News Hunter: a semantic news aggregator

Christensen, Ole Andreas

Villanger, Kjetil Jacobsen

Department of Information Science and Media Studies

at the University of Bergen, Norway

2017

Dissertation date: June 01, 2017

Abstract

This thesis presents a research project where a semantic news aggregator system called News Hunter was developed and evaluated. News Hunter was a collaboration between UiB and Wolftech that aimed at merging semantic web technologies with newsroom systems. The goal was to provide those working in newsrooms with quick and easy access to relevant background information during their research and writing process. To achieve this goal we built a system that could extract named entities and keywords from incoming text, and then store and use these to gather data from other resources. News Hunter was capable of presenting journalists with up-to-date background information on incoming news messages, as well as live-updated information when writing new stories. Journalists and domain experts tested the system, and answered questions about the usefulness of each feature. Some features were tested with F1-scoring and others by comparing it to publicly available systems with similar functionality. The accuracy of the system and the results from user evaluation give encouragement for further development of semantic newsroom systems and additional research into the field.

Acknowledgements

We would first like to thank our thesis advisor Professor Andreas Lothe Opdahl. His door was always open whenever we had questions about our research or writing.

We would also like to thank Wolftech for giving us an office space, ideas and encouragement.

Finally, we express our gratitude to our families for giving us support and encouragement throughout our years of study. This accomplishment would not have been possible without them. Thank you.

Contents

List of Figures

Chapter 1

Introduction

1.1 Introduction

The news industry is centered around information. This information can be gathered from several sources. Traditionally the different news outlets have collected information through methods such as investigative journalism, event reporting or leads from the public. In this era, where information flourish, news agencies can get other sets of inputs, that can be delivered straight to their desk. Some companies have already foreseen the opportunities, and have started storing massive amounts of data from different data streams continuously. This data could be very useful for journalists in search for their next story. With the massive quantities of data coming in, a sizable portion can be characterized as noise. Making it near impossible for human agents to filter through it. Good and viable information can be hidden in the vast amount of data. Searching through and labeling large quantities of data is a job computers are very capable at. A system that can monitor the data, and search for patterns within the noise could prove valuable.

News Hunter is a project that takes advantage of semantic technologies to collect, annotate, and analyze data streams from traditional news outlets and social media. The goal of the project is to provide journalists in newsrooms with enhanced investigation tools to give them the most relevant data available continuously through their work with news stories. The News Hunter project is a collaboration between UiB and the media company Wolftech, which have newsroom systems in development.
The goal of this thesis was to aid Wolftech by exploring tools, technologies and methods that

can be used in the creation of a semantic news aggregator system for professionals.

1.2 Research questions

RQ1 - How can semantic web technologies be used to improve news aggregator systems?

RQ2 - How can NLP and machine learning be used to connect stories in semantic news aggregator systems?

RQ3 - How can a semantic news aggregator aid journalists during research and writing?

1.3 News Hunter

As previously stated the goal of this thesis was to develop and explore solutions for tackling the problems that the News Hunter system was trying to solve. To fully explore tools, technologies and methods, a prototype was built. Parts of the prototype was influenced and inspired by previous work done for Wolftech, as described in detail in section 2.2.3.

Figure 1.1: Outline of the News Hunter system

Figure 1.1 presents an outline of the system built in this project. The system has two main input channels, news messages from external sources and news messages created internally. These inputs are analyzed continuously, and the results are stored in a semantic graph. The graph's content can be enriched with data from external resources. When journalists are browsing through the news feed or writing a new story, they can make use of the graph's data to get up-to-date background information.

Chapter 2

Theory

2.1 Semantic Web

The semantic web as introduced by Tim Berners-Lee is a way to bring structure and semantics to web pages as an extension to the World Wide Web (WWW). The goal is to make structured information available not only to humans, but also intelligent machine software agents. This will aid these agents to find, share and integrate structured information (Shadbolt et al., 2006).

2.1.1 Web Ontology Language (OWL)

OWL is a W3C Recommendation that deals with the representation of ontologies and knowledge (Group, a). Ontologies are formal representations of knowledge that are explicitly defined for some purpose. An ontology describes things or objects that are formalized and represented with properties and relations between them using a specific vocabulary (Gruber, 1993).

2.1.2 Resource Description Framework (RDF)

RDF is a W3C specification dealing with the storage and interchanging of information on the web with the use of metadata (Group, b). With the use triples, it can build statements about resources and their relationships.

A triple is an expression made up of a subject, a predicate, and an object (Shadbolt et al., 2006).

Subject Denotes the resource

Predicate Gives a relational aspect between a subject and object.

Object Target of the predicate

Eg: ”The mountain has the height of 3000m”

$Subject \rightarrow Themountain : Predicate \rightarrow height : Object \rightarrow 3000m$

2.1.3 Linked data

Linked data is essential to connect the semantic web (Berners-Lee, 2006). The term refers to a set of best practices for publishing and connecting structured data on the web (Bizer et al., 2009). Berners-Lee (2006) proposes four rules which outline how data should be published on the web to become a part of the global data space.

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

2.1.4 Graph databases & Triplestores

A triplestore, or subject-predicate-object databases are repositories for storing RDF statements that also allow for querying of data using SPARQL. Some triplestores can be built from the ground up, others can be built on top of SQL databases. Graph databases can also store resources and the relationships between them, but use a graph structure to do so (Vicknair et al., 2010).

2.1.5 SPARQL

SPARQL is a query language for the Resource Description Framework (RDF) (W3C, 2008). It is used to retrieve resources denoted as triples from a database. SPARQL can be used to express queries across diverse data sources, whether the data is stored natively or viewed as RDF through middleware (Prud’hommeaux and Seaborne, 2008).

2.2 Newsroom systems

2.2.1 Wolftech

Wolftech Broadcast Solutions is a company founded in 2011 in cooperation with TV 2 Norway. They are creating innovative solutions for broadcasters and other media companies. Their goal is to deliver news and sports content fast and cost efficiently. Their product portfolio includes technical products such as Wolftech Production and Wolftech Sports whereas Wolftech News is a system aimed at news production (Wolftech, Wolftech).

Wolftech News

Wolftech News is a product for managing an efficient news production workflow in a newsroom environment. Wolftech News assists journalists and executives with planning, news monitoring, resource management, news editing, and multi-platform publishing.

2.2.2 News Hunter

Wolftech Broadcast Solutions have previously collaborated with UiB about a system called News Hunter. The idea behind Wolftech News Hunter is to use semantic technologies to gather, organize and leverage traditional and social media streams and other big data sources for news editing. The goal is to continuously analyze incoming news messages as well as stories that journalists are writing. During analysis, the system will dissect each news message to extract meaning and patterns. The results should provide the user with relevant and prepared background information that is gathered from traditional media, news services, social media and other resources.

2.2.3 Previous Work

System

Wolftech have previously made some efforts to mark up their content semantically. The previous effort was a prototype system that was developed focused around semantically lifting Facebook posts. The previous effort was called *WT Semantic Prototype*. The first step for *WT Semantic Prototype* was to run data gathered from Facebook through Google's API for

translation if the text was not already in English. After the text was translated, it was analyzed through an online analyzer API to extract additional metadata. The online analyzer extracted data such as named entities, keywords and sentiment score. The extracted data, in addition to the included metadata in Facebook posts was used to mark up each Facebook post semantically. The semantically lifted Facebook post was then stored in a triplestore. The codebase was written in C# as an ASP.NET application with a BrightstarDB database for semantic storage of the graph. The data used in the *WT Semantic Prototype* was gathered using the Facebook API on public profiles and the analysis of text was done by the Alchemy API from IBM.

Ontology

The Wolftech News ontology was created for *WT Semantic Prototype* in order to model incoming news messages.

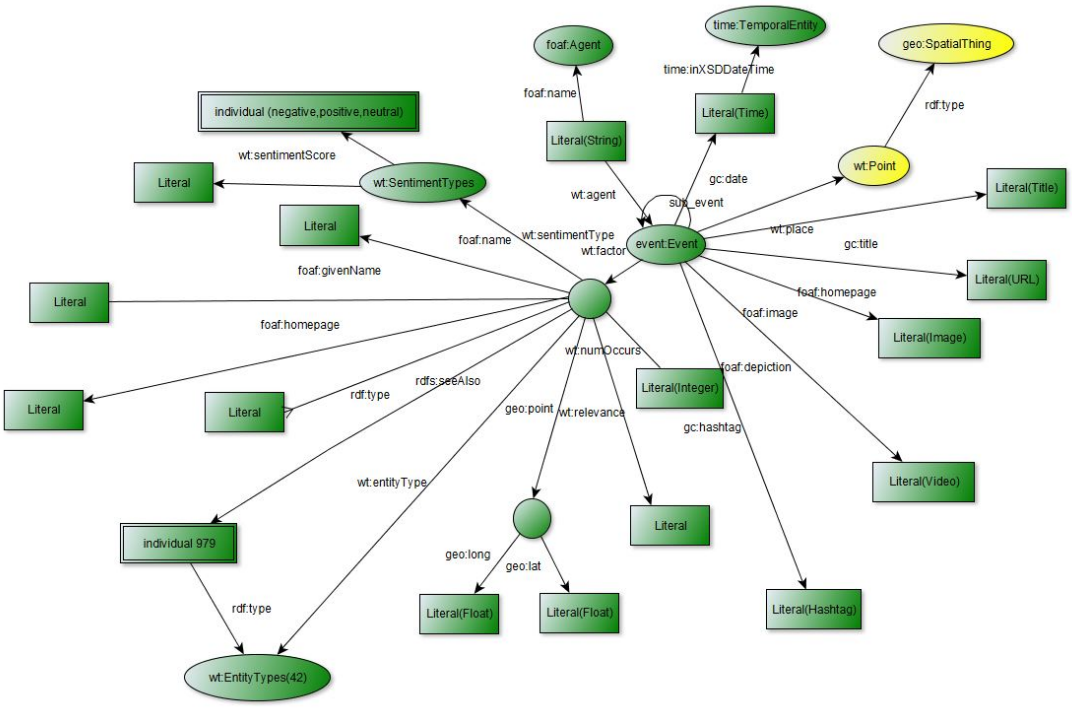

Figure 2.1: The Wolftech News ontology as used in *WT Semantic Prototype*

The ontology models news messages as recursive events with descriptive metadata as properties. Example of the metadata include, but are not limited to, date, title, and place. The model also contains nodes that describe the textual content of the news message.

