadlines and by not always following best practices, it can be cumbersome for new contributors to add any value right away. Handover of this project consisted on getting access to all of the source code, a quick demo of the second generation of News Hunter and getting answers to some technical questions over chat. The remainder of the time was then used to figure out how everything connected and worked together. In the previous developers defence, there was no indication given that a new student would inherit the system, but in hindsight the project will always be rewarded if contributors can assume that someone else will one day take over development.

4.4 Technical Debt

When this project was continued in August, it started with some technical debt. the lack of documentation and tests made it harder to start development right away. This meant that everything done in August and the start of September was exploratory and any code written would potentially introduce even more technical debt. As it was developed on further, technical debt did keep accumulating in the project. Because this project has not been shipped to any end users during development, the technical debt was kept at a low volume, but it is still important to think about during the development phases.

Ward Cunningham, the coiner of the term technical debt has this to say about the topic

> "With borrowed money you can do something sooner than you might otherwise, but until you pay back that money you will pay interest.
>
> I thought borrowing money was a good idea. I thought that rushing software out the door to get some experience with it was a good idea. But that of course you would eventually go back and as you learned things about that software you would repay that loan by refactoring the program to reflect your experience as you acquired it."

and it can be summed up by the words of Letouzey and Whelan (2016):

> When taking short cuts and delivering code that is not quite right for the programming task of the moment, a development team incurs Technical Debt. This debt decreases productivity. This loss of productivity is the interest of the Technical Debt.

Technical debt can drastically reduce the development speed if it is not being considered and taken care of. If it is not acknowledged as a problem before it is too late, it may take much longer when refactoring the code when implementing new features.

4.5 Refactoring

Refactoring is the process of changing a software system in such a way that it does not alter the external behavior of the code yet improves its internal structure (Fowler and Beck, 1999). When this projected was continued in August, it would have been a nightmare to refactor. No documentation or tests, and little-to-no understanding of the internal system worked meant it would be nearly impossible to add new functionality without breaking the code. This meant that I had to dive straight into the code and it was just a matter of time before the code would break. The positive side of doing it this way was that I eventually understood all of the program and was then able to implement new features in the code.

4.6 Development Process

The development process of this application has been an agile one with hints of Lean and Kanban development. With only one developer on the team, a few of the key characteristic for these approaches have been undermined or left out, such as a daily stand up meeting and continuous integration with the customer. There has not been a customer present at all times, but the process has still tried to deliver new functionality as the customer demands it, without a long and tiresome process before writing code and delivering it.

For keeping track of the whole project as it went forward, a Kanban board has been used

in Trello. Tasks have been registered in the backlog, and moved between the *backlog*, to the *doing* tab, to the *finished* and *testing* tabs. This has been of great value when developing and enhancing News Hunter, as it meant the project could focus on only the tasks at hand.

4.7 Functionality

The functionality of this development phase of News Hunter has been a collaborative process between Wolftech and I. The features here are all new implementations, except for the editor which was also present in the second generation of News Hunter, but has been remarkably upgraded in this development phase. Wolftech and I have discussed different opportunities in regards to my own skill levels, as well as their wishes and tried to come to agreements between their and my wishes throughout the period of development. This section will briefly describe the dynamic functionality that might not come across clear enough through screenshots of the application.

Live Editor

The live editor lets the user write a story, while simultaneously feeding the user with new information about the entities and keywords written in the story. Behind the hood, the editor waits for the user to take a natural pause in the writing. This ensures that the user does not hit the server with a request on every letter typed, as this might make the UI slow, and put unnecessary load on the servers. On the right hand side of the editor, there is a column showing the data fetched from the servers as the system has been able to figure out what entities and keywords the text is about. The user is able to change the confidence value for what entities should be displayed. The greater the confidence, the more confident the system must be in serving entities to the user.

There is also an update-button for the user to click if they change the confidence, but does not want to write any more in the story to trigger a new analysis.

The entities listed after analysis are also color coded, where a typical blue link color is displayed if there is additional data for the entity present, while a standard black color is chosen if there is no additional data. If the user clicks on a blue link, the application will

show an iframe from DBpedia with additional data about the entity clicked on.

Lastly there is a button for saving an article to an imaginary backend system, but as of writing this thesis there is no backend connected for saving these stories. This is a potential feature for future work, either by connecting to the semantic graph and saving the stories semantically there, or as plain JSON in Elasticsearch. I see this as an opportunity for discussion based on a number of factors, such as what backend system News Hunter would benefit most from, versus what backend system the company using News Hunter is using at this time of writing.

Underneath the editor, there is a drop-down component where the users can see the latest news from other news outlets. When the user clicks on a choice in the drop-down, the system requests the 30 latest news from the Elasticsearch database and displays them to the user, with links to the story on the domain of the news outlet.

Politician Overview

The politician overview page starts with a listing of all the countries where data about politicians are present. The names of the countries are presented as links, where clicking the link takes the user to a detailed view of the country chosen. The user is then presented with some graphs on the left, showing the distribution between men and women governing the politics in the chosen country. On the right, the user sees a map of where the country is located.

Underneath the graphs and map, the user is displayed with a table of the most recent government. The user can then filter on previous years if they want. In the table, the user is able to see a picture (if such is present in the data), the name of the politician, birth date, what party they are representing and links to Wikipedia if present. The links are filtered by the English, Norwegian (bokmål) and Norwegian (nynorsk) Wikipedia sites.

Invitation Builder

The invitation builder lets the user create an invitation that can be sent to a person before an interview is taking place. The send functionality is not implemented in this version of News Hunter, but based on current technology and state of development, that would be a trivial feature to implement. On the left hand side there is a form where the user can enter information about the interview object. On the right hand side there is a panel showing the latest news mentioning the interview object, which is meant to be a guide to what has been written about the interview object in the news lately. The user can filter on all sources, international sources or Norwegian sources. Underneath the latest news, the user is able to input the person's Twitter username to see the latest news from the interview object.

When the user "sends" the invitation, the invitation is saved to the semantic graph and the data is then available for other parts of the application.

Register New Person

The register new person feature lets the user add a new person to the semantic graph. When the data has been entered and the person has been saved, the data is saved to the semantic graph. As of this feature of News Hunter, person knows-relation between the person entered and the persons chosen by the user does not work properly, but that is definitely something that should be worked on as future work for News Hunter.

Graph

The graph visualization between people is meant as a way to let the user get a better visualization between who knows who before doing an interview. This way, if say a journalist does not have too much background information before taking an interview, they can take a look at the graph to get an overview before they might say something stupid to the interview object.

This is possibly the feature as of right now which should not be part of a production build,

but it is there as a prototype and to get feedback from user testing.

5 Iterations

Since this project has followed as a design science reserach method, and has been developed using an agile development method, the development process has been a series of iterations. Each iteration has been building on the previous one, each with a set of goals needed to be accomplished.

5.1 Iteration 1 - Overview and setup

The project I inherited was not very well documented. It lacked any sort of comments in the code, there was little to no readme documentation and the two persons last working on the project were no longer available for questions during development. This lead to the first iteration being a slow and cumbersome affair. The lessons learned from this first iteration is that it is nice when some documentation has been provided. It provides a more clear roadmap of what to expect of the code and what the code should do. Without this sort of documentation, it becomes a tedious task to figure out what the code does and how each component in the source code fits together.

5.1.1 Goals

The goals for this first iteration was to properly understand how to configure and setup the system, as well as get a understanding of the underlying code. The goals can be listed as following:

1. Get the system up and running
2. Develop an understanding for how the system works
3. Starting getting my feet wet by touching the code

5.1.2 Setup of Development Tools

The first thing needed to get this project started was to acquire a machine with Windows installed. Wolftech uses mainly C# in their day to day development, and therefore this project was also started as a C# project, which meant I would need a Windows machine to do the development. After notifying Wolftech about this, they want ahead and bought a Windows license for me, so I could use Windows on my Macbook Air.

Integrated Development Environment - IDE

The previous prototype for this project was developed using Microsoft Visual Studio IDE[15]. I decided to also use this IDE, as it would be the easiest solution to get everything working. For the Python development of this project I used a combination of PyCharm[16] and Visual Studio Code[17].

Semantic Graph Database - BrightstarDB

The project I inherited was already using BrightstarDB as their semantic graph database of choice, and I had no problem to continue using it. BrighstarDB is a native RDF database for the .NET platform. BrightstarDB has an extensive API for working with the database and because it is based on a semantic RDF graph it does not require any sort of database schema to get started. Adding and integrating data in various forms is an easy task, and one of the real strong benefits of using a semantic graph in this sort of project.

Programming languages

The programming languages used in this project is C# with the .NET framework, AngularJS and Python with the Flask framework. It should be noted that AngularJS is becom-

[15] `https://www.visualstudio.com/vs/`
[16] `https://www.jetbrains.com/pycharm/`
[17] `https://code.visualstudio.com/?wt.mc_id=DX_841432`

ing a deprecated framework and should in the future be upgraded to a newer version of Angular or another more modern framework such as Vue.JS og React.JS.

GIT - Version Control

For such a large project it is vital to have good versioning and code control management. This project uses Git as version control system, with Bitbucket[18] as the backend repository.

Trello - Kanban board and project management

The project has used Trello as its project management tool of choice. Trello is nice for keeping track of deadlines, user stories and to see what needs to be done at every stage of development.

Validation

The first iteration had the following goals:

1. Get the system up and running
2. Develop an understanding for how the system works
3. Starting getting my feet wet by touching the code

The first goal of this iteration, to get the system up and running turned out to take much longer than expected. This was mainly due to poor documentation and handover, but also as a direct result of not having hardware which the system could run on. The first step was therefore to install Windows on my machine, so that the system could be started via Visual Studio IDE. I then had to make sure that I was able to pull down news articles and semantically annotate them. I did this by starting in the Python code to see how it

[18] `https://bitbucket.com`

parsed the articles and how it eventually pushed it to other parts of the application to annotate them. It turned out that the Python scripts only retrieved and stored the news articles as JSON formatted data. It then took another Python script to send them to the C# application which would annotate them semantically. Because there was no flow diagram of the application, it turned out to take much longer than needed to figure this out.

After this the next step was to go through the code and visually and mentally create a picture of how data flowed through the system. This was also a task that took time, again because of bad documentation. After a few days of diving through the code, I found that a lot of the code was no longer in use and could therefore be deleted. After some refactoring of the system I was ready to start writing some code to understand that my beliefs and hypothesis about the system were correct.

The third step was to start getting my feet wet by touching and writing my own code. It started out as a painful experience, where code I wrote would not be registered at save, and so I often had to rebuild the whole project for every minor change I made. This made it a tedious and painfully slow experience. I eventually discovered how to enable auto-refresh and after some time the experience of developing on the application became a little less painful.

Results

The results of the first iteration of the project was both encouraging and frightening. Encouraging because I got to refactor a lot and make the code more "mine", frightening because the slow development experience and complete lack of documentation made the next iterations seem like a very daunting task. I eventually understood more of how the code was connected and what each component did. This made it easier to point out a course of action for the next iterations.

The goals for the iteration was to get the system up and running, developing an understanding for how the system worked and to start getting my feet wet by writing some code.

The results from the first goal were positive at last. After a somewhat troubling start, the

application and source code were eventually ready to be worked on and run on my own machine.

The second goal was also to a large extent done after the first goal was finished. Even though if was not clear what all the code was doing, the more time I spent with the code, the more obvious it became what each component did.

The third goal was a success when I was able to implement a filtering feature on the front page, where the user could then get to see a view of articles connected to a specific type of genre.

5.2 Iteration 2 - User Stories from Wolftech

The second iteration of this project started by meeting up with Wolftech to get their views on what to do next. Since they were the stakeholders of this whole project, it is viable that they provide their ideas, input and feedback. We had a good discussion about what my ideas were, what they wanted and we evaluated my skill set so to not bite over too much at once.

We got to an agreement where I would start this iteration by improving the previously developed editor. Wolftech had a vision where the user would get automatic feedback as they were writing their stories. The previous version of News Hunter would provide some feedback, but only after the user pressed a button. This is a disruptive move when the user is typing away a new news article, so this was the first step of this iteration.

We also discussed a bit about the research part of this thesis. We discussed machine learning, and made an agreement that I would look into it, but I told them that it might be changed later on in one of the next iterations.

5.2.1 Goals

The goals for this iteration was as follows:

1. Improve the editor functionality
2. Provide feedback as sentiment, keywords, entities and categorization

5.2.2 What have you done?

The editor functionality as it was before this iteration was just a simple text area where a user could enter text. In the meeting with Wolftech before this iteration started, they informed me that they had bought a license to use Froala[19] as their WYSIWYG[20]-editor.

Froala Editor

I had previously been working with Froala, so it did not take too much time to switch out the text area with the new editor. After making the switch I started to look into how to send the text for analysis automatically. Froala already provides methods for capturing data with ease, and one of those methods was to capture data when the user had a natural pause, e.g punctuation, thinking about what to write, etc. This made it easy to only send the text to analysis when the user naturally paused. It was typically quick enough to analyze it so that it did not feel as something the user had to wait for.

Analysis of Content

The analysis part of this iteration used a lot of the previous work done in News Hunter. The Python Flask API was straightforward to understand, so it did not take too much trial and error before I had a working analysis of the content being written in the editor. Since there was already code that did the analysis of text, I simply had to send the content as JSON to the API, and wait for it to being finished and analyzed. After that it was just a matter of making the result available to the user as can be seen in figure 7

[19] `https://www.froala.com/wysiwyg-editor`
[20] what you see is what you get

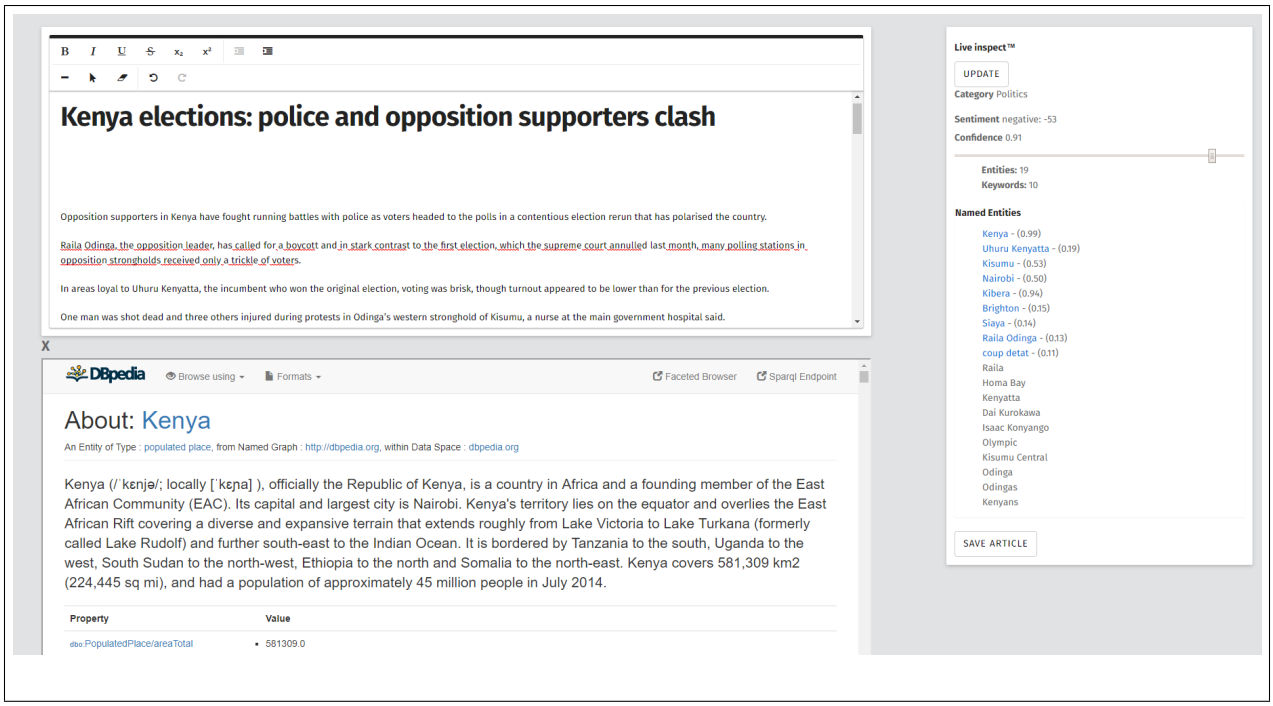

Figure 7: Editor view

Improving the User Experience

So now that the user is able to get information about categorization proposals, sentiment analysis on the text as well as a listing of keywords and entities, the next step was to improve the user experience. It would be nice if the data from the entities could be displayed inside the editor windows, instead of going back and forth between browsers. I therefore added an iframe to the editor view, so that the user could then easily click on any entity with a URL, and get their DBpedia page at an instant. This greatly improves the user experience, as the user is now able to write their whole story in the view, adding information as needed from DBpedia.

Validation

For this iteration to be successful, there was a few key steps that needed to be done. Firstly to switch out the text area with the Froala editor. Second to find out how to automatically analyze the text as the user writes their articles, and third to display the data to the user

in a nice and user friendly way.

Because I had used Froala before, it only took about an hour with some trial and error to switch out the text area and include the Froala editor, with a variety of formatting tools.

Since Froala already came packed with event listeners to automatically capture data when there was a pause in the writing, it made sending the content to analysis really easy.

When the data was analyzed, it was displayed to the user in a user friendly way by displaying the data on a panel to the right of the editor.

Results

The goals for this iteration were to change the text area from the previous version of News Hunter to a new and more user friendly WYSIWYG-editor, as well as further develop the user experience of using the editor.

Changing the text editor worked very well and improved the user experience to the better. It is now possible to add images with drag and drop functionality, format text and lists, as well as adding links directly in the editor. This would be seen as a minimum viable product for any user that wants to write their news articles on a daily basis.

Under the hood of this feature was also the ability to automatically analyze the text as it it being written. It does no longer require the user to press a button to analyze the text, but it is instead being analyzed as the user is working on their story.

As the user needs more information from their stories, it is now also possible to click on any of the entities being extracted from the analysis to display information from DBpedia automatically, while still working on the article.

The user is also able to get instant feedback on the sentiment of their writing, as well as under which category they should publish and save their article.

5.3 Iteration 3

The previous iteration ended with a new and more functional editor. It went from being a simple text area to a WYSIWYG-editor with automatic analysis of the content as it is being written in the editor. Before the start of the previous iteration, we started to talk about machine learning and automatic news categorization based on learning data. I did some investigation, but it quickly became clear that it would take too much time to potentially end up with a poorly implemented solution.

This iteration therefore started with a new meeting with Wolftech where we had a good discussion about what I wanted to do, and what Wolftech was looking for. We ended up with a few bullet points as can be seen below

- Twitter feed for sports athletes
- Show last news where the sport athlete has been mentioned
- A feature for building automatic invitation for interview objects
- Represent relations in a graph

This list included features that all would be feasible with semantic technologies, but because of uncertainty as how fast each feature would be developed, we decided to start working on two features first, then add the two features I would not finish to the backlog and the next iteration.

5.3.1 Goals

For this iteration, I decided to primarily focus on

1. Implement a Twitter feed for sports athletes
2. Represent relations between people in a graph

5.3.2 What have you done?

After the meeting and having the features written down, I started the development cycle for this iteration with having a look at the Twitter API and registering for API keys to fetch data. I had a look at some Python tutorials describing how to get data from the API, and went on with coding up the backend. I also had a look at D3.js[21] to see how I could build a graph for displaying the relationships between people. D3.js is a Javascript library for visualizing data. There is a great community using D3.js and a lot of tutorials and examples on how to model and display the data.

Twitter feed

The Twitter API has very good documentation, and it was easy to get it up and running. I implemented the fetching of tweets using an endpoint created in the Python flask code. I could then write the connection to the Twitter API in Python and send the tweets as JSON to the frontend. The tweets can be seen on the left in figure 8.

[21] `https://d3js.org/`

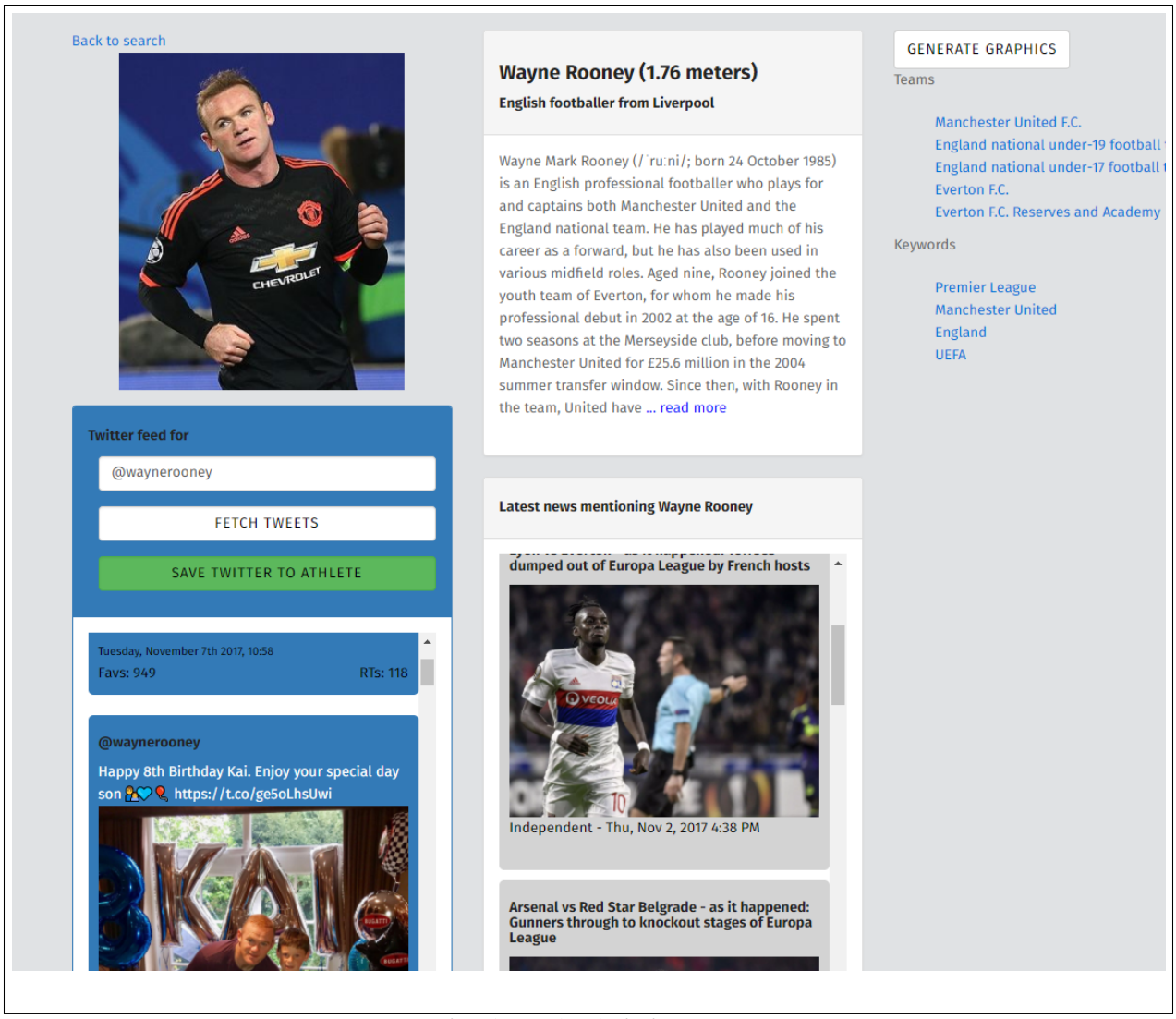

Figure 8: Twitter feed on the left for easy access to tweets

When the user has found the Twitter username for a user, the username is at this point in the development only being saved as a browser cookie. This is not a viable solution for a production environment, but it has been added to the backlog as something that can be improved in the future.