2.3 Semantic News

News consumers want to receive news about events as soon as they occur. They want to get information about events of interest, and not be bothered with seemingly useless information. Fresh information and relevance is important to get the consumer's attention. Different approaches to aid in reaching this goal have been attempted by different news agencies. One of the attempts has been to insert news stories into categories, another has been to add keywords or tags to each story. These approaches have their limitation with the amount of manual labor required, as well as the time consumed in the process (Fern et al., 2006). With the advancements in natural language processing and artificial intelligence techniques much of this work can be automated.

Semantic news deals with annotation of news data to enhance the understanding of news items for intelligent agents and to enable more precise searches for news content. The metadata for each news item is automatically annotated, and can be enriched with annotated results from the techniques used in natural language processing, and artificial intelligence.

2.3.1 Related work

There have been previous projects aiming to solve similar problems.

NEWS

Getting finer grained annotation of news item contents into newsrooms was done by Fern et al. (2006) in their NEWS project. This project had the aim of using semantic technologies to annotate news articles using NLP technologies to bring high quality information to new agencies. Some of the main techniques used to achieve this was: named entity recognition, word stemming, and Part-of-Speech tagging. They created an algorithm called IdentityRank based on PageRank which could perform named entity disambiguation and link the same entities together. For news agencies to take advantage of the system, it was offered as different web services. This enabled users to cherry-pick which parts of the system they wanted to integrate. The annotated data were modeled using an ontology specific for the journalism domain. They found that NLP techniques could extract meaningful metadata from news items.

BBC

The BBC applied linked data technologies to their BBC programmes and BBC Music to link together resources from different systems. Before this technology was used all their resources were integrated into microsites that were unable to cross-communicate information about resources such as artists, songs, TV shows etc. Named entity recognition was used in conjunction with named entity disambiguation to identify the resources in DBpedia and use that as a link (Kobilarov et al., 2009).

News aggregators

Krstajić et al. (2010) used semantic analysis of news articles from European Media Monitor (EMM) as a real-time system. EMM is a news aggregator system that applies clustering and named entity recognition on multi-lingual corpuses. It provides information about the recognized named entities. Other arguably more open news aggregator systems such as Google News and Yahoo News are used by a large audience. These publicly accessible aggregators show the latest breaking news sorted by the number of sources reporting about a specific event, and the number of articles surrounding the event. Another approach is to try and visualize the events, this is an approach Newsmap have attempted, where they use the aggregated news from Google and visually encode in into a TreeMap visualization. These news aggregators are built around dealing with the latest news. This can be a drawback if you want to do temporal analysis on news stories (Krstajić et al., 2010).

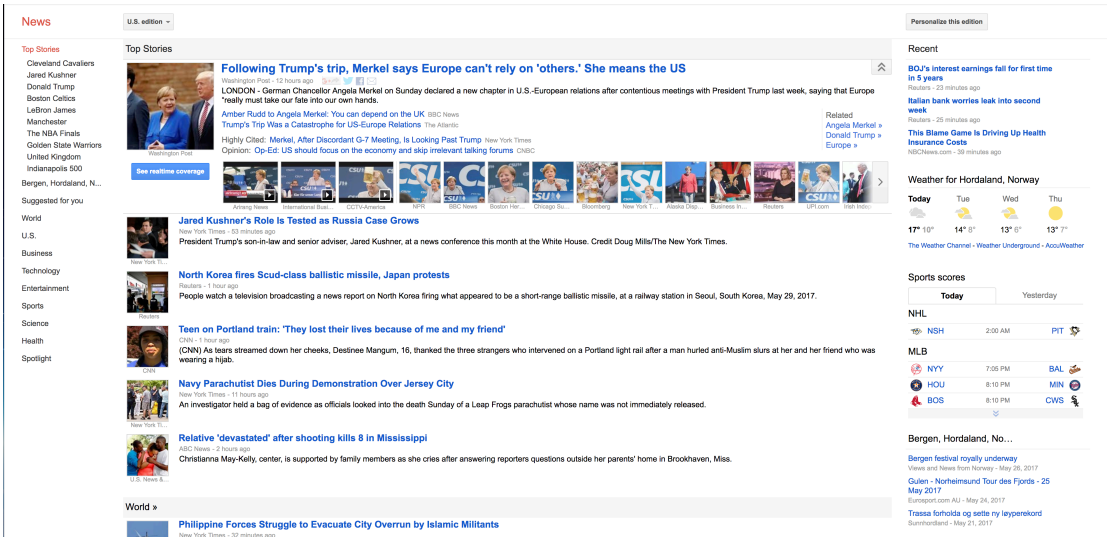

Figure 2.2: Screenshot of Google News

The screenshot above displays a news story within Google's news aggregator system, Google Trends. It contains multiple graphics displaying various information about the story and its popularity.

EventRegistry

The EventRegistry project is a news aggregator that collects news articles from a variety of multi-lingual sources. It then identifies which news articles describe an event which is defined as any significant happening that has been reported by several articles. In addition to grouping events it also extracts semantic information about the groups. This can be things such as locations, dates, and who is involved. The collection of news is done by RSS feeds. Information is extracted through named entity recognition, where the entities are normalized through named entity disambiguation that point to Wikipedia resources. This process is called Wikification. Another form of semantic enrichment is to categorize news articles using the DMOZ taxonomy (Leban et al., 2016).

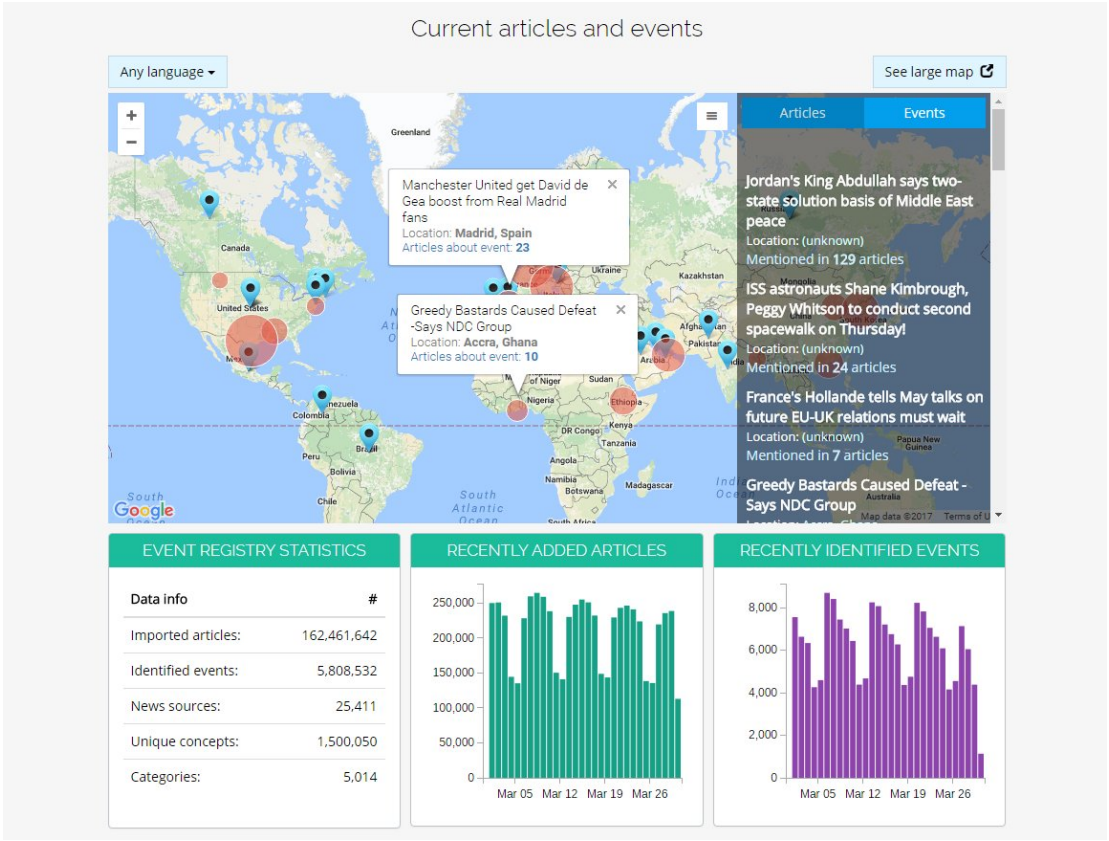

Figure 2.3: EventRegistry showing map of current events

2.4 Machine learning

2.4.1 Unsupervised learning

Unsupervised machine learning methods are techniques for discovering patterns in data without previous knowledge or objective evaluations. No training set is needed when using unsupervised learning algorithms (Gentleman and Carey, 2008).

k-means

K-means is an unsupervised algorithm for clustering data used for many different applications. It works by initializing K number of clusters with random data points being assigned as a centroid. After the initial state has been set it starts an iteration by assigning each data point to a cluster based on the nearest center. It then updates the clusters and calculates new centers for the next iteration. It stops when the iteration criteria have been met (Jin and Han, 2016).