Relations in a Graph

The second feature in this iteration which was requested from Wolftech was the ability to display a graph of who knows who. This would be useful if a journalist which does not typically cover a certain topic (e.g. politics), needs to get out in the field to cover politics, but are not sure what to talk to a politician about. It would be useful for the journalist to take a look at who the politician knows, so they do not say something wrong to the politician. The prototype graph visualization can be seen in figure 9

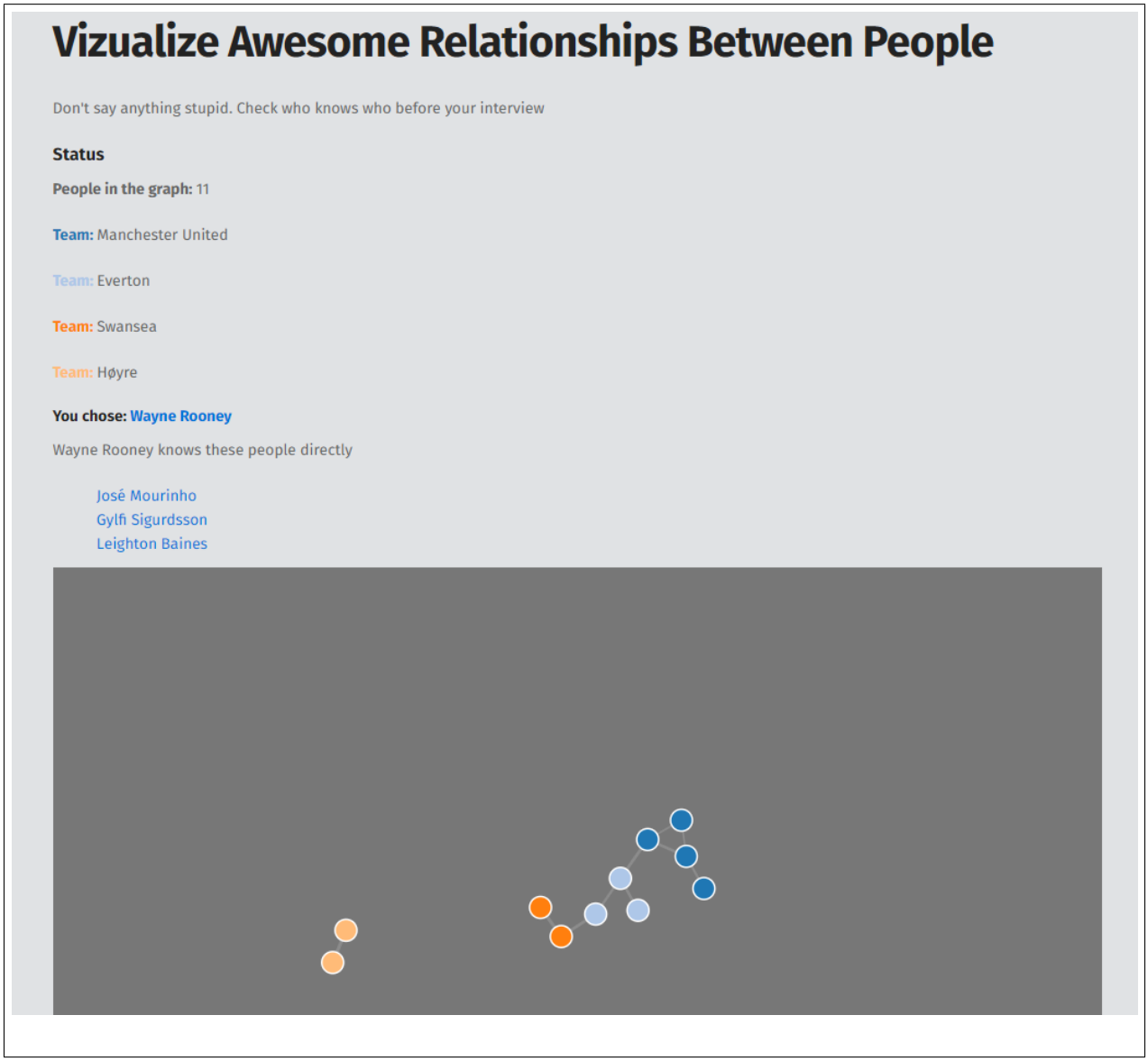

Figure 9: Graph visualization using D3.js

This feature, after this iteration, is the one that lacks the most features. It is possible to see the graph, but this iteration is only a proof of concept for the feature, and the graph only uses hard coded data. This is because I encountered some problems with how the data was semantically annotated. I followed a guide from the Semiodesk Trinity[22] website, which in theory should be able to properly annotate and infer a one-to-many relationship, but without luck it was hard to get it working, and because of a tight deadline I chose to just hard code the data for a proof of concept at this point in the development cycle.

Validation

The goals for this iteration were as follows:

1. Implement a Twitter feed for sports athletes
2. Represent relations between people in a graph

For validation of this iteration, I tested that each feature performed as expected without any visible bugs. For the Twitter feed, I ensured that tweets were being fetched when the user requested them. I tested that it worked to tell the system how many tweets they wanted, as well as setting a default of the 30 last tweets. Since the second feature became a proof of concept, I only tested that the graph was displayed properly using the D3.js library.

Results

As a result of this iteration, News Hunter now has two new features which will aid the user in the planning phase of creating a news story. By checking the last tweets from a sports profile, when doing a check on their data provides more up to date data about what the player have been concerned with in recent time.

[22] `http://www.semiodesk.com/products/trinity/`

For the proof of concept graph representation, the user/journalist can now do a quick check on what he should or should not mention when doing an interview, making sure he or she does not make a fool of himself in an interview setting.

5.4 Iteration 4

The last iteration started with a meeting with Wolftech, where we discussed the next features for News Hunter. We ended up with focusing on two features then, and then I would do the remaining two features this iteration.

The two features for this iterations were

- Implement a feature for displaying the last news where a person has been mentioned
- Implement a feature for automatic invitation building for interview subjects

5.4.1 Goals

The goals for this iterations were as follows

1. Implement a feature for displaying the last news where a person has been mentioned
2. Implement a feature for automatic invitation building for interview subjects

Last News

After implementing the Twitter feed, I started to map out what I would have to do to fulfill the next feature. The next feature would be an overview over last news where a person was mentioned. The feature implemented can be seen in figure 10. For the feature, I started to have a look at Elasticsearch, which I had heard from peers would be

a very good match for searching in large corpora of text. Elasticsearch has also very good documentation and it was easy to incorporate it in to the existing Python code.

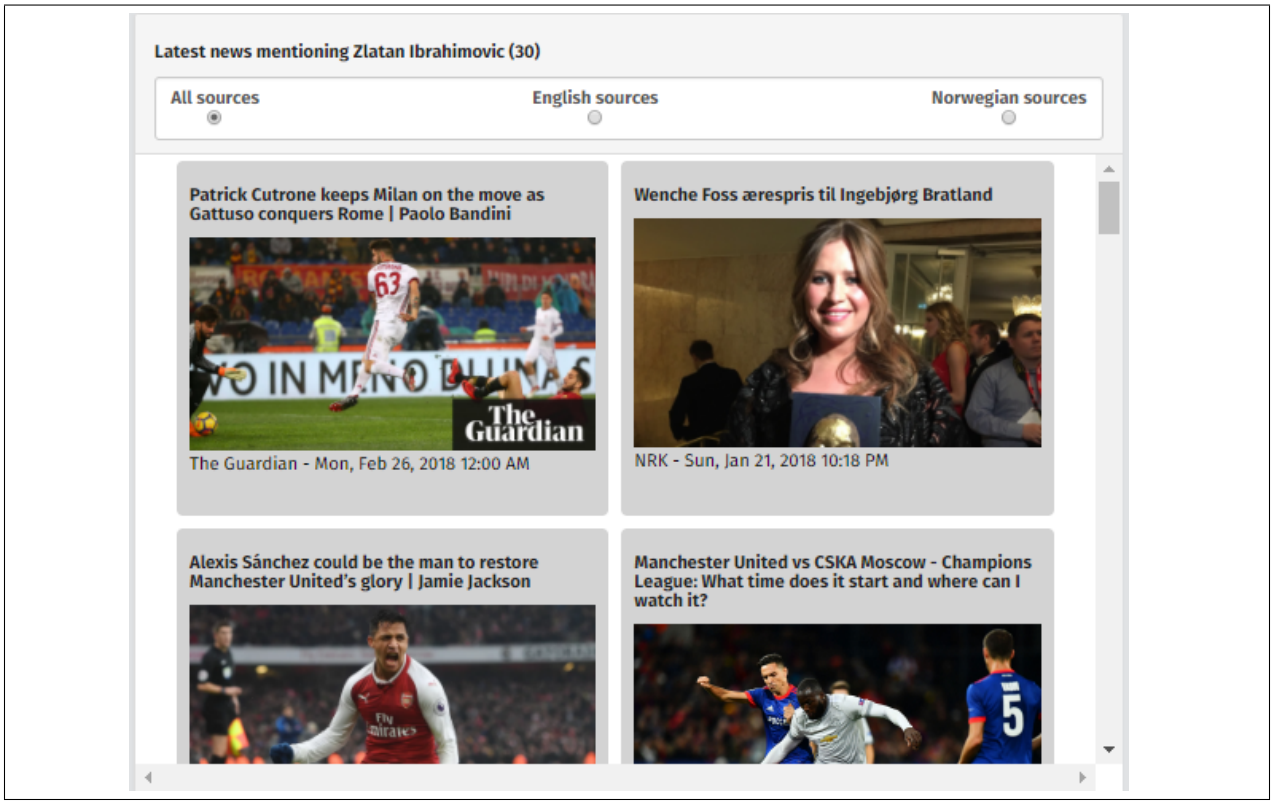

Figure 10: View of last news in connection with the invitation builder

This feature followed the same pattern as the Twitter feed. I implemented the backend code in the Python scripts where I also fetch news. I then wrote a separate function that would read all the JSON data and populate the database.

When populating the database, Elasticsearch can take an ID per JSON document, and then warns you if you try to populate the database with the same news article. This is nice if you want to have an automatic script running each day, but do not want to populate the database with duplicates.

Elasticsearch provides a very nice API for populating the data in the database, as well as querying the data. Elasticsearch provides a lot of query possibilities, and this feature in News Hunter only uses a very simple search. When the user requests the profile page for a profile, News Hunter requests the last 30 news where the name of the profile is either

mentioned in the title or in the body of the news article.

Invitation Builder

One of the more interesting features that Wolftech wanted to test out was an "automatic" invitation builder, which can be seen in figure 11. The idea is that when a journalist or other editorial worker needs to plan for either a TV interview or other form of interview, they can have a standardized form for creating this invitation. When the user types in the name of the interview subject, they get the latest news and can also type in the subjects Twitter name to get their last tweets. This would then enable the journalist to prepare some questions or write some notes that the interview subject can prepare.

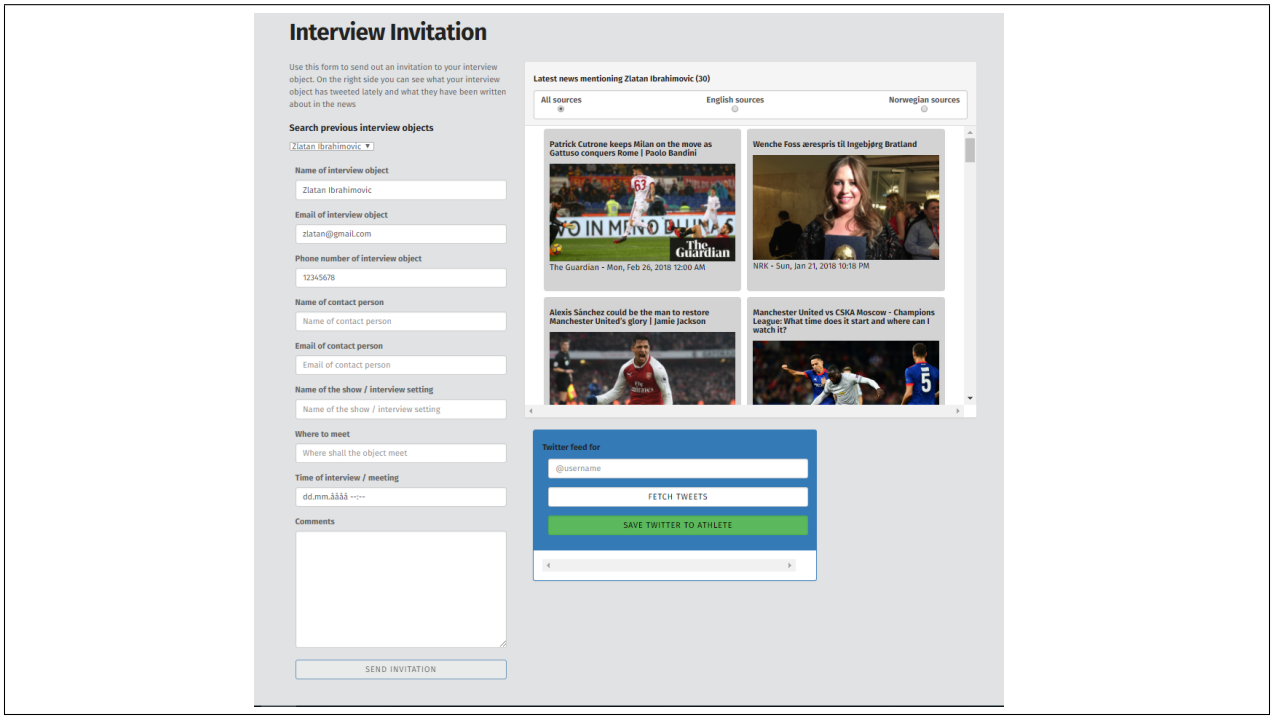

Figure 11: Form for creating invitations

This feature saves the invitation to the semantic graph, which means it can be queried later and used in other parts of News Hunter in the future. The Twitter username is not saved, and as for the profile page, this has been added to the backlog for a future iteration.