DBSCAN

Ester et al. (1996) proposed the DBSCAN algorithm to solve the problem of discovering clusters without requiring a lot of previous domain knowledge. One of the drawbacks with the k-means algorithm is that the amounts of clusters must be set before running the algorithm. When dealing with substantial amounts of data where the number of clusters is unknown there should be an approximated guess based on incomplete information. DBSCAN allows the discovery of clusters with arbitrary shapes while requiring as little domain knowledge as possible for the input parameters. It is also designed to work fast and efficiently on large sets of data such as those from databases.

2.4.2 Supervised learning

Supervised machine learning methods use training data as a guide to which output or label an input should result in. Training sets are created from labeled data examples that correspond to some real-world use that the function is trying to solve. After a model has been trained using a training set it can be used to make prediction on new data (Kotsiantis, 2007).
One of the major uses of supervised learning methods is the task of automatic document classification. This task deals with the process of automatically assigning x labels to documents.

The labels are predetermined based on existing knowledge. An example of this is to classify news articles into a predefined set of news categories, such as politics, health, science, technology. Multi-class classification is when there are more than two classes. (Veloso et al., 2006). Multi-label classification problems deal with assigning potentially multiple labels to a single document (Zhou and Zhang, 2016).

In the context of classification models the term accuracy can be used to measure how close a model is to the reality being modeled (Sammut and Geoffrey I. Webb, 2010).

Support vector machines (SVM)

Support vector machines is a set of algorithms most commonly used for tasks such as regression and classification for supervised learning. SVM have been used to solve a variety of machine learning problems including: face detection, voice recognition etc. While SVM are binary classifiers they can be used for solving multi-class labeling problems (Byun and Lee, 2002).

Neural networks

Neural networks or artificial neural networks is computational approach to problem solving based on mimicking a biologic brain. Neural networks use artificial neurons that take inputs and estimate outputs after running them through the layers of the network. Each of the neurons has weights that determine how an input is to be handled. The neurons are connected through layers i.e. input layer, hidden layer(s), and output layer (Wang, 2003).

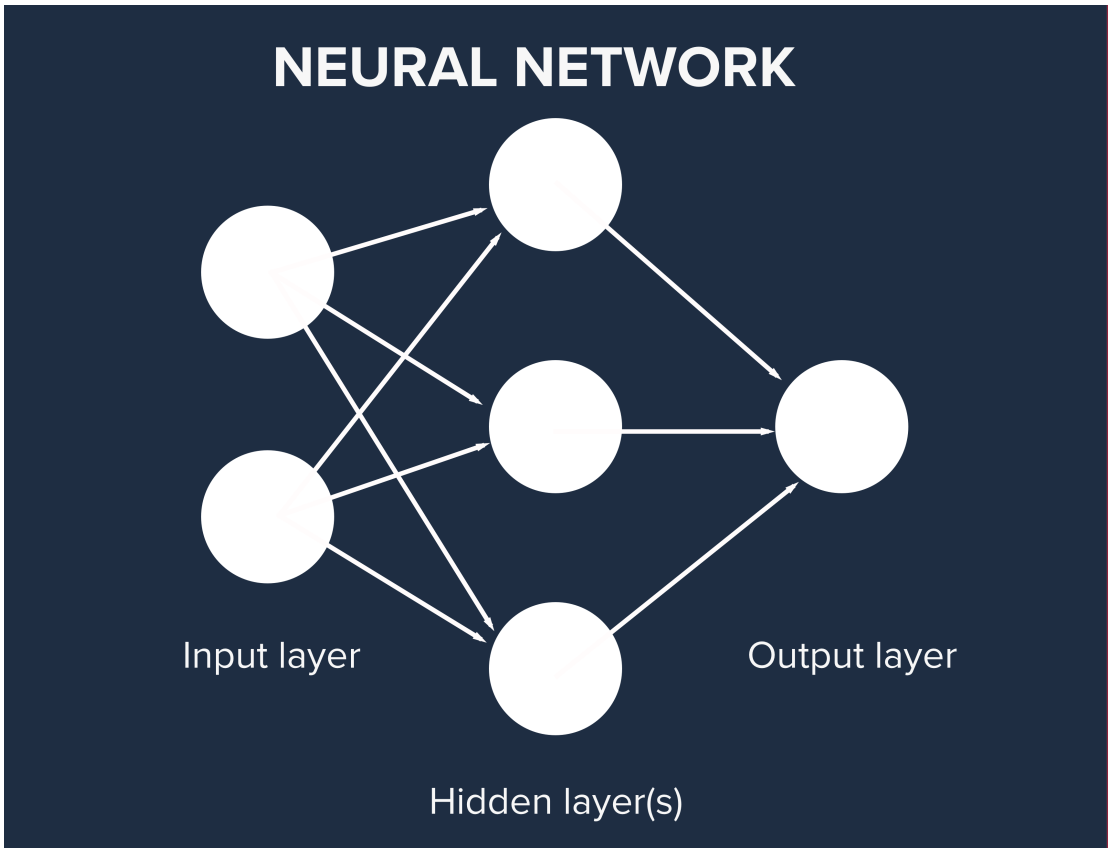

Figure 2.4: A neural network with two inputs, one hidden layer and two outputs

One neural network architecture that is widely applied is backpropagation. After signals have been sent from the input through one or more hidden layer(s) it checks to see how close the actual output is to the desired output. When it misses on an output it performs a backpropagation task that tells the layer(s) to change its weights, to get closer to the desired output (Hecht-Nielsen, 1989).

When multiple hidden layers are combined in a neural network it is often referred to as a deep learning network. In recent years deep learning networks have been successful in many tasks previously deemed difficult and continue to break new ground in artificial intelligence. The usage of GPUs (Graphical processing unit) to perform these computationally demanding tasks has enabled more widespread usage of deep neural networks due to models being trained quicker (Oh and Jung, 2004).

2.4.3 Precision, recall and F-measures

Precision and recall is used to measure the performance of information retrieval and information extraction systems. F-measure is a weighted combination of precision and recall

(Makhoul et al., 1999).

Precision deals with substitution and insertion errors (Makhoul et al., 1999).

$$Precision = \frac{True\ positive}{True\ positive + False\ positive} \tag{2.1}$$

Recall deals with substitution and deletion errors (Makhoul et al., 1999).

$$Recall = \frac{True\ positive}{True\ positive + False\ negative} \tag{2.2}$$

The F1 score or F-measure is a way of assessing the accuracy of a model using precision and recall.

$$F1 = 2\,\frac{Precision \times Recall}{Precision + Recall} \tag{2.3}$$

2.4.4 Dimensionality reduction

Dimensionality reduction removes information that is redundant or not generally helpful, while extracting useful and key features. In this context, the features are words and phrases in news articles. Removing grammatical features such as conjunctions and pronouns in addition to standard stop words which are present in most articles will reduce the feature space (Webb et al., 2011). Dimensionality reduction is often called feature extraction.

2.5 Natural Language Processing (NLP)

Natural language processing or NLP for short is a research field that explores approaches to making computers understand natural language.

2.5.1 Named entity recognition (NER)

Named entity recognition, sometimes also called named entity recognition and classification is an information extraction task that deals with recognizing named entities such as people, places, and organizations in text. This process can also be combined with the Wikification task of linking named entities in text to already known ones. One such example is the DBPedia Spotlight project (Mendes et al., 2011).

2.5.2 Sentiment analysis

Sentiment analysis, sometimes known as opinion mining is the process of identifying the sentiment or opinion in text and speech towards other opinions, topics, events etc. (Liu, 2010). Sentiment analysis of microblogging has been increasingly popular as these social networks have attracted more users over the years. One of the simpler approaches to performing this task is looking through each word in a microblog post and matching it to a list of known positive and negative words. Such a list can also have a scoring system to indicate how positive or negative it is, and there is the possibility of including emoticons (Nielsen, 2011). Other approaches are to use supervised or semi-supervised models trained on a corpus (Zhou et al., 2013).

2.5.3 Text summarization

Automatic text summarization reduces a text with the intention of only retaining the important aspects. There are two general approaches to performing this task; extraction and abstraction.

Extractive approaches use statistical and linguistic methods to select the most important sentences, paragraphs etc. in the document. After that it presents a concatenation of the most important sentences as a summary.

Abstractive methods attempt to understand the text and retell it using fewer words than the original document (Gupta and Lehal, 2010).

2.5.4 Keyword extraction

Keyword extraction is the task of identifying the most important words or terms in a document automatically. This can be either singular keywords or key terms and phrases (Lott,

2012). Keyword extraction methods can be either supervised as explained in section 2.4.2, semi-supervised, or unsupervised as elaborated on in section 2.4.1.

Part-of-Speech tagging (Pos tagging)

Part-of-Speech tagging is the process of assigning grammatical tags, also called Part-of-Speech, to written text (Sammut, 2010).

Lemmatization

Lemmatization is the process of reducing the inflected form of a word to its dictionary or lemma form. This will reduce dimensionality by reducing the amount of distinct words. The process of stemming also tries to solve the problem of reducing the inflected form to a base word, but unlike lemmatization it is not based on part-of-speech or a dictionary (Manning et al., 2008).

2.5.5 TF-IDF

Term frequency-inverse document frequency is a way of statistically represent a document based on the importance of the words. Each word is counted within each document in the corpus. The term frequency count is then compared to an inverse document frequency count (Blei et al., 2003).

2.6 Software Engineering

Software engineering is an engineering discipline that focuses on the practicalities of developing and delivering useful software (Sommerville, 2010).

2.6.1 Prototyping

When developing software that is to a degree exploratory, prototyping can be useful to rapidly test new solutions and make changes early in the development process (Sommerville, 2010).

> A prototype is an initial version of a software system that is used to demonstrate concepts, try out design options, and find out more about the problem and its viable solutions (Sommerville, 2010).

2.6.2 Technical debt

When adding components and features incrementally at a rapid pace, the risk of technical debt is a concern. Developing software where the long-term state of the system it not reflected upon can lead agile projects into technical debt (Kruchten et al., 2012). Cunningham, who introduced the term in 1992, described it as:

> Shipping first time code is like going into debt. A little debt speeds development so long as it is paid back promptly with a rewrite (Tom et al., 2013)

Code that requires refactoring is a form of technical debt (Tom et al., 2013)

2.6.3 Refactoring

Refactoring can clean up existing technical debt, and avoid debt that can occur in the future. It involves making changes to the code structure in such a way that it does not have any impact on its functionality. The process can make the code easier to understand and help make it ready for functionality added later (Davis, 2016).

2.6.4 Software development process

XP XP or Extreme programming is a methodology focused around improving software quality and rapidly changing customer requirements (Beck, 1999).

Kanban The Kanban method in software development was inspired by the lean manufacturing system implemented by Toyota with a focus on delivering products Just-In-Time. Kanban deals with managing the workload in such a way that developers are focused on a few tasks, instead of being overloaded with way too many (Gross and McInnis, 2003). Some of the important concepts of Kanban include:

1. Visualizing the workflow - Make the status of each task visible
2. Limit workflow - Only focusing on a few tasks at any given time will help getting the tasks done faster.
3. Learn and improve continuously - Continually reflecting upon the process using retrospectives will help improving the overall process and ensuring that the practices are helping us perform as well as possible.

Iterations

Iterations are short time-boxes where a certain number of features are worked on. The idea is to keep delivering the customer a working product at the end of every iteration and continually adapt to new requirements and changes (Larman and Basili, 2003). Developing this way will ensure that the project can respond to changes.

User stories

User stories is a way of managing requirements in software development in both Kanban and XP. User stories describe features software needs to contain the necessary functionality expected by a user, and can be used to split the requirements into smaller pieces that are more manageable to work with (Ambler, 2013).

2.7 Research methods

2.7.1 Quantitative methods

The data collected using qualitative methods are measurable. When you have data that are measurable and comparable, you can use quantitative methods to analyze these data (Punch, 2005). In system development, qualitative methods can i.e. be used to measure numerical data outputted by the system.

2.7.2 Qualitative methods

Qualitative methods differ from qualitative methods in the way that you get more complementary data. One example of a qualitative method is an interview. The interview object will often give detailed answers, and the purpose of an interview in many cases is to get in depth about a subject (Punch, 2005). In the field of system development, it can be used to get response on the users experience with a system, or their needs.

2.8 Technology Acceptance Model

The Technology Acceptance Model (TAM) was introduced by (Davis, 1986). The model is used to provide measures for predicting and explaining use of information systems (Davis,

1989). ”The goal of TAM is to provide an explanation of the determinants of computer acceptance that is general, capable of explaining user behavior across a broad range of end-user computing technologies and user populations, while at the same time being both parsimonious and theoretically justified” (Davis et al., 1989). A model that is not only able to predict, but also able to give an explanation to why a system is acceptable or unacceptable is ideal.

TAM focuses on two theoretical constructs:

Perceived usefulness ”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 ”The degree to which a person believes that using a particular system would be free of effort” (Davis, 1989).

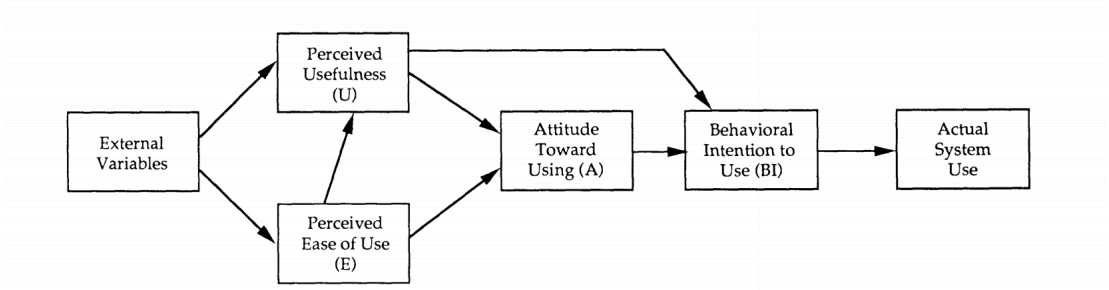

Figure 2.5: Technology Acceptance Model (TAM) (Davis et al., 1989)

Figure 2.5 displays the Technology Acceptance Model. Perceived usefulness (U) and perceived ease of use (EOU) is influenced by external variables. U is also affected by EOU. If improved EOU can enable a person to accomplish more with the same effort is will have a direct impact on U (Davis et al., 1989). If two systems are equally easy to operate, but one of them provide better results, the one with the better results will be perceived as the most useful. Same external variables that may influence EOU include: training, documentation, and user support consultants (Davis et al., 1989).

2.8.1 TAM2

TAM2 was proposed as an extension to TAM. Figure 2.6 shows the additional constructs added to the model.

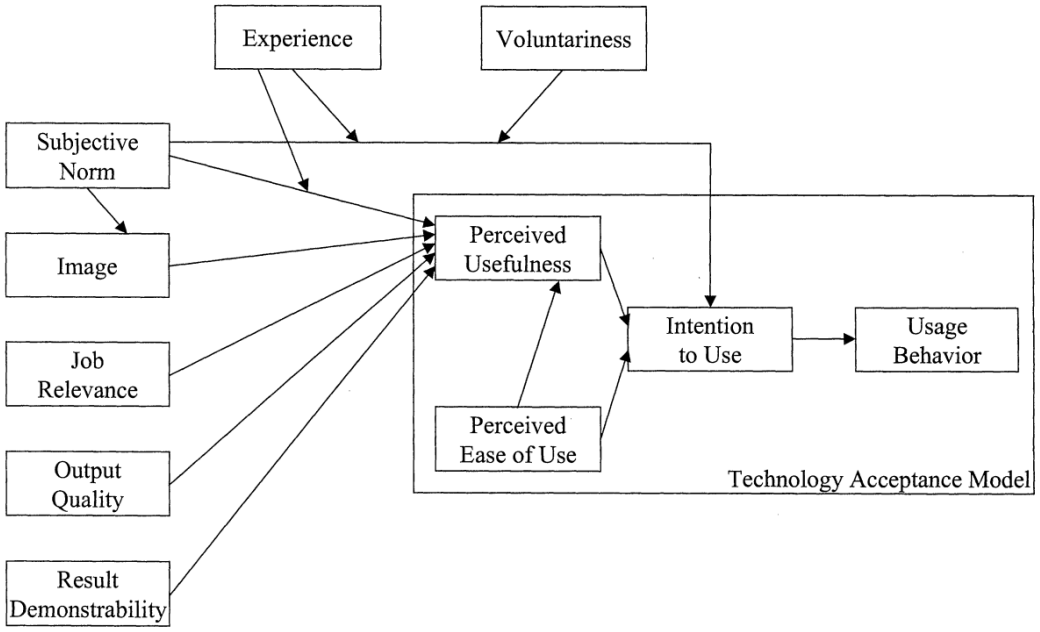

Figure 2.6: TAM2 - Extension of the Technology Acceptance Model (Venkatesh and Davis, 2000a)

The constructs are split into two categories:

Social influence processes

The Social influence processes consists of three interrelated social forces impinging on an individual facing the opportunity to adopt or reject a new system. The three social forces affect perceived usefulness and usage intention (Venkatesh and Davis, 2000b).

Subjective norm ”Person’s perception that most people who are important to him think he should or should not perform the behavior in question” (Venkatesh and Davis, 2000b).

Voluntariness ”The extent to which potential adopters perceive the adoption decision to be non-mandatory” (Venkatesh and Davis, 2000b).

Image ”The degree to which use of an innovation is perceived to enhance one’s ... status in one’s social system” (Venkatesh and Davis, 2000b).

Cognitive instrumental processes

People judge perceived usefulness in part by cognitively comparing what a system can do with what they need to get done in their job (Venkatesh and Davis, 2000b).