Validation

For validation of this iteration and the two new features I tested that the news being parsed from the Python script reflected the real world and I compared it with sport profiles I knew there had been stories about lately.

For the automatic invitation builder I made sure that the news fetcher and Twitter feed still worked when the user wanted to invite a interview subject.

Results

The results from this iteration showed that keeping a modular workflow when developing is key, as it makes it easy to integrate modules into new components. For this instance it meant it was easy to include the Twitter feed to the invitation builder.

The results also showed how powerful and easy to use Elasticsearch is. It can search through and fetch the last news for a given person very quickly, and it is easy to extend the filtering and querying of the underlying data at a later stage.

5.5 Iteration 5

The last development iteration was used to refactor, clean up and document the code better. Some of the implementations done in the previous iterations were not as good as I wanted them to be, or they had some minor bugs. I therefore took the time to finalize the development process by documenting and rewriting some of the components.

5.5.1 Goals

The goals for this iteration was to refactor the code to a quality level I was happy with.

1. Refactor code to be better and more readable.

Refactoring the Codebase

I started this iteration by going through each file in the project and read through every line of code. I would document where I needed to refactor the code, before mapping out which files would be touched by a refactoring. For example changing the way the backend sent some JSON data might mean I had to parse that data on the client side somewhat different. Knowing how all the code worked together helped me to not make too many mistakes while working.

Some parts of the code would only need to rename some variable names, while other parts was in need of me extracting a module to its own separate entity, typically a service which connected to some external data source. This would aid the next developer(s) working on the project to hook up new functionality, but still using the same data sources.

I used Standard.js[23] to help me refactor the frontend code for this project. Standard is a set of standard rules for formatting and code conventions. The tool can be run from the command line and parses each Javascript file and tells the developer what is wrong. It has a fixing feature which can take care of most of what is wrong with your code, but it also outputs on what line something is wrong, and why it is wrong.

Validation

For validation of this iteration I wanted to be certain that I had done a best possible job on refactoring. I had to make sure that the naming conventions were fulfilled throughout all of the code and that the code adhered to the same standard all over.

Results

The results before running the *standard*-command in the terminal window showed that it was hard to be sure I had adhered to the same standard all over the codebase. After running standard.js in the terminal window, the code became much more adherent to the

[23] `https://standardjs.com/index.html`

same standard of quality and it fixed a lot of small inconsistencies in the codebase.

6 Overview of each Component of News Hunter

This section takes a look at each component of News Hunter. It gives a thorough explanation of each component without going into too many technical details. The components listed are both new components developed during this development phase, as well as previous implementations. The previous components may have seen changes in the source code, but the main functionality still remains the same. Figure 12 shows an overview over the whole system.

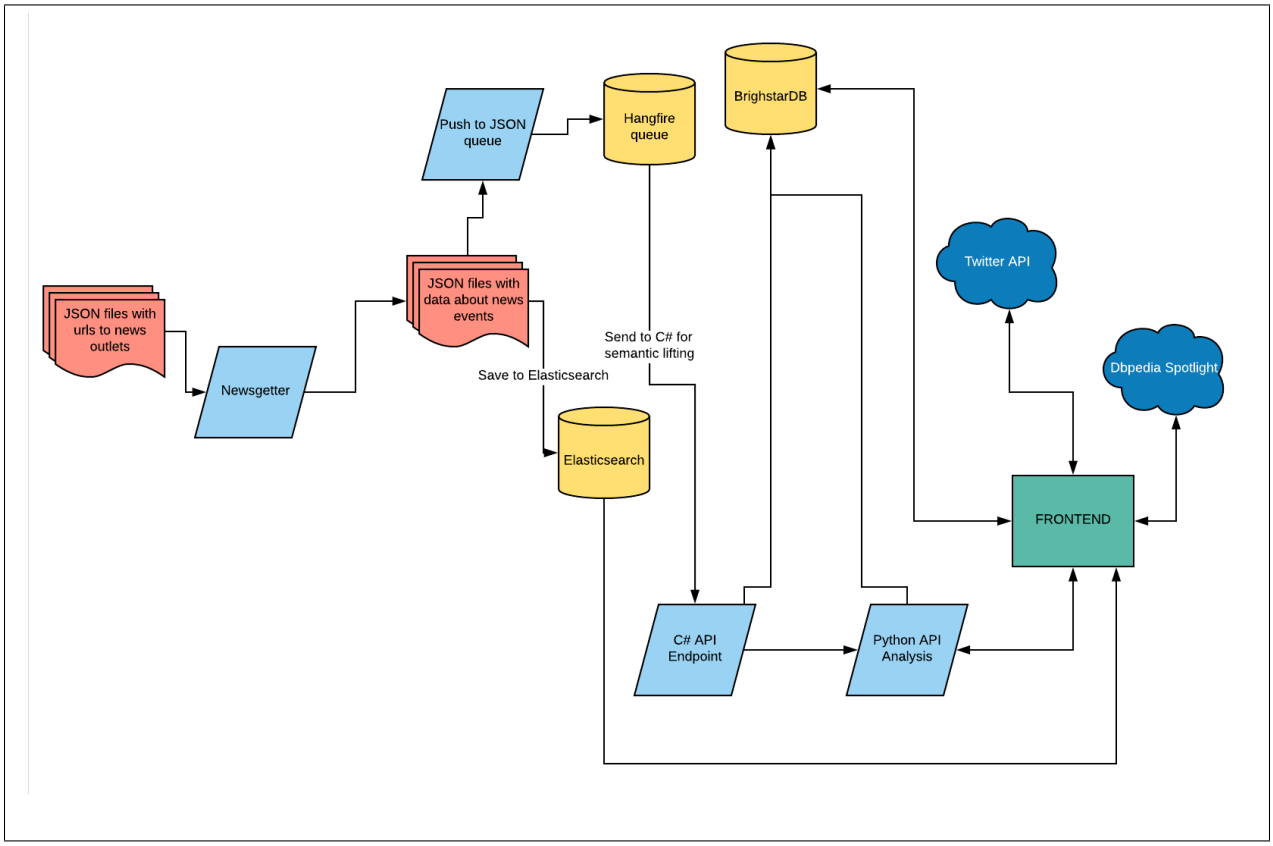

Figure 12: Overview of News Hunter

6.1 Newsgetter

The first steps of gathering data into News Hunter is the use of a Python script to fetch news from over 80 different categorizes from a variety of news sources, as well as all the

biggest news providers in Norway. The script downloads and parses RSS feeds which are in turn converted to JSON files and stored in their respective folder (e.g politics, sports, food, movies). The overview of the Newsgetter can be seen in figure 13.

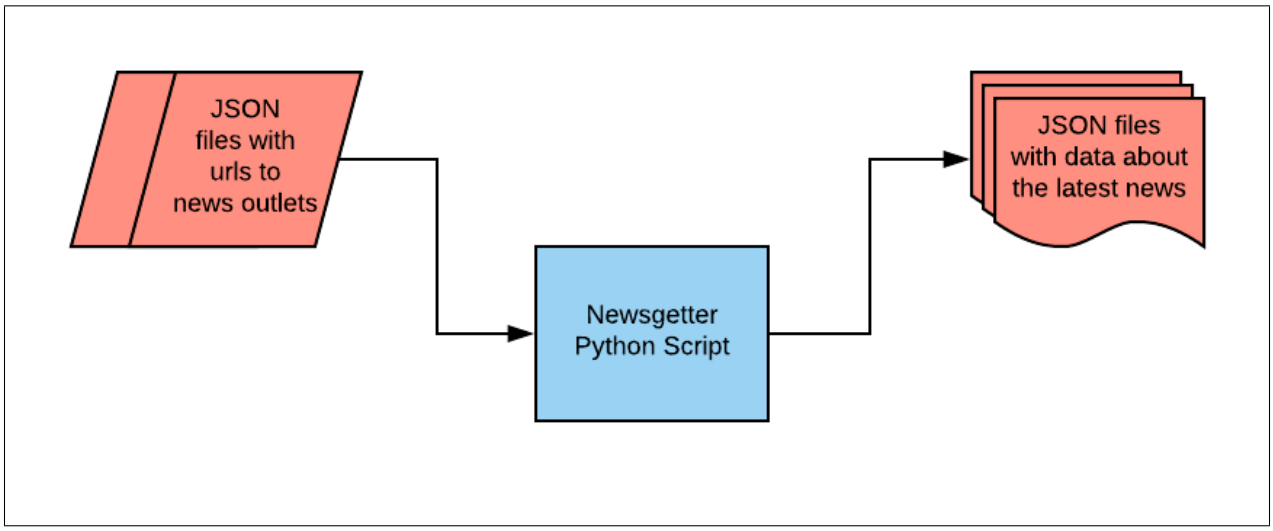

Figure 13: System overview of the Newsgetter component

This script takes about 40 minutes to run. This script should in the future run as a parallel task where different workers could go and fetch the news at the same time. Now the script gets articles from URL 1, before moving on to URL 2 and so forth.

6.2 Send to semantic annotation

After the news articles has been downloaded, we can send them to a C# .NET application for analysis and semantic annotations, which can be visualized in figure 14. This is done through the use of a REST API. The Python script goes through each of the folders containing JSON files and sends them as a JSON object to a handler in the C# code, where the data is semantically annotated with the use of a queue.

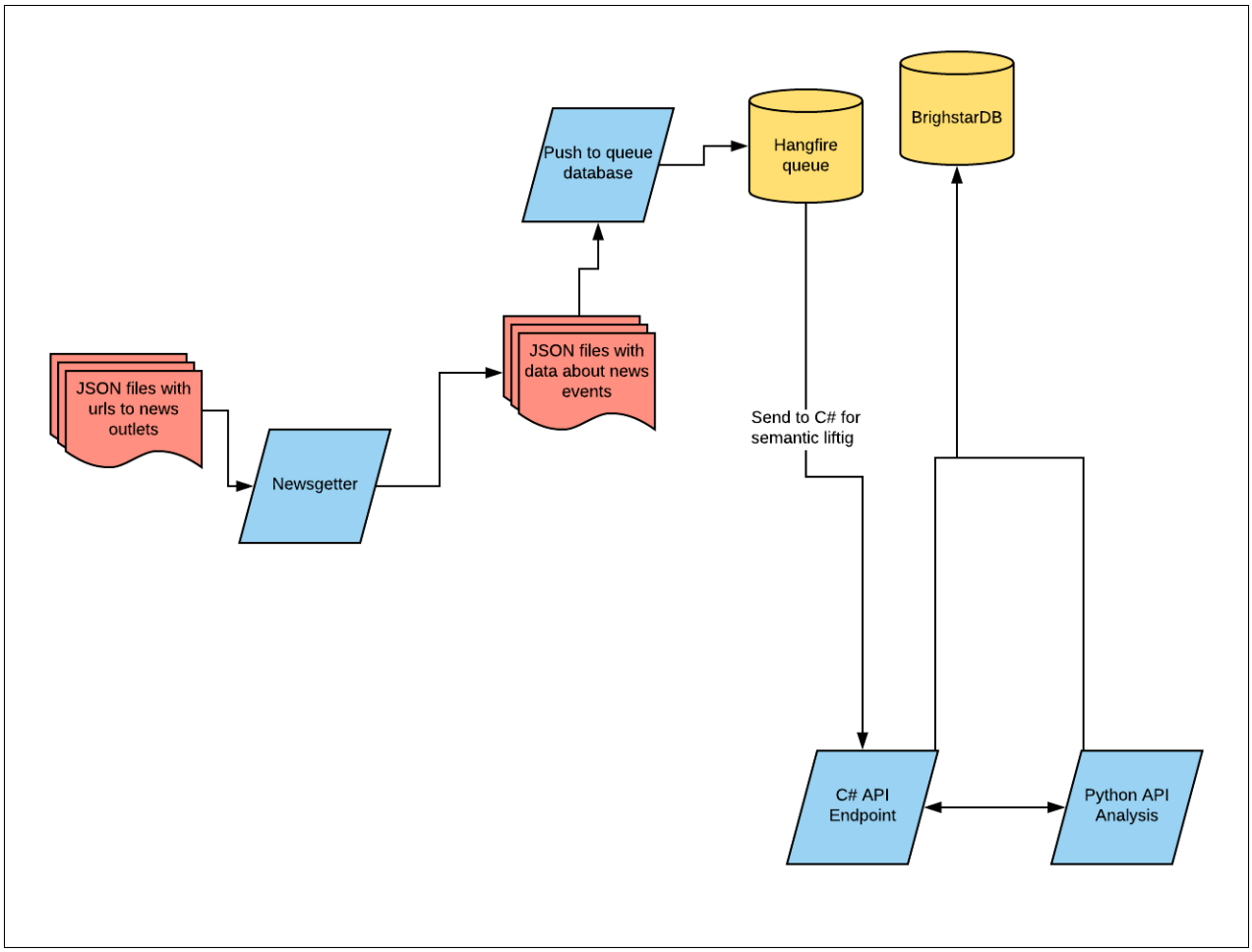

Figure 14: Data being sent to semantic annotation via a queue database

This component is the most brittle in the sense that it breaks at random times, by returning a outOfMemory-exception from the C# code. This should be one of the components that would benefit most from a rewrite in a future version of News Hunter.

6.3 Push to Elasticsearch

Meanwhile, when the newsgetter component has downloaded all files, another Python script can be used to send the data to the Elasticsearch database. This process is shown in figure 15. This step can be included as a standalone component that the developer/user will have to manually start, or it can be called after the newsgetter script has downloaded all files. Either way, in the future it should all be an automatic process running perhaps

every other to retrieve the most up to date news articles.

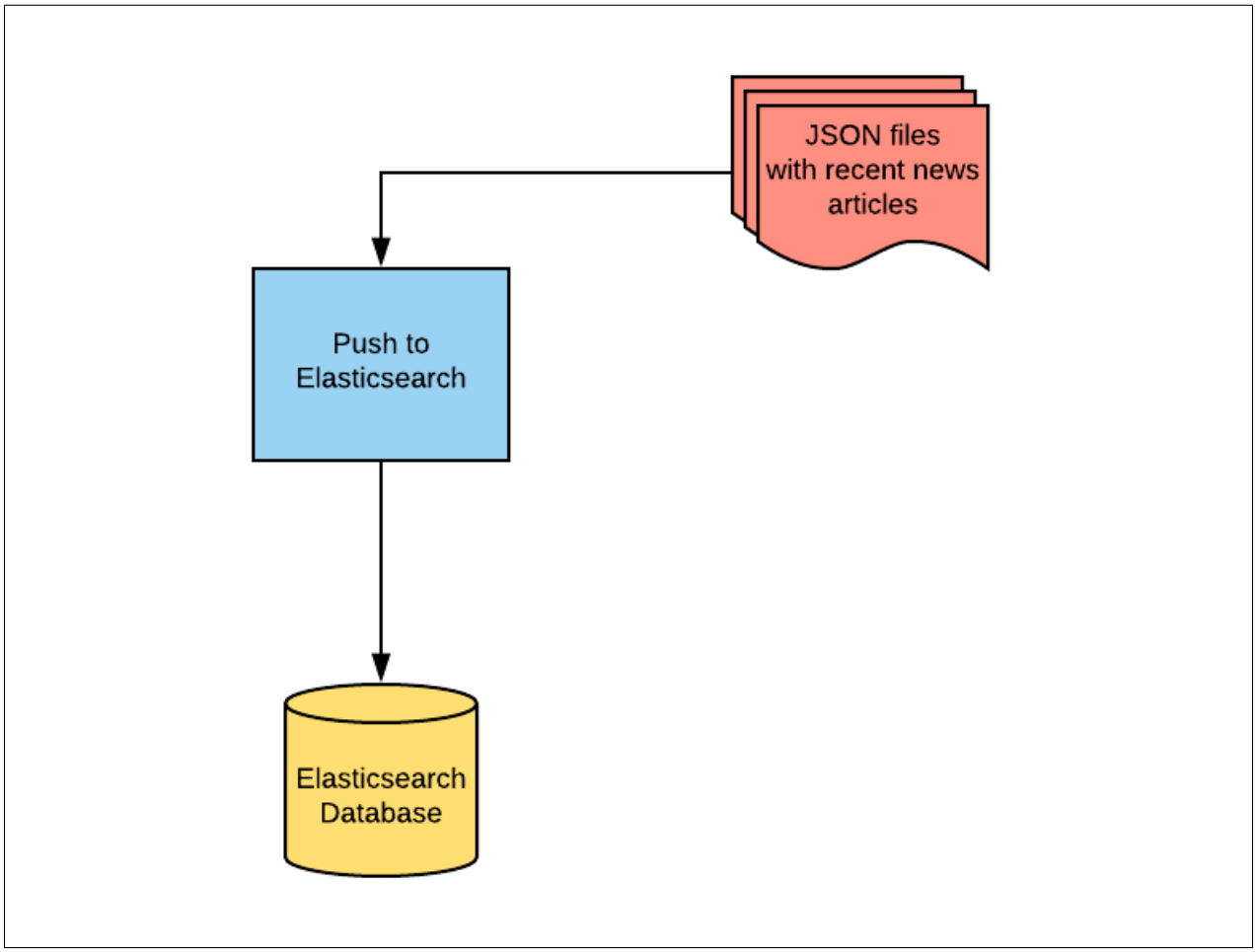

Figure 15: The Newsgetter component pushes JSON to Elasticsearch

This script also takes care of duplicate data in the Elasticsearch database, so if you run it twice with the same data, it will output errors in the log telling the user that the data is already present in the database.

6.4 Analyzer components

When the data has been semantically lifted and annotated through the C# pipeline, it can be sent back to Python as seen in figure 16, for analysis of sentiment, categorization, entity extraction and keyword extraction. The analyzer components use different text

and machine learning libraries for Python such as Scikit-learn[24], AFINN[25] and Textacy[26], among others.

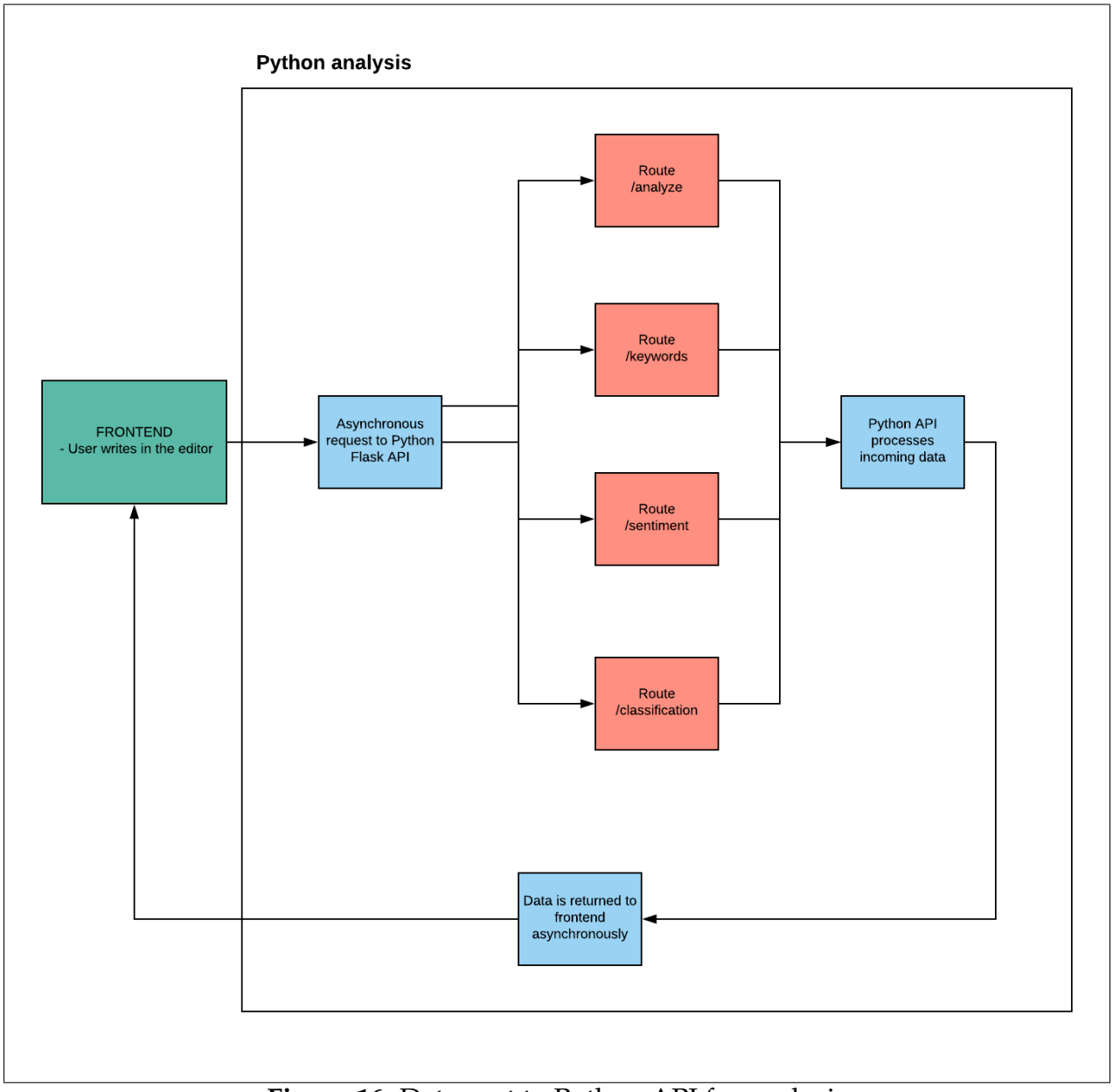

Figure 16: Data sent to Python API for analysis

[24] http://scikit-learn.org/stable/
[25] https://github.com/fnielsen/afinn
[26] https://github.com/chartbeat-labs/textacy

6.5 Twitter feed

The Twitter-feed component is a simple API endpoint that connects the Python code to the official Twitter API, and lets the frontend code request the N most recent tweets from a Twitter user. This component can be seen in figure 17. An improvement here would be to store the tweets in a database or semantically lift them and connect the data to new entities, much in the same way as the editor works. The difficulties with such approach is that tweets often use slang or abbreviations, and it may not be as trustworthy as the user would like.

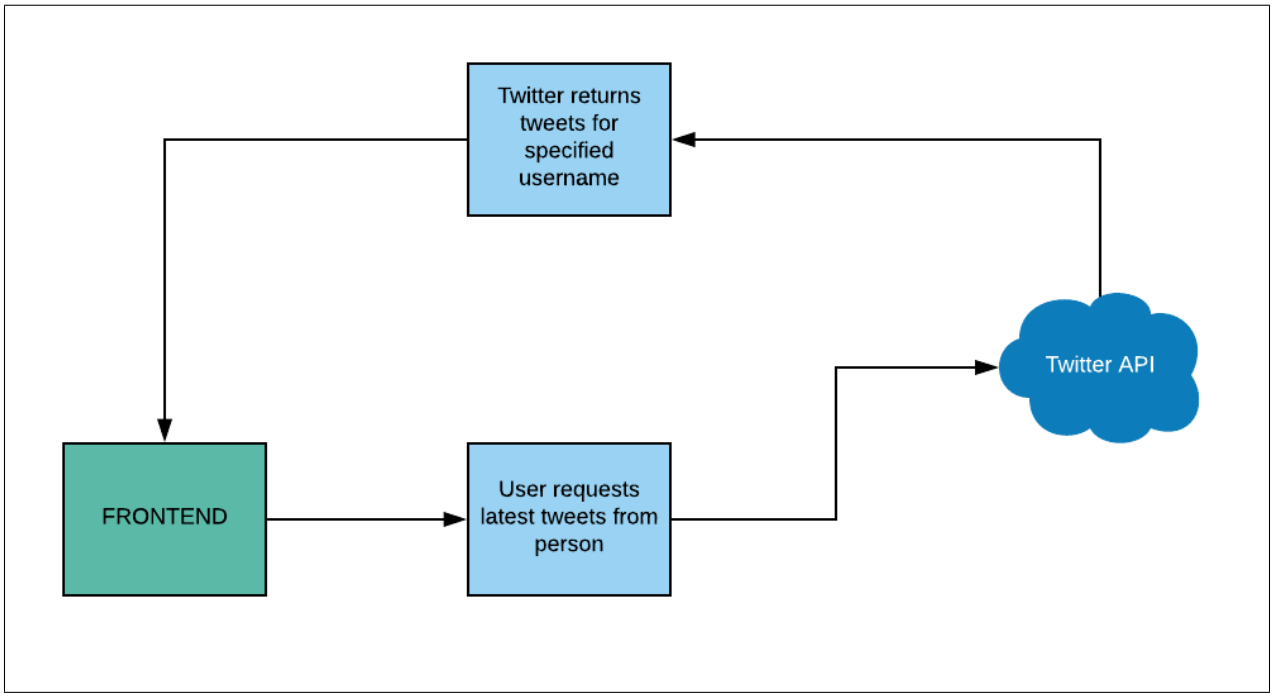

Figure 17: User requests tweets

6.6 Web API

All the underlying data stored in the BrighstarDB semantic graph is accessible via a REST API endpoint. It can be extended by developers to add access to more data in whatever way suits the needs of the next development phase of News Hunter. The web API serves responses in JSON format.