Job relevance ” An individual’s perception regarding the degree to which the target system is applicable to his or her job” (Venkatesh and Davis, 2000b).

Output quality How well a system perform its tasks (Venkatesh and Davis, 2000b).

Result demonstrability Individuals can be expected to form more positive perceptions of the usefulness of a system if the correlation between usage and positive results is clear (Venkatesh and Davis, 2000b).

Perceived ease of use ”The degree to which a person believes that using a particular system would be free of effort” (Venkatesh and Davis, 2000b).

Chapter 3

Research Method

3.1 Design Science Research

This part contains a description of the research method that will be used in the project to help answer the research questions and develop a prototype of the application system.

3.1.1 Design science

For this project, the research framework called design science research has been chosen to develop a prototype of the system. Design science research involves the development of artifacts and subsequent analysis of the performance of the artifacts to measure their effectiveness (Hevner et al., 2004). The goal is to improve the understanding of different parts of information systems (IS) through developing and evaluating products for use in IT settings.

Artifacts or outputs in design science research can take on different forms depending on what the research goal is. Vaishnavi and Kuechler (2004) provide a list over these and their research contribution intention:

Constructs The conceptual vocabulary of a domain.

Models Sets of propositions or statements expressing relationships between constructs.

Frameworks Real or conceptual guides to serve as support.

Architectures High level structures of systems.

Design principles Core principles and concepts to guide design.

Methods Sets of steps used to perform tasks - how-to knowledge.

Instantiations Situated implementations in certain environments that do or do not operationalize construct, models, methods, and other abstract artifacts; in the latter case, such knowledge remains tacit.

Design theories A prescriptive set of statements on how to do something to achieve a certain objective. A theory usually includes other abstract artifacts such as constructs, models, frameworks, architecture, design principles, and methods.

Design as an artifact

Hevner et al. (2004) describes an artifact because of design-science research in information-systems, created to provide a solution to a specific problem. Artifacts are not necessarily meant to be complete implementations of an information system, but rather serve as showcase of innovations, ideas, and products.

Problem relevance

The relevance of a project hinges on it being relevant to a problem space where it addresses a problem face by people, organizations in information technology (Hevner et al., 2004).

Design evaluation

Evaluating the artifact is important to determine its effectiveness and measure how well the goals have been met. The chosen evaluation methods must demonstrate that the design artifact is of high quality and provides the utility it is meant to (Hevner et al., 2004).

Research contributions

The research performed must contribute not only to a business perspective, but also a research perspective. In order for a design artifact to contribute, it must either solve a previously unsolved problem, create the foundation for future models or methods, or be creative in development and use of evaluation methods(Hevner et al., 2004, p. 87).

Since the goal of the thesis is to create an instantiation of a design artifact, the artifact itself and the experiences with implementation and experimentation will provide contribution to the research area according to Vaishnavi and Kuechler (2004).

Research rigor

Research rigor is an important part of design science research as the application of rigorous methods in both development and evaluation ensures that the quality of the research contribution will be of high value.

Design as a search process

Design science is an iterative process where the goal is to search for an effective solution to a problem. The iterations take the form of cycles where new alternatives are created and tested to provide feedback on how well it performed.

The artifact in question will be developed according to agile methodology with iterations when developing the artifact to react to changes in specifications based on results. This will also allow for new features to be planned and implemented during the development process.

Communication of research

The research results must be presentable and understandable for both technology oriented and management oriented audiences (Hevner et al., 2004). For a technology oriented audience, the research contribution must provide enough details to be implemented in other settings. For the same audience, an inclusion of which methods have been used and details of the parameters will be included. The creation of front-end tools for exploration of features and images with detailed descriptions will be used to communicate the results of the research to a more general audience.

Chapter 4

Development Process

4.1 Requirements

Due to the exploratory nature of this research we had a continuous dialog with Wolftech to get feedback and ideas for new features. We developed the software using an iterative process.

It was a goal to make News Hunter compatible with the standards and formats Wolftech were already using in their newsroom systems. This included using similar tools and technologies that they were utilizing in their development process. It was therefore necessary to get an overview of their developing environment at an early stage, and obtain the required knowledge.

4.2 System overview

The system is divided into distinct parts that each have their own responsibility.

4.2.1 News feeder

The News Feeder component is responsible for retrieving and receiving messages. The messages can come from a variety of different data streams. It then uses an API to create a summary of the full text if it is over a given character threshold. The full message and the summary is stored in an SQL database along with the message metadata. The News Feeder also has the option to translate text into English using commercially available APIs. This

feature makes the system multilingual. After the message has been successfully stored, it is sent to the Message Analyzer for deeper analysis.

4.2.2 Message Analyzer

The Message Analyzer is responsible for analyzing the received messages. It uses different APIs for information extraction tasks. It extracts keywords and named entities from incoming messages, which can then be used to describe the message in a knowledge graph. All the data retrieved from the analysis process along with some metadata identifying the message is semantically lifted and stored in the triplestore.

The Message Analyzer is also responsible for finding other messages that describes the same event using a clustering algorithm. These events can then be grouped together. Topic modeling is also implemented to find out which topics are present in messages that are part of an event.

4.2.3 Web API

The Web API component is a REST API that exposes the data layer to the client side front-end. It contains built in queries for retrieving messages along with semantically enriched data.

4.2.4 Front-End

The front-end layer of the stack is built to display the information stored in the graph in a web browser. The purpose is to show the system capabilities and prototype the potential use cases.

4.3 Tools

Throughout the development of the News Hunter system a variety of technologies and tools were used.

In section 4.4, these tools and technologies will be introduced at the stage they were used for the first time.

4.4 Iterations

The News Hunter system built in this project was created with the principles of minimum viable product (MVP) in mind. The goal here is to build the minimum number of features that is required for the system to work as intended, and then evolve from there. This will make the system more adaptable, as there will be a fewer features, and the features that are implemented with each iteration will be thoroughly tested before new features are introduced.

The following iterations are divided by features that are meant to be included in the system rather than by a specific time length. Iteration 1 started in the beginning of August 2016.

4.4.1 Iteration 1: Overview and setup

When we examined the *WT Semantic Prototype*, explained in section 2.2.3, and the previous work found in literature, we identified some core components that were shared. The shared components were considered important, and laid the foundation for this project. Some of the shared functionality of these projects were the extraction of named entities and keywords in text. This functionality is shared by a variety of different actors that extract and process potentially useful data from news content. EventRegistry (Leban et al., 2016) and NewsBrief (Krstajić et al., 2010) were some of the tools found in the literature.

As for the retrieval of text for analysis, most of the found literature was focused on news articles. Since the existing code from *WT Semantic Prototype* was built around Facebook posts, we were using that as a starting point instead of traditional news articles.

Goals

The goal of the first iteration was to get an overview of the *WT Semantic Prototype*, compare the efforts to related work found in literature, and then set up a development environment. In addition to this we wanted to extract the main functionality and include it in the new project. The functionality we wanted to include in our first build was to be able to gather data from Facebook, analyze it with the existing analysis pipeline and store the results in a graph.

1. Overview of existing work
2. Set up development environment

3. Extract the key working components from previous work

Tools and technologies

Visual Studio 15 Community Edition The main IDE for development is Visual Studio 15 Community Edition due to the existing work being built on ASP.NET combined with a wish from Wolftech that the work is being done using technologies and platforms that mirror their working environment.

Text editors Lightweight text editors have been used for especially front-end development. Atom and Visual Studio Code have both been used extensively.

C# The main language used for back-end technologies was C# due it being the standard when working with ASP.NET applications and due to its widespread popularity and adoption.

BrightstarDB The main triplestore provider. It can take advantage of RDF/XML format for introducing existing ontologies into the database. This was used in the implementation we inherited, and we decided to continue using it going forward with the project.

Alchemy API This API uses machine learning algorithms to perform NLP and computer vision tasks. One of its features is to extract keywords, entities, sentiment etc. from unstructured text.

Version control These systems are useful when working on nearly any development project. They give the flexibility to try out new things without having to think about the implications. They also enable cooperation. In this project, a version control system called Git is used. Git is widely used among developers, and is a very robust system. The code is stored locally and on a private remote repository on GitHub. With the use of a Git client (i.e. SourceTree) it's possible to work effectively with Git. The use of a remote repository is also keeps a backup of the code in the cloud.

Project management New features that were under planning or development were put into the project management tool Trello.

Development process

In order to get an insight into the workings of *WT Semantic Prototype* we set up the development environment. The components from the prototype were investigated in Visual Studio. To create a simpler testing environment a new command line application project was created. The code from the previous work was then tested to inspect the results and to decide which parts of the project we could reuse in further development. The functional parts that we decided to continue using were the gathering of data through the Facebook API, semantic lifting with BrightstarDB, and data enrichment with the Alchemy API.
To get a good insight, the previous work had to be replicated. A developer account and a Facebook application was needed to get the data from the Facebook API. We then included a Facebook SDK into the project to enable the command line application to get data directly from the Facebook API for storing and further analysis (Foundation, Foundation). The data returned were Facebook posts from selected user accounts of US politicians in JSON format. For the Alchemy API, we used the existing project credentials to gain access to the service. The most essential functions that we decided to continue using were getting keywords, and named entities from text to describe the content of the posts.
When the results were returned from the Facebook API and analyzed through the Alchemy API we stored it into a triplestore using BrightstarDB as our RDF database. We used the existing setup for the database with the included ontology mapping, and stored it in memory for the time being. To replicate the data flow, we stored the returned JSON data from Facebook and Alchemy on disc.