6.7 Frontend

The frontend of the system, shown in figure 18, is used to display data to the user, and let the user interact with and discover the possibilities of semantic technologies in a news desk workflow environment. During this development phase, most of the views in the frontend have been refactored into a more flexible design, meaning it can be viewed on different screen sizes. This was a choice done because users expects the application to work on both mobile and desktop environments.

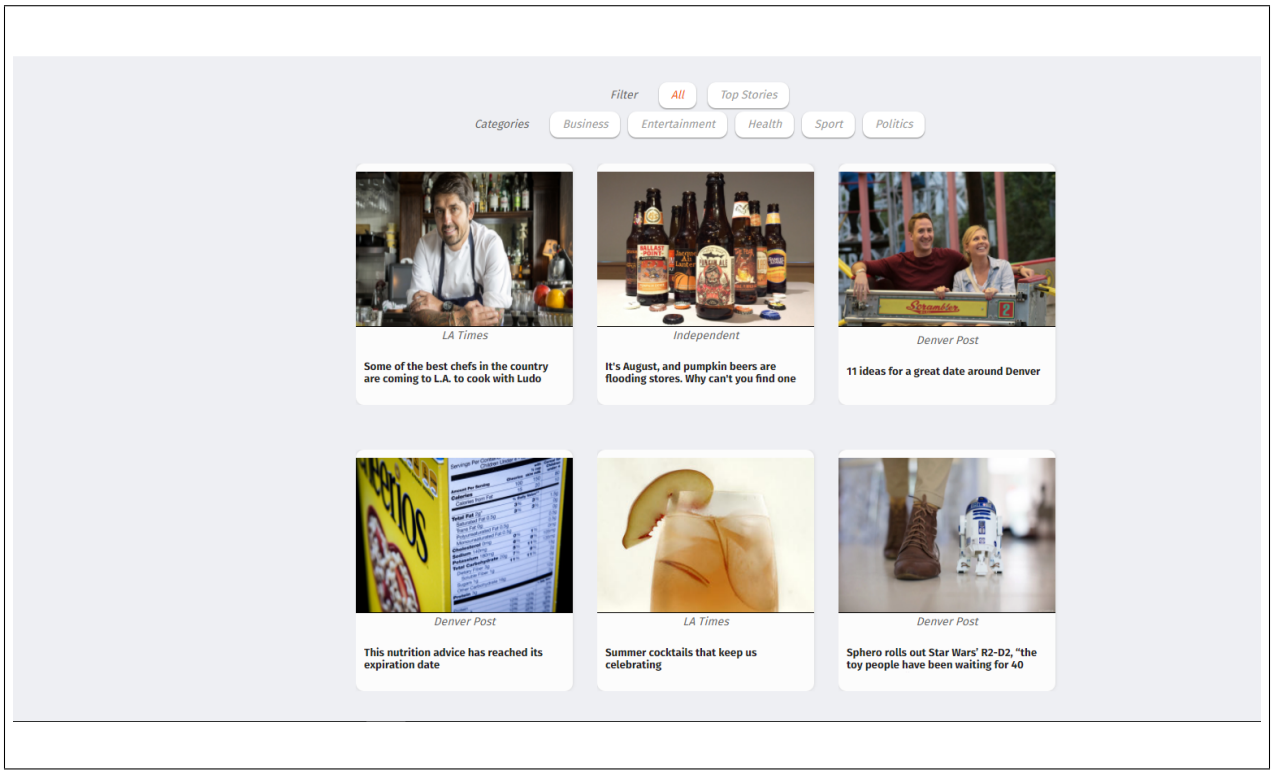

Figure 18: Front page view developed by Christensen and Villanger (2017)

6.8 Future Improvements

No system is without any bugs, and News Hunter is no exception. The most vital bug is an outOfMemory-error which occurs when trying to feed the system too many JSON documents at the same time for semantic lifting and annotation. This slows down development speed and is fairly crucial if you want a system that has up-to-date news from all

sources. So the first step the next developers should do is to take a closer look at what is causing the error and hopefully fix it. There are also some minor tasks that should be done, such as making sure the application scales well across all views in the frontend, while using the tool on both mobile or desktop.

7 Evaluation

One of the key processes in Design Science Research is to use rigorous evaluation methods. This section talks about what evaluation was conducted, how it was conducted and the results form the evaluation.

7.1 Evaluation Objectives

The objectives for the evaluations were to get an overview of what the end users thought about the system and to get feedback on future improvements. The results are presented qualitatively by having discussions in focus groups. The evaluation has been done in two different settings. One for already working journalists and graphic reporters from TV 2 Norway, while the other focus groups consisted of journalism students from the University of Bergen.

7.2 Evaluations from users

The users evaluating this iteration of News Hunter were graphic reporters working with live graphics on TV productions, journalists working for TV 2 Norway and journalism students from the University of Bergen. One of the participants in the evaluation session was also a participant in the survey conducted before development started. There were two evaluation sessions. One with employees of TV 2 and one for the journalism students. The students were a mix of those studying regular journalism, as well as those studying investigating journalism.

The evaluations were conducted by demonstration, discussion, hands-on reviews, questions and observations.

7.2.1 Focus Group for TV 2 Norway

In this focus group, there was a total of five participants evaluating this new and enhanced version of News Hunter, three graphic reporters and two journalists from TV 2 Norway. The graphic reporters has worked for TV 2 Norway for at least 2 years and can be seen as experts in their field and day to day tasks.

Introduction to users
For the evaluation session, users were first given a quick demonstration of the whole system. Because there were new users evaluating and testing the system as opposed to when Christensen and Villanger did their evaluation in 2017, the users were briefed about what the evaluation session was about, the expectations to each evaluator and why this evaluation was important for the end product.

After the demonstration, each user was given a set of tasks, where each task had a list of steps to follow. After following the steps and completing the tasks, the users were asked a set of questions. The questions had the same structure as the questions found in TAM and TAM2. This is also the same type of questions used in Christensen and Villanger's evaluation (2017).

7.2.2 Questions

The questions were presented to the focus group as a whole, and for each question followed feedback and discussion about what they felt about the feature, if it was useful, if they felt something was missing and what features the users would see implemented in their current workflow. All questions from the survey can also be found in Appendix A.

7.3 Results from Focus Group

Since both focus groups was asked more or less the same questions, I have only added the thoughts of the journalism students where the discussion differed from the focus group consisting of employees of TV 2.

7.3.1 TV 2

Questions

Overview of Politicians
The overview of politicians gives the graphic reporters and journalists an easy way to discover and find out who is who in politics. This is often done when broadcasting live sessions from the Parliament, or interviewing a number of politicians at the same time.

The feedback from the users were positive. Participant 1 said that this feature would be useful for researching what title to give to an interview subject during a live interview session. Participant 2 thought it was useful to have pictures for each politicians.

Participant 3 responded that it was missing what role or title the politicians had. E.g, Erna Solberg, the Norwegian prime minister was only listed as an representative for Høyre (Norwegian right wing). Participant 1 also noted that it should be more sources than just Wikipedia links for the politicians. It was requested that a more credible source was also listed.

All participants responded that the feature was useful, but would like to see it more elaborated in the future.

Search for athletes
The search for athletes lets a user search for a sports athlete in DBpedia[27], as well as viewing the latest news articles where the athlete has been mentioned.

Participant 1 thought it was useful to see what news articles the athlete had been mentioned in lately. Participants 2 and 3 liked the fact that you got an overview of different types of information instead of having to do multiple searches on the web for the same athlete. Participant 4 responded that the featured lacked functionality for him to able to use it.

Participants 2, 4 and 5 agreed that if this feature were to be specifically useful for them, then the system had to be more integrated to their graphics tools for broadcasting. Par-

[27] http://wiki.dbpedia.org/

ticipant 1 and 3 would like to have more sources present, e.g Premier League statistics[28], Opta Sports data[29]. They all wanted more flexibility on how to generate graphics, and they would like to be able to choose what data was exported to the tools they use for the live graphics during broadcasting.

It was consensus among the participants that the idea of the feature was useful, but that it lacked too much data to be very useful for their daily operations.

Writing in the Editor
The editor view lets the user write up their story for publishing, while having automatic analysis of entities and keywords, sentiment and categorization. The editor view also lets you browse the latest stories from other news outlets while writing your story.

The participants thought this feature would make their day writing articles easier. Participant 3 found the automatic extraction of keywords and categorizations to be extra useful, as she often found herself writing down keywords after the text had been written, and she mentioned that it felt like doing a job twice.

ticipant 2 and 5 gave feedback that they missed dates for when the articles were published from the other news outlets. They would also, in agreement with participant 1 like to also see what other stories their colleagues were working on. They thought that it would make their jobs easier by knowing who to collaborate with when working on the same stories.

All participants found this feature useful, but would see more functionality in the future.

Get latest news about a person/Interview booking form
The feature for getting the latest news about a person and book them for an interview is a tool for interview booking and to get a quick overview over what the other news outlets have been writing about the interview subject before an interview takes place.

Participants 4 and 5 thought this feature was useful when they needed to do an interview, but weren't fully up to date on what the other news outlets had written. Participant 2 thought it was useful as an easy way to make a booking in the system.

[28] `https://www.premierleague.com/`
[29] `https://www.optasports.com/`

Participant 1 and 3 liked the idea, but they would also like to see when the person was last booked and to see if anyone else had already made a booking for the interview subject. They would also like to see if there would be any comments for an interview subject, such as questions the interviewer should not be asking.

The participants found the feature to be useful, but some of the participants thought it would be better suited for booking management employees and planners, than for graphic reporters and journalists

Graph of who knows who

The graph of who knows who is meant to be a tool for visualizing a person's connected network. It is meant as an assisting tool for journalists that need to do an interview, but want to make sure they are not talking badly about someone close to the interview subject.

The participants all agreed that this feature was a good idea, and that it was useful to easily see who knew who in the graph. They liked the visualization, but thought it could need some more refinements before being really useful in their workflow.

Participant 1 stated that she would like to also have access to the employees of TV 2, not just interview objects. Participant 3 added that this would be really useful to see what type of contact network a journalist had, if you needed a phone number or information about certain people. A lot of journalists working in a specialized field of expertise creates a wide network that they use when investigating and writing about a case.

Participant 2 would also like to see more information about each person, than just who knows who. It was requested that information such as emails and phone numbers should also be present. As of writing this, the journalists would often send out emails in form of *does anyone have the phone number to xxx*, but they thought they could get rid of those annoying emails if this feature had the phone numbers present.

The participants found this feature useful, but not finished.

7.3.2 Focus Group of Journalism Students

The evaluation for the journalism students were very similar to the one done with employees of TV 2. The difference was that we focused more on what features they would like to see in a future version, than on just evaluating the features that had been implemented. I found during this evaluation that a lot of the feedback on the current features as very similar to the one from TV 2, and I will therefore focus on the differences between the two groups.

Because the employees of TV 2 has a better understanding of how a journalist works and what tools they use on a daily basis, the major difference between them and the students were that while the employees focused on how they could write their stories faster and more accurately, the students were more concerned by what data News Hunter would be able to pull down and analyze. The students had a lot of good ideas for further development, such as adding dictation to the editor, so that the journalist can speak their story, instead of typing. They would also like to see a better connection to social media, and they came up with the idea of monitoring hashtags from Twitter to see what people were talking about. This feature can already be delivered by ScribbleLive[30] and by hooking up to their API, this functionality could add more information to News Hunter.

7.4 New information regarding TAM2

The thesis has used TAM2 as a guideline throughout development, which can be seen in subsection 3.3.2. After evaluation with end users, new information has become available in regards to the perceived usefulness of the application. The noticeable points for TAM2 and perceived usefulness (PU) are; subjective norm, image, job relevance, output quality and result demonstrability.

After evaluation, the subjective norm shows that the intention of the user is that they think that the more useful and easy to use News Hunter could be a good addition to their workflow. The users also agreed that if a few of them would use the system, they anticipated that more users would use News Hunter in the future because they would create

[30] `https://www.scribblelive.com/products/`

an image that News Hunter is useful for them and users that does not use News Hunter would see that and start to use it. Because this version of News Hunter does not support other languages than English, the job relevance for now is not good enough. Since the evaluation has been conducted on norwegian speaking employees and journalism students, they have reported that they liked the features, but they would not be able to use them as they write in Norwegian. The output quality has not been tested well enough in this thesis, but feedback from interview subjects suggests that some of the daily tasks could be done more efficient by using this more useful version of News Hunter. The result demonstrability has only been demonstrated in a closed focus group setting. Venkatesh and Davis (2000) says that result demonstrability is seen when users can see that the application is set to production and the users sees results immediately. For this evaluation, the users got a demonstration of what the system could do, but it is not enough to conclude that they would use the features in a working environment. Future work should focus on trying out the system in a more work simulated environment.

7.5 Evaluation Summary

From the first survey conducted before development started, to getting responses from end users in the focus group, it is noticeable to see how the planned features changed during the iterations. There are not too many direct responses from the first survey, to the feedback received in the focus group, but taken together, they given an impression that it is still possible to see what the end users wants in this type of system.

The most noticeable link between the survey and the focus group is that both groups of respondents would like a better system for knowledge management. 75% answered in the first survey that they would like a better system for storage of knowledge, and if we see that in connection with the graph visualization and the responses from that feature, we can draw the conclusion that the end users wants a better system. Right now the users has the ability to save a new person to the graph, but a new phase in the development of News Hunter might be able to add more data about persons to store and better functionality to reuse the existing data in the knowledge graph in new and better ways.

Another important aspect is that respondents from the survey sometimes felt that how they worked now might lead to stupid mistakes being broadcasted. The participants in

the focus group said they got a better overview over the data they typically worked with in this version of News Hunter. They also mentioned how they liked that they could see what other news outlets were working on. This last point was both important when they were working on a story themselves, but also if they had been on vacation or as some of the participants who are freelancers noted, it helps to get an overview over what has happened in regards to for example sports athletes while they had been away.

7.6 Evaluation of Own Work

This sections explains and describes how I have tried to follow best practices when working on this project. I will talk about how I proceeded at the start of the project, how I followed through when refactoring and cleaning up the code and how I have documented the project in a more suitable way of delivering it over to the next persons maintaining the code base.

7.6.1 Documenting How Everything Worked

I have previously said that when I took over the project, it was barely documented, and it clearly showed that time was tight at the end of the deadline for Christensen and Villanger. There were some functionality not being used in the frontend, which led to confusion at the initial start of the project. I therefore started the project by removing some of the functionality that did not work as expected, to be able to focus on the parts that did work.

7.6.2 Clean Code

During the iterations of the project, I have tried my best to follow Robert C. Martins advices in his book *Clean Code - A Handbook of Agile Software Craftmanship*. In his book, C. Martin writes very well about how to keep code clean and not make a mess. A mess will cost in the long term, as every new feature breaks some other parts of the code, meaning that the developer(s) will be working at a snails pace (2009). For this project, clean code

meant that I had to refactor and extract bigger functions into their own functions to make the code more readable. C. Martin says that functions should be small, and then they should be smaller than that (2009). I think this is a very good point, and I have tried to follow this practice as best as possible. Small functions makes everything more readable and easier to comprehend.

7.6.3 Comments

When I went through the code for the first time in this project, I could only find one or two comments, and those were also outdated and wrong as the code had changed since the code had been written. Misleading comments happen, but as C. Martin writes, they might make another programmer working on the project in the future find himself in a debugging session because he called a function with the best intentions based on the comments, while the comment was in fact misleading (2009). I have gone through the code and commented where I have seen it fit. I have not documented every line of code, as I am a firm believer that the code should speak for itself, however I have commented those parts of the code that at this point have some behavior that might break a feature if removed or called wrongly.

7.6.4 Formatting

Robert C. Martin writes in his book that formatting is important for how another programmer reads someone else's code. It needs to be properly formatted both vertically and horizontally (2009). It should also be intended properly, and for that purpose I have used Standard.js[31] for the frontend code to adhere to a standard. I think that this way the next developer(s) that work on this project can keep adhering to the same standard and not have to think about how it should be formatted.

[31] `https://standardjs.com/`

7.6.5 Error Handling

The frontend part of the code in this project did not have very much error handling. I assume this has to do with the tight deadline they had at the end, but I have tried to incorporate a lot more error handling to keep error messages easier to debug when something went wrong. This has helped along the way when further development of the project has lead to unwanted behavior and bugs. Being able to easily identify and solve bugs are key to better productivity.

7.7 Adherence to Design Guidelines

The adherence to design guidelines has been done with both management and system developers in mind. The development of this more useful News Hunter should appeal to both management and system developers. Management because they can see what is possible in a news environment, and for system developers as they will see that a great effort has been invested in more readable code and better prerequisites for further development.

I will now inform the reader about how I used the design guidelines from section 3.

7.7.1 Guideline 1: Design as an artifact

The first guideline was to design an artifact. This was done by doing the software development in iterations. For each iteration, a subset of the features Wolftech wanted was chosen, and then given a score based on criteria for how hard I assumed it would be to implement, how useful the feature would be and how long time it would take. I usually then started with the feature with the lowest score, as I assumed that would be the easiest to get done. I chose to do it this way as I at least would have some more to add to the minimum viable product this way. This worked very well for the development iterations.

7.7.2 Guideline 2: Problem relevance

The problem relevance for News Hunter is that news only travel faster and faster, and to keep up with writing new stories in an editorial workplace, the user needs to have tools that can aid in writing of said stories. Therefore this third iteration of News Hunter focused on adding more useful features to the problems the journalists often has. The new features includes as said earlier, automatic analysis of text when writing stories, quick lookup of politicians, and visualization of who knows who, and more.

The new features has not been chosen at random, but they are a selection done by the author of the thesis, as well as Wolftech. Wolftech, being the main contributor to this project and where the project originated from, their input in the problem relevance field is of great value.

7.7.3 Guideline 3: Design evaluation

I held two focus groups for potential end users of News Hunter. In hindsight, these focus groups could be done better by including a larger set of participants, as well as having them try out the system in more realistic settings. I was planning on doing a contextual inquiry, but did not as I thought it might focus too much on the user experience, instead of the features I had implemented (not focusing too much on UX).

7.7.4 Guideline 4: Research contributions

I have contributed to research by making this thesis public, and written about wether and why end users found the new features useful. I have also contributed by being able to implement an automatic analysis tool for writing news stories, giving the users feedback on sentiment of text, keywords, entities and categorizations.

7.7.5 Guideline 5: Research rigor

The research has been done by iterative and agile methods. The research has been documented throughout the process by using a Trello board to keep track of features and progress, Bitbucket and Git to keep track of code changes, and Deadline, a Trello plugin, to keep track of deadlines.

7.7.6 Guideline 6: Design as a search process

The design as a search process has been followed by communicating with Wolftech throughout the development. They have been a very good asset and they know what their customers want and why they want it.

I have also got feedback from journalists and graphic reporters from TV 2 Norway during development.

Looking back, I must admit that this guideline could be followed even better if I was stationed at Wolftech during development. Since Wolftech was in the process of moving offices during development, I did not sit at their place. This would have provided even better communication between me and them as stakeholders and I could have asked more questions during development.

7.7.7 Guideline 7: Communication of research

The communication of research has been followed by keeping the style of writing not too technical, while still keeping the technical aspects of this project. I have made sure that both technical people such as developers and non-technical people such as management and users can understand the main goal of this new version of News Hunter. The progress of the research has also been communicated to Wolftech through continuous communication.

8 Discussion

8.1 Research Question 1: How to improve usefulness and ease of use in tools for journalists?

This thesis has shown how we can improve usefulness and ease of use for an existing prototype tool for journalists. The thesis has also shown how ease of use can be improved by building upon previous knowledge in the semantic application domain. The findings in this thesis shows that usefulness is not only what the end users see and how useful they find the application. Equally important in this thesis is the focus on continuous ease of use from a developers viewpoint. Cleaner and more comprehensible code will make new developers working on the project more efficient earlier on, and by providing a better documentation of the system, both management and developers can understand what News Hunters goals and motivations are.

Before the work on this thesis was started, News Hunters state of the art was a fully functioning backend, consisting of two components, one written in C# and one written in Python. There was also frontend which lets the user navigate and explore the data. Extending the usefulness for News Hunter, I found during development that the earlier prototype did not have an automatic analysis of stories written by journalists. This was also noted by Christensen and Villanger (2017) in their thesis, when they write

> *News Hunter's editor should have made more use of semantic web technologies. In its current state, it sends the written text to an analyzer service. The analyzer service returns keywords and named entities. These can be used to connect a story to other stories in the graph, but there are no queries to external resources in the current implementation. Providing the journalist with background information while writing should be considered in future projects.*

I found that a lot of the foundation for their future plans for News Hunter was already developed, but not implemented. During this development, News Hunter has therefore seen more use of semantic web technologies and inclusion of third party data. The third

party data is displayed for the user in the same view as the editor when a user writes their stories. I found that during evaluation, all the participants in the two focus groups appreciated the improvement in the editor view. Keeping the information gathered at one view, some of the evaluators said that it kept clutter at a minimum and it would let them focus more on writing the stories instead.

Regards to the evaluation using TAM2, the clearest findings were those related to job relevance, result demonstrability and perceived ease of use. While demonstration was being held for the two focus groups, most of the evaluators gave positive feedback with reference to job relevance. The evaluators saw how especially the editor would be useful for their main jobs, which are to write news stories. The graphic reporters which were evaluated also commented that having the ability to see which politician were who would help them during their live coverage of news from politics.

Even though the system is not yet in production, the evaluators also noted that result demonstrability were positive for them. The demonstration showed them that they would see instant results by using this type of tool when working. When the users see instant results, this also affects their perceived ease of use. I found during evaluation that the evaluators did not have too many questions in relation to how News Hunter works, and the project should also in the future have a clear vision of keeping the system as easy to use as possible.

8.2 Research Question 2: How to organize projects so that ease of use and usability can be maintained?

During development of this project, there has been used project planning tools, as well as a code repository for all the code written and refactored. The code repository used in this project has been Bitbucket, and the tools works very well as self-documentation of what has been done since the start of development. The commit history from Christensen and Villanger (2017) was unfortunately not kept, but this thesis will recommend that such history is kept for future developers and contributors to the project.

Trello has been used for project planning during development. Trello uses the notion of *boards* which contains lists, and the board used for this iteration of News Hunter should

also be passed along the new contributors in the future.

During the development phase, I have seen that it is important to keep the source code clean and adhere to a standard for code formatting. This will also make the project more tidy and new developers will see that the previous developers has taken pride in keeping their code clean and well formatted. Today there is a lot of different code linters, which can format the code for you. This thesis will therefore argue clearly that not having a standard for code formatting can only affect the project in a negative way, and that there is no real excuse for not having well formatted code. Examples of code linters are ESLint, for Javascript[32], Standard.js, also for Javascripthttps://standardjs.com/ or FxCop for C#[33].

I have also found throughout this project, that having a distinct goal to work against is very helpful, and this goal should be very clear during development. Without such a clear goal, developers can drift of their path and lose sight of the important parts of the project. This clear goal, along with good communication with stakeholders (Wolftech) and documentation will help contributors and developers maintain ease of use and usability.

8.3 Critique of Methods

One of the problem with this research is the number of evaluators. Ideally, there should be even more participants in the two focus groups. Evaluation should also be carried out in a real work environment, not just as a demo. Evaluators should try to write stories using the editor, as well as the other new features of News Hunter, while being monitored and questioned afterwards. I have also found that in the future, the next development phase of News Hunter should include a "customer" while developing, as this would adhere more to a lean and agile approach, where instant feedback drives the development cycle per iteration. It is not good practice to only assume what the customer wants, during a half year period.

[32] `https://eslint.org/`
[33] `https://msdn.microsoft.com/en-us/library/bb429476.aspx`

9 Conclusion and Future Work

9.1 Conclusion

In this thesis, the main goal and motivation was to research how a prototype tool for journalists can be made more useful and easier to use. The thesis have presented a lot of new functionality for the News Hunter web application. I have refactored much of the code and written a more thorough documentation of the whole system. The new functionality includes an updated real time WYSIWYG-editor, an overview over politicians, an overview over the last news generated by other news agencies, booking of interview objects and a graph visualization of who knows who. This thesis has been influenced a lot by the previous work done by Christensen and Villanger (2017). Their thesis was a collaboration between the University of Bergen and Wolftech. This thesis follows the same path, and the collaboration has continued throughout this development phase. The development has followed an iterative approach, focusing on delivering the minimal viable product for each iteration, in this thesis described as one or two new features per iteration. The goal of the thesis was to continue to add functionality for the News Hunter application, and to make it more useful and easy to use. This has been done by refactoring a lot of the code, and written more documentation. The refactoring of the code has made the code easier to understand and reason about. From a user perspective, more of the application is now responsive, and can be used on different screen sizes. More features has also made the application more useful, not only for journalists but also for graphic reporters working with live television productions in a fast-paced environment.