When we had some insight into how each part worked, we cherry picked the parts that we wanted to combine into a new working pipeline. This new pipeline was then put into the command line application and used that as the basis for an MVP.

Validation

To validate the work done in the iteration we tested each component to see if it performed as expected. Getting data into the application from the Facebook API using C# was the first component that was tested. The test was deemed successful because data coming into the pipeline contained the expected result from the Facebook API.
The next component was dependent on the first component to function since it needed the

textual data gathered from the Facebook API in order provide data for analysis. If the component could send data to the Alchemy API and handle the returned results, the success criteria had been met.

The last component was dependent on both the Facebook API textual data and results from the analysis to create testable material. If Facebook posts and the data from the analysis could be stored in a triplestore, and then return data through a query, it would be successful.

1. Return data from the Facebook API
2. Get results from the Alchemy API
3. Store to triplestore and query

Results

We managed to get an insight into the key working components of the existing project and extract the core concepts from it. We ended up with a command line application that could pull Facebook posts from public profiles given a Facebook profile ID and parse the content to analyze it using the Alchemy API to gain keywords, named entities, and the sentiment of the text. We also implemented the possibility to query the database using SPARQL based on the existing ontology to look at the results of the semantic lifting process.

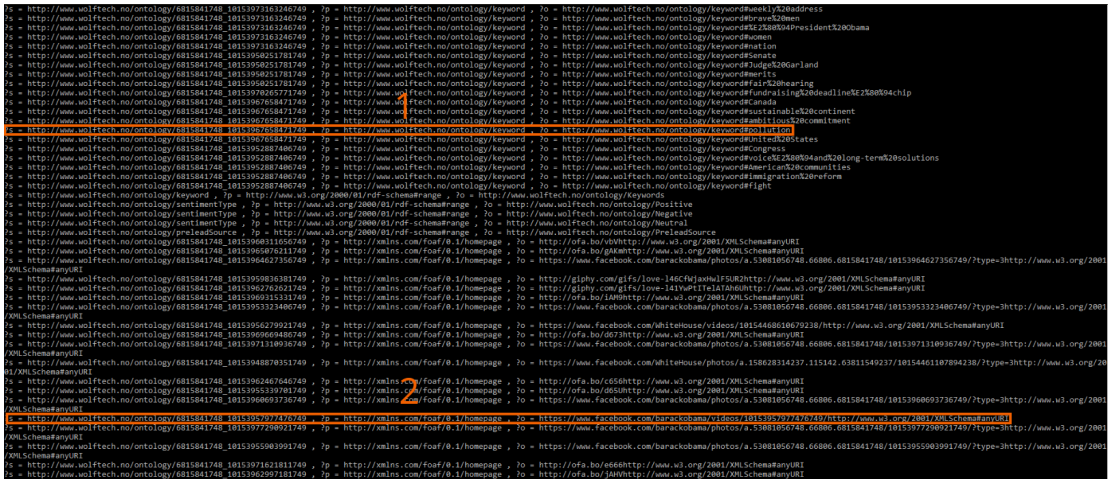

Figure 4.1: Results from command line application

The above image shows a result from the working pipeline with the triples printed out in a console window. The first marked triple shows the subject event which has a predicate keyword that points to an object for the keyword in question. Marked triple number two shows an event tied to a URL through the foaf homepage predicate.

4.4.2 Iteration 2: Ontology changes

We found that the ontology that was used in the previous system needed some changes to better fit the content that we wanted to include. The Wolftech News ontology was modeled around Facebook posts and Alchemy API data. In the planning of this system the inclusion of messages from additional sources was central. It was also a priority to make the system not reliant on Alchemy API data. To fit these requirements the ontology needed to be more generalized and scalable.

Goals

For the second iteration, our goal was to transition into a more precise ontology for the domain. We planned a switch to using BrighstarDB with Microsoft Entity Framework for handling the semantic lifting into the graph. This would also entail changing the graph over to an on-disc database instead of storing it in-memory.

1. Rework the ontology
2. Rewrite lifting process to ORM with EF
3. Store data in on-disc database

Tools and technologies

Entity framework Microsoft Entity Framework (EF) is an Object/Relational Mapping (ORM) framework that assists developers when working with relational data. Using this in conjunction with BrightstarDB allowed us to store and query data in the graph using programmatic models instead of writing them as strings which was the old way of doing it.

LINQ Language integrated Query is a .NET component that allows native data querying to domain specific models. Combined with BrightstarDB this allowed us to write LINQ-to-SPARQL queries in code that would be automatically translated to SPARQL, making it easier to create data-type specific queries.

Development process

The inherited ontology had its limitations, but some parts were reused in the version created in this iteration. The major change was the ”Descriptor” class. This class could be attached to messages. A keyword or entity could serve as descriptors.

In order to improve upon previous work, we changed from using the inherited approach with hardcoded string values for inserting triples into the store and instead employed the capabilities of BrightstarDB to take advantage of the .NET environment with the Entity Framework. This allowed us to insert and query the database by using an Object-relational-mapping (ORM) making it possible to query the data by transforming LINQ queries to SPARQL. The database was changed from in-memory storage that would be wiped on application shutdown to an on-disc storage that would be persistent.

Validation

To validate this iteration, we ran tests from the first iteration to see if they still performed as expected.

1. Pipeline still returns expected results

2. Data exist after application restart

Results

Figure 4.2: First version of the new News Hunter ontology

The system still got Facebook messages and analyzed them with the Alchemy API before storing them in a triplestore which was queried. The triplestore stored the data on-disc, and it was possible to query it after application restart. Since the pipeline still returned the expected results, the changes to the ontology was successful.
Apart from the database and data handling changes, no additional features were implemented at this stage. Our work in this iteration allowed us to more confidently and easily work with the data in the graph, both when lifting and querying.

4.4.3 Iteration 3: Full-stack application

During the analysis of different systems in iteration 1, the benefits of making the results accessible became clear. Therefore, this was the area of focus for the third iteration. Making the results accessible both through an API and a graphical user interface.

Goals

The goals for the third iteration was to make our system more scalable and enable the data analyzed to be made available through a GUI. In order to achieve this goal, we decided to convert from a command line application into a fully functional full-stack ASP.NET application. This would make the data accessible through a REST API, and a dynamic web front-end where the data could be presented and more easily navigated.

1. Set up ASP.NET application
2. Create REST endpoints
3. Design user interface
4. Implement AngularJS front-end framework
5. Implement the designed user interface

Tools and technologies

ASP.NET Microsoft ASP.NET is an open-source server side web application framework that was developed to allow the creation of dynamic web pages, web applications, and web services. For our project, we have created multiple web services exposed through a REST API using ASP.NET Web API REST architecture.

AngularJS Front-end Javascript framework. It is built for empowering data driven web applications. With the use of a variety of add-on packages asynchronous calls to the back-end can be very efficient. If the system continuously updates data, AngularJS makes it possible to update the view without having the user reload the page. We chose to use AngularJS over other popular front-end frameworks such as React, VueJS, or EmberJS because of Wolftech's existing technology stack which includes AngularJS and our goal to create something compatible.

Sketch Light-weight design tool with vector capabilities. It is widely used for user interface design and has a wide variety of additional packages that can be downloaded to fit most needs.

Marvel Web-based prototyping tool that is possible to integrate in Sketch. This enables rapid prototyping of the different screens. It is easy to use, and it empowers collaboration.

HTML & CSS This is two of the main technologies required for building web-pages. HTML (Hypertext Markup Language) provides the structure for a web page, and CSS (Cascading Style Sheets) provides the structured content with a layout and style.

REST API Representational State Transfer is used for transferring data over the HTTP protocol.

Software stack A software stack is used when different applications work together towards a common goal. A common method is to divide the system into a back-end and a front-end. The back-end of the system is often called the data access layer. It communicates with the database, and takes care of the heavy logic. The front-end is the presentation layer of the stack. It is what the end-user will interact with. For this project, the front-end has been designed to be experienced through a web browser.

Postman Postman is a software for API development that allows for easier testing of REST endpoints.

Development process

Our first challenge in the iteration was to set up an ASP.NET application, which required us to learn the architecture and best practices. Using Visual Studio templates and learning material we were able to set up a functional starting project with a REST API endpoint. After we had created the new application, the code from our existing command line application was added to it and endpoints were created. The endpoints allowed for external applications or clients to query the data available in the triplestore which was returned in JSON format.
For our front-end we first created some drawings of what we wanted the user interface to look like. The chosen drawing was then digitalized in Sketch App. A low fidelity prototype was created with the use of Marvel.

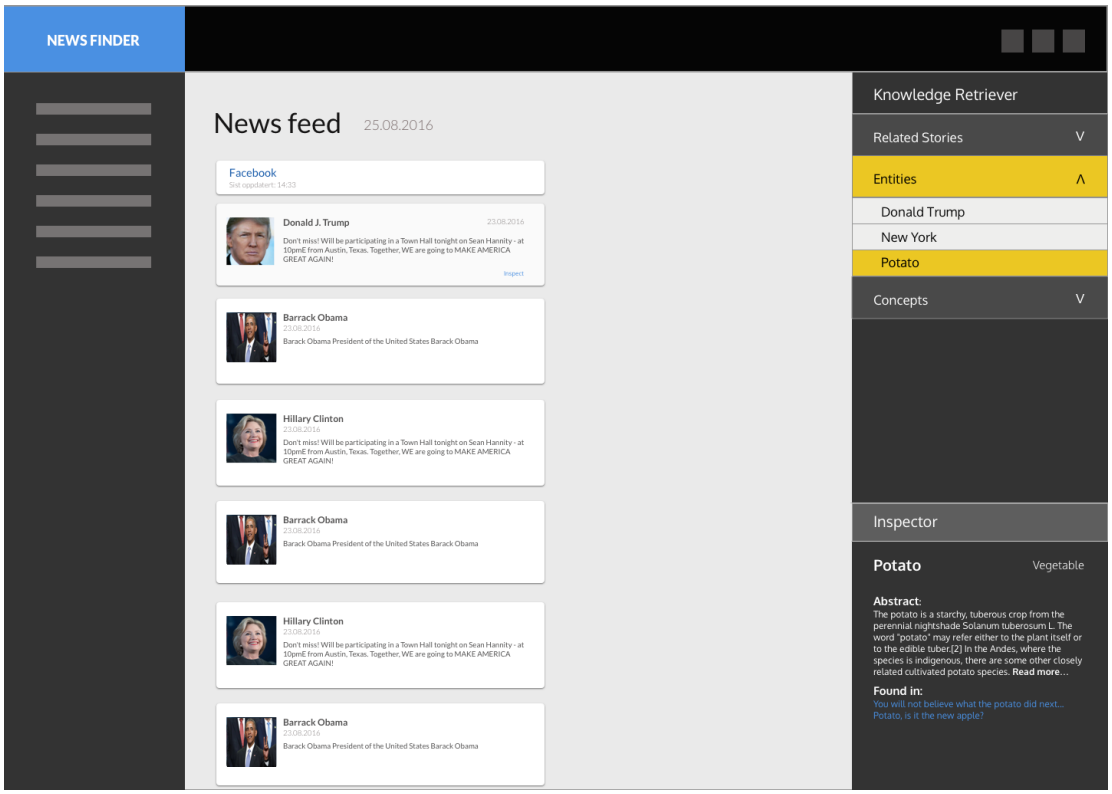

Figure 4.3: First version of the front-end design created in Sketch

Figure 4.3 shows the first design created in Sketch and then prototyped with Marvel. The main section of the window shows how the news feed was envisioned. A stream of the newest or searched for messages would be displayed as a list. Each of the elements were clickable, enabling the user to inspect the message. When a message was inspected, the right section of the interface, the Knowledge Retriever, would display named entities, keywords, and related stories. Each of the elements in the Knowledge Retriever section were themselves clickable, enabling additional information at the bottom of the section. I.e. clicking on a named entity would show its entity type, as well as its DBpedia abstract among other enriched data.

The design then went through a few short iterations before it was accepted and implemented with HTML and CSS combined with AngularJS to get dynamic capabilities. The two-way data binding enabled in AngularJS allows the front-end view to change when the data model is altered with new or updated data. It also enables the view to make changes to the data model.

Validation

To evaluate the iteration, we ran the web front-end to see if the results were displayed correctly.

1. Display triplestore data in the web front-end

Results

At the end of our iteration we ended up with a new functioning software stack. Our back-end could query the Facebook API for new data, process this using the Alchemy API, and store the results of the analysis locally in a persistent graph which was accessed through a REST API. With the use of Postman, we tested the REST API and got the desired data returned.

The front-end was able to make queries to the REST API and display the data results in a news feed with the ability to inspect each Facebook post for its keywords and named entities.

4.4.4 Iteration 4: Related stories and keyword extraction

To showcase some of the benefits of storing analyzed data in a semantic graph we wanted to make connections between different news stories. During this iteration, we based our work on a hypothesis that stories that share named entities can to some extent be considered related.

With the amount of data that ran through the system at this stage, the request limitations of the free version of the Alchemy API we were using started to become a constraint. Therefore, we started considering other libraries and algorithms that could extract named entities and keywords from text data where the request limitation was not an issue. The main priority was to find libraries or frameworks that could run locally to reduce the amount of external API calls for a more reliable system. One of the first algorithms we tested for extracting keywords from text was RAKE (Rapid Automatic Keyword Extraction).

Up to this point the only data in our system was retrieved from the Facebook API. This data has its limitations. Posts are often short, and the content is not always newsworthy. In some situations, a post can only include an image. We wanted our system to be able to analyze a variety of textual data from various sources. Social media, such as Facebook and Twitter can provide newsworthy content, but we also wanted to include news from traditional news

outlets. The first step to make our system capable of retrieving textual data from multiple sources was to create a standardized format for exchanging data.

Goals

Our target for this iteration was to display some of the capabilities achieved with having content stored in a semantic database. To show this, we planned the implementation of a related story feature that would allow the end-user to easily find other news stories that were seemingly related. We also considered some of the potential limitations of the analysis of text. We also explored the possibilities of including data from multiple sources which resulted in us trying to create a standard format for incoming data for analysis.

1. Create an algorithm for finding related stories using named entities
2. Standardize input format to enable data from various sources
3. Circumvent limitations with the free Alchemy API account

Tools and technologies

RAKE Rapid automated keyword extraction is an algorithm for unsupervised keyword extraction in text.

Development process

Our first goal was to find related stories. We designed a simple algorithm to connect stories based on their mutual named entities. The algorithm in this stage was simplistic and found stories that linked to the same named entities and ranked the related story based on the number of shared named entities. In the front-end, we added the possibility to view the related stories during the inspection of a Facebook post.
Our implementation of RAKE was made possible due to an existing library for C#.
To standardize the file format, we received some sample data from Wolftech that we could base our format on. The sample data included textual data from Twitter and RSS structured in JSON.

Validation

1. Check results from RAKE algorithm

2. Test related stories functionality

Results

The results from the related stories feature were visible in the user interface. The algorithm worked as intended, however it had predictable limitations in terms of accuracy. Since it only required shared named entities, many stories that were seemingly not related were regarded as such by the algorithm. While these stories were related in the sense that they included the same people or places, the actual situations (Who, what, where, when, why) were not taken into consideration. This was especially the case for stories of greater length, due to the amount of named entities found in these stories. This caused them to appear as related to many stories.
The implementation of the RAKE algorithm allowed the system to analyze textual data for keywords in textual data regardless of language without the need to send data out of the application and wait for a response from an external server. This allowed for quicker analysis of keywords that was also not held back by a request limitation.
The standardization of the file format for data input allowed us to create a basis for more information streams to be included in later iterations.

4.4.5 Iteration 5: Linking to open data

One of the advantages of storing the data in a semantic graph is the possibility to connect it to open linked data, as described in 2.1.3. This is a feature that differentiates semantic news aggregators from non-semantic news aggregators in the way that it enables semantic enrichment of the data. This enables News Hunter to take advantage of Wikification, which both BBC, NEWS, and EventRegistry has taken advantage of as explained in 2.3.1.

Goals

For this iteration, we explored the possibility of including data from external resources. The Alchemy API provides links to external resources for the extracted named entities. We wanted to give the end-user the possibility to connect the named entities found in messages with enriched background information. The background information should be retrieved from open linked data To do this we wanted to view data from external sources in the front-end.

1. Create endpoint for querying DBpedia data

2. Display results from external sources in front-end

Tools and technologies

DBpedia A project that extracts information from Wikipedia and structures it according to semantic web standards.

Development process

We chose to work primarily with DBpedia as our source for background information data as it is easily and freely accessible in addition to us having previous experience with the service. A new REST resource was created that would take in named entities and send SPARQL queries to DBpedia for retrieval of the background information we wanted. We solved this by creating pre-built SPARQL queries for different types of named entities such as people or places, and return some relevant information that we could show to the end-user. Our front-end was changed to accommodate this functionality and give the user the ability to request background information on a named entity.
These changes allowed the system to send named entities from the front-end via a REST resource and translate it into a SPARQL query for DBpedia that would return the requested information for example as an abstract or an image of the named entity. The system was restricted when it came to what information it could retrieve from DBpedia due to some named entities containing different predicates than others, so we decided on some of the most general and commonly used across different entity types. This allowed us to show how the data from the graph could be combined with different external resources and give more information about named entities.

Validation

The efforts could be considered a success if we were able to show the results from DBpedia queries in the front-end user interface.

1. Display DBpedia results in the web front-end

Results

Figure 4.4: Screenshot of a shortened DBpedia abstract for the city of New York as it appeared in the front-end

The figure 4.4 displays the results from DBpedia when the named entity New York City is selected.

4.4.6 Iteration 6: Local named entity recognition

In a news production environment, there will be a massive amount of stories that need to be processed. Extracting data from these stories will require a lot of calls being made to different services. Having these services locally will require less processing time. The free version of Alchemy API has request limitations, so parts of the system will eventually break as workload grows. Identifying and leveraging named entities is one of the core functions in the system, and a lot of requests are being sent to the Alchemy API for entity retrieval. Exploring different local options for this task could therefore prove to be beneficial in regards to scalability and processing time.