The technology stack is the same as used in the previous version of News Hunter. The main backend is written in C# and the ASP.NET framework. For semantic annotation, BrightstarDB's C# library has been used. Natural language processing, categorization and sentiment analysis has been done by different Python libraries developed by Christensen and Villanger (2017). The data is accessible for the application through different REST API-endpoints through the framework Flask for Python. Python is also used to fetch news articles and save them as JSON-documents. An addition to the previous version of News Hunter is the ability to save the downloaded JSON-documents to a database instance of Elasticsearch. This gives the application and its developers an easy interface

for querying a corpus of text with different search criteria in the future. The frontend of the application has been continued to be written in AngularJS, but with the addition of the CSS-framework, Bootstrap[34] for easier development when making the application responsive.

The main activities for this thesis has been further development and enhancing of an already existing prototype of News Hunter. Most of the time has been used to refactor code, document code and implement new features. The new features has been chosen by Wolftech and the developer based on what Wolftech wanted to have implemented, and based on the skill level of the developer. The projects source code now offers code comment where necessary, rewrites of source code where refactoring were necessary and readme documentation for each code repository.

Evaluation has been conducted on potential users working for TV 2 Norway as journalists and graphic reporters, as well as journalism students from the University of Bergen. The evaluation has been in the form of focus groups, where the users has been given a demonstration of the system, and a discussion has followed for each feature. Most of the features were appreciated by the participants in the focus groups, but more work could be done to refine their use cases.

The main findings after evaluation was that News Hunter as a tool was something the evaluators found interesting. This is backed by findings done by Christensen and Villanger in their thesis as well (2017). The evaluators came with solid feedback on what features they thought would be of most value to them, and the findings show that what feature is the most important very much comes down to what responsibility and work tasks you have in the company. A journalist liked the editor, more than the graphic reporters, which liked the automatic aggregation of politician and athlete profile data.

The greatest challenges during development was to be familiar with the previous codebase. With little to no documentation beforehand, it became challenging to be productive early in the stage of development. Another challenge was that there were some bugs in the previous codebase that was not very easy to locate. Some of these bugs are yet to be found, but they are noted in the readme files of their respective code repositories. It should also be noted that new developers on this project should be co-located better with

[34] `https://getbootstrap.com/`

Wolftech. It will be easier to have better communication throughout the project when co-located, and questions can be asked and answered continuously, instead of by email.

During development, the most valuable experiences and insights found in this project has been that good communication, with both customers and previous developers are very valuable and makes the development experience more pleasant. Having a good handover in form of documentation lets the new developer(s) become more productive earlier in the development phase, and by communicating more often with the customers, it will be easier to be agile and change direction earlier.

9.2 Future Work

This development phase of News Hunter has seen more documentation, refactoring of existing codebase and added functionality. Below follows some thoughts about what future work should be considered done, on a day to day basis, as well as what Wolftech should think about doing, and lastly what I think are some of the paths this project can take in the future.

9.2.1 Development

For future work in regards to day to day development, the developers of News Hunter should refine features such as the graph visualization. This was very much a prototype and as it is right not, not meant to be used in production.

Another aspect should be to introduce testing in the project. It is hard for new developers to contribute when every change may be breaking a feature in the code. There are a lot of different testing frameworks that can be used, such as Jest[35] or Mocha.js[36]. The codebase and project would benefit greatly if tests would be introduced. As of now, the codebase has no tests. If automated tests would be introduced, the new developers would be more confident that the code they write does not break functionality.

[35] `https://facebook.github.io/jest/`
[36] `https://mochajs.org/`

Thirdly, the project should focus on implementation of better news harvesting. The scraper should be able to fetch new and updated news articles as soon as they are available. As of right now, the script for retrieving news articles has to run manually, and if all categories should be retrieved the system will take about 40 minutes to finish downloading the JSON documents. Since news agencies and news desks needs to be able to quickly react to events happening in the real world, the system should not be the bottleneck in keeping up with the latest events. The developers should also look into the bugs that can arise when semantically annotating large amounts of data. The most noticeable being an outOfMemory-exception which occurs in the C# code.

Lastly, the frontend code as of right now is written in AngularJS[37] which is the first version of Angular. Angular has since its start seen a total rewrite and is now more competitive against other frontend frameworks such as Vue.js[38] and React.js[39]. I believe that this project should in the future strive to be written in a more modern frontend framework. This will lead to better maintainability as it is better documented online, and it will be more sought after as new developers would want to work with more up to date tooling.

9.2.2 Wolftech

The next phase of development of News Hunter should also strive to include the users even more throughout the development process. This could be done in several different ways, but I propose that Wolftech would notify their users of Wolftech News that they are searching for users of Wolftech News that may be interested in testing a prototype for a new application. For this thesis, I conducted a survey before starting development, and focus groups after development had ended. A better strategy for future work would be user evaluation after each iteration.

I also think that Wolftech should update new developers on what tools and frameworks they use at Wolftech at the time of development. If Wolftech has updated to a newer version of Angular for instance, then new developers should be made aware of that and possibly be given an introduction to new updated version so that the frontend could be

[37] https://angularjs.org/
[38] https://vuejs.org/
[39] https://reactjs.org/

rewritten in the future.

9.2.3 Paths for the Project

When meeting with Wolftech this autumn we discussed the ability for the system to be able to give advice as to what features of a story that makes it a good story. I believe that this would be a great step towards aiding the journalists in writing news articles that will get more clicks online and aid the journalists in writing better stories, based on historical data. This feature would be subject to train an artificial network with previous data about what stories that has rated well click-wise based on a number of different features. The features could be what the title was, what genre the story had, what topics the story covered, what images were used, when it was published, and then teach the artificial network what to look for, when a journalist writes their articles. The system should then be able to give hints as to what needs to be added to make the story stand out.

Another path would be to add let users with administrator rights add new feeds to the system. Right now, the feeds are hard coded into JSON files. Moving the feeds to a database and giving users the ability to add or remove feeds would be a nice feature to have. It would also be nice of those same users could add non-semantic feeds, but perhaps given the choice to tag them manually, and let the system annotate the system as new data enters the system.

References

Ajzen, I. and M. Fishbein (1977). Attitude-behavior relations: A theoretical analysis and review of empirical research. *Psychological bulletin 84*(5), 888.

Angeletou, S. (2014). Linked Data: new ontologies website.

Auer, S., C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives (2007). DBpedia: A Nucleus for a Web of Open Data. In K. Aberer, K.-S. Choi, N. Noy, D. Allemang, K.-I. Lee, L. Nixon, J. Golbeck, P. Mika, D. Maynard, R. Mizoguchi, G. Schreiber, and P. Cudré-Mauroux (Eds.), *The Semantic Web*, Berlin, Heidelberg, pp. 722–735. Springer Berlin Heidelberg.

Bergamaschi, S., F. Guerra, M. Orsini, C. Sartori, and M. Vincini (2007). Relevant News: a semantic news feed aggregator. In *Semantic Web Applications and Perspectives*, Volume 314, pp. 150–159. Giovanni Semeraro, Eugenio Di Sciascio, Christian Morbidoni, Heiko Stoemer.

Børdalen, G. (2017). NRK bygger infrastruktur for å finne igjen programmene.

Borsje, J., L. Levering, and F. Frasincar (2008). Hermes: A Semantic Web-based News Decision Support System. In *Proceedings of the 2008 ACM Symposium on Applied Computing*, SAC '08, New York, NY, USA, pp. 2415–2420. ACM.

Christensen, O. A. and K. J. Villanger (2017). *News Hunter: a semantic news aggregator*. Ph. D. thesis, University of Bergen.

Davis, F. D. (1989). Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. *MIS Quarterly 13*(3), 319–340.

Davis, F. D., R. P. Bagozzi, and P. R. Warshaw (1989). User Acceptance of Computer Technology: A Comparison of Two Theoretical Models. *Management Science 35*(8), 982–1003.

Fowler, M. and K. Beck (1999). *Refactoring: improving the design of existing code*. Addison-Wesley Professional.

Garrido, A. L., O. Gomez, S. Ilarri, and E. Mena (2011). NASS: news annotation semantic system. In *Tools with Artificial Intelligence (ICTAI), 2011 23rd IEEE International Conference on*, pp. 904–905. IEEE.

Group, R. W. (2014). Resource Description Framework (RDF).

Hevner, A. and S. Chatterjee (2010). Design science research in information systems. In *Design research in information systems*, pp. 9–22. Springer.

Hevner, A. R., S. T. March, J. Park, and S. Ram (2004). Design Science in information systems research. *MIS Quarterly 28*(1), 75–105.

Kobilarov, G., T. Scott, Y. Raimond, S. Oliver, C. Sizemore, M. Smethurst, C. Bizer, and R. Lee (2009). Media Meets Semantic Web – How the BBC Uses DBpedia and Linked Data to Make Connections. In L. Aroyo, P. Traverso, F. Ciravegna, P. Cimiano, T. Heath, E. Hyvönen, R. Mizoguchi, E. Oren, M. Sabou, and E. Simperl (Eds.), *The Semantic Web: Research and Applications*, Berlin, Heidelberg, pp. 723–737. Springer Berlin Heidelberg.

Krstajić, M., F. Mansmann, A. Stoffel, M. Atkinson, and D. A. Keim (2010). Processing online news streams for large-scale semantic analysis. In *2010 IEEE 26th International Conference on Data Engineering Workshops (ICDEW 2010)*, pp. 215–220.

Letouzey, J. L. and D. Whelan (2016). Introduction to the Technical Debt Concept. *Agile Alliance*, 4.

Martin, R. C. (2009). *Clean code: a handbook of agile software craftsmanship* (First ed.). Pearson Education.

Mendes, P. N., M. Jakob, A. Garcia-Silva, and C. Bizer (2011). DBpedia Spotlight: Shedding Light on the Web of Documents. In *Proceedings of the 7th International Conference on Semantic Systems*, I-Semantics '11, New York, NY, USA, pp. 1–8. ACM.

Neuman, L. W. (2013). *Social Research Methods: Pearson New International Edition: Qualitative and Quantitative Approaches* (7th editio ed.). Pearson Education.

Nielsen, F. Å. (2011). A new ANEW: Evaluation of a word list for sentiment analysis in microblogs. *arXiv preprint arXiv:1103.2903*.

Pérez, J., M. Arenas, and C. Gutierrez (2009, sep). Semantics and Complexity of SPARQL. *ACM Trans. Database Syst. 34*(3).

Simon, H. A. (1996). *The sciences of the artificial*. MIT press.

Swartz, A. (2002, jan). MusicBrainz: a semantic Web service. *IEEE Intelligent Systems 17*(1), 76–77.

Venkatesh, V. (2000). Determinants of perceived ease of use: Integrating control, intrinsic motivation, and emotion into the technology acceptance model. *Information systems research 11*(4), 342–365.

Venkatesh, V. and F. D. Davis (2000). A Theoretical Extension of the Technology Acceptance Model: Four Longitudinal Field Studies. *Management science 46*(2), 186–204.

Venkatesh, V., M. G. Morris, G. B. Davis, and F. D. Davis (2003). User acceptance of information technology: Toward a unified view. *MIS quarterly*, 425–478.

W3C, S. W. (2012). Web Ontology Language (OWL).

Wolftech (2016). Wolftech News.

A
Interview Questions for Interviews

Overview over Politicians

1. Have you missed a better overview over who is who in politics? E.g when covering press conferences, question times, interviews etc.
2. What do you think about this feature?
3. What do you like about the feature?
4. What do you not like about the feature?
5. What do you miss about this feature?
6. What improvements do you wish to see for this feature?

Search for Athletes + Twitter

1. Have you missed this feature?
2. What do you think about this feature?
3. What do you like about this feature?
4. What do you not like about this feature?
5. What do you miss about this feature?
6. What improvements do you wish to see for this feature?

Writing Stories in the Editor

1. Have you missed this feature?

2. What do you think about this feature?
3. What do you like about this feature?
4. What do you not like about this feature?
5. What do you miss about this feature?
6. What improvements do you wish to see for this feature?

Overview of Latest News About a Given Person

1. Have you missed this feature?
2. What do you think about this feature?
3. What do you like about this feature?
4. What do you not like about this feature?
5. What do you miss about this feature?
6. What improvements do you wish to see for this feature?

Visualization of Who Knows Who

1. Have you missed this feature?
2. What do you think about this feature?
3. What do you like about this feature?
4. What do you not like about this feature?
5. What do you miss about this feature?
6. What improvements do you wish to see for this feature?