Goals

The goals for this iteration were to get up and running with a local NER service to circumvent the need for external services and aid in further testing and development of the system. This would include considering possible libraries and solutions, choosing one which could help us achieve our overarching goal, and implementing this into the system.

1. Create local NER service

Tools and technologies

Python General-purpose high-level programming language that is widely used. It is a popular language for data gathering, machine learning, and text analysis. It has an extensive library of packages that can be utilized for this purpose.

Pycharm IDE for developing Python applications.

Flask Microframework for creating servers with REST endpoints.

Spacy Spacy is a natural language processing library for the Python that can perform tasks such as part-of-speech tagging, named entity recognition and deep learning.

DBpedia spotlight Tool used for annotating textual data with entities that link to their respective DBpedia sources and can be used as a basis for Wikification.

Development process

Our criteria for choosing a NER system or library were primarily accuracy, speed, and ease of use. To find a service that would fit our needs, we explored different solutions in different programming languages. Python was chosen as the programming language, due to its use within the field of machine learning and its extensive choice in available libraries. When comparing Python to other languages, like Java, C#, R etc. we emphasized simplicity and considered other planned features to find a language that would fit all needs. This would allow us to avoid having to use additional languages for each purpose.

With the use of Python, there are a variety of different NER libraries available. We chose Spacy due to it being fast, easy to use, within the target goal for accuracy, and having a permissive license. Since we chose to use a library native to Python we ended up implementing

a REST API using the Flask micro-framework that would be able to run locally alongside the rest of the system, and that would allow us to use the library with much of the existing code intact on the back-end side.

```
@app.route('/')
def index():
    return "Welcome to the NLP rest api for News Hunter"


@app.route("/analyze", methods=['POST'])
def analyze():
    if request.json is None:
        entities = analyzer.analyze_text(request.form['Title'] + " . " + request.form['Body'])
    else:
        entities = analyzer.analyze_text(request.json["Title"] + " . " + request.json["Body"])
    return jsonify(entities)


@app.route("/summarise", methods=['POST'])
def summarise():
    if request.json is None:
        summary = txtrank.get_summary(request.form['body'])
    else:
        summary = txtrank.get_summary(request.json["body"])
    return jsonify(summary)


@app.route("/sentiment/<string:text>", methods=['GET'])
def sentiment(text):
    sent = SentimentAnalyzer.get_sentiment_of_text(text)
    if sent is None:
        return jsonify(SentimentAnalyzer.get_neutral())
    else:
        return jsonify(sent)


@app.route("/sentiment/nor/<string:text>", methods=['GET'])
def nor_sentiment(text):
    return jsonify(SentimentAnalyzer.sentiment_scores(text))


@app.route("/keywords", methods=['POST'])
def keywords():
    if request.json is None:
        entities = get_keywords(request.form['Title'] + ". " + request.form['Body'])
    else:
        entities = get_keywords(request.json["Title"] + ". " + request.json["Body"])
    return jsonify(entities)

```

Figure 4.5: Flask end-point as it stands at the end of the development process

The Alchemy API includes links to DBpedia and other additional external resources for open linked data. Spacy does not have a feature like this, which forced us to include DBpedia Spotlight. This service allowed us to maintain a connection between named entities and their DBpedia links so we could continue using the SPARQL queries to DBpedia for background information. DBpedia Spotlight can be installed locally in addition to being available as a service.

The system was now a composition of multiple services that exchanged data through REST calls to create a working pipeline. Waiting for the entire system to run when API functionality is tested can be time consuming, therefore it is often a necessity to use tools to test the different endpoints. In this project, we chose to use Postman for this purpose. The tool gave us the opportunity to thoroughly test the different endpoints, and therefore potentially save time otherwise spent compiling and/or debugging.

Validation

In order to evaluate the iteration, we set up a server using Flask to expose the capabilities of the different NER solutions that we chose. We used Postman to see that the results from each solution worked as intended. We also set up a way of combining the two approaches to see if this could provide better results.

1. Check results of Spacy
2. Check results of DBpedia Spotlight
3. Check results of both combined

Results

Our stack allowed us to easily test the system without the need for paid services. We could circumvent our cost limitation when it came to the amount of data being analyzed per day which would make further testing and development less costly. The downside of this was that it also required more components to be maintained and that increased the complexity of the whole project.

4.4.7 Iteration 7: News harvesting from web & updated front-end design

Up to this iteration we had only used Facebook posts as our source for textual data. Social media is useful for modern journalists when researching and looking for new stories, but retrieval of data from news outlets is still an important source for journalists and should therefore be included.

Goals

One of the goals of the system is to be compatible with data from multiple sources. The priority was to include data from some of the major news media outlets as a basis for further testing. With the inclusion of data from other sources, the front-end had to undergo changes.

1. Set up the Python scripts for getting news articles
2. Create scripts for sending news articles for analysis
3. Create and implement the next version of the front-end

Tools and technologies

Web crawling This approach were used for the purpose of gathering stories from a variety of different news sites. The bot (spider) traversed through news sites and scraped all the news content.

Newspaper (Library) Library for retrieving news articles from the web and performing basic NLP tasks on the retrieved data.

Hashing A hash function can take a string as an input and convert it into a usually shorter fixed sized value or key that corresponds to the initial string. This process is called hashing.

Development process

We decided to create a service for gathering news articles using Python due to the availability of libraries like Newspaper. This would make it possible to set up a service that would be able to gather the necessary data and do most of the parsing without the need to create something from the ground up. When posts from Facebook were used, the ID of each post was used as the unique ID for the message in the semantic graph. With the introduction of messages from other sources a different approach to finding unique identifiers had to be decided upon. Using XXhash algorithm, with salting disabled, on the URL of the news article or Facebook post provided a solution to this problem. This approach ensured a low probability of collisions between ID numbers. The processed news articles would then be read and posted to an endpoint that would analyze and store the messages in the graph.

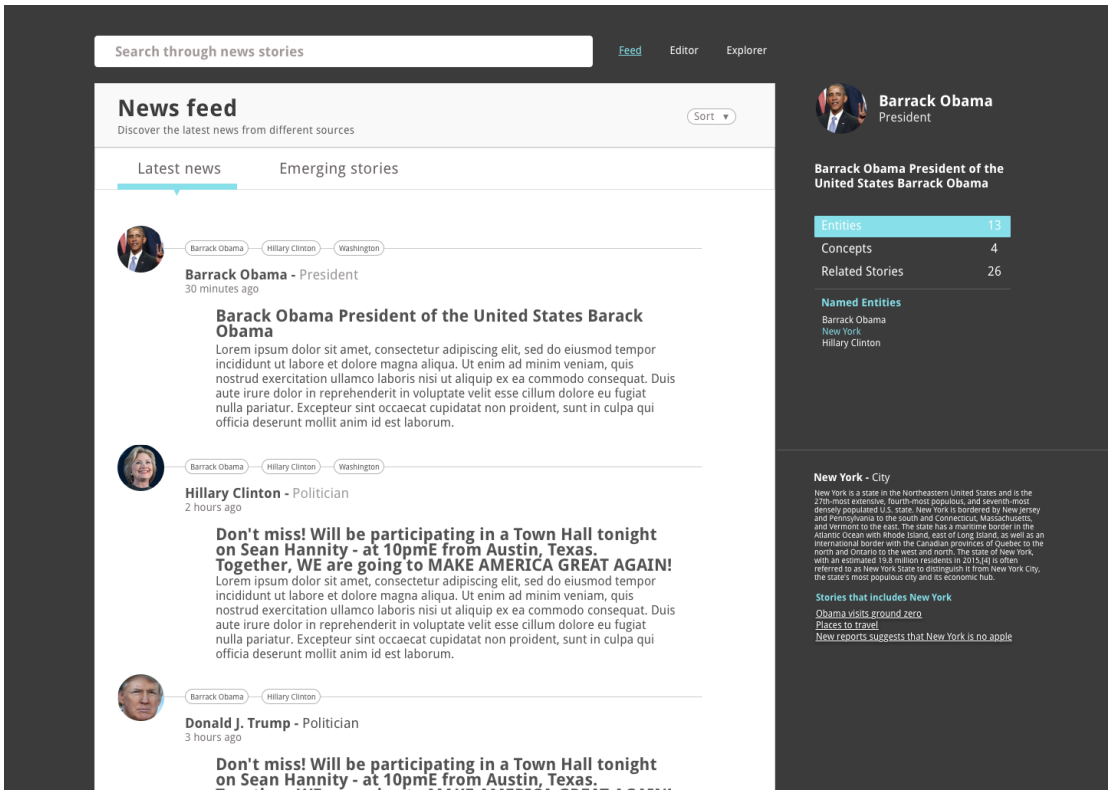

Figure 4.6: New design for the front-end news finder with the ability to inspect stories and show data from DBpedia

The new design as shown in figure 4.6 was intended to be a step forward that would address some problems with the previous version and make room for new features that were planned. The main news feed was altered to take better advantage of the available space and make room for the message text to be readable inside the view.

Validation

To validate this iteration a new version of the front-end with capabilities to display the results from the web crawling process was created. If stories from other sources than Facebook was shown, the iteration could be considered a success.

1. Display the results from web crawler in the updated version of the front-end

Results

The resulting service could crawl multiple major news media sites and download parsed version of the news articles for use in the system. For it to scrape a site, the site URL had to be manually added to a list. A downside to using this library was that it did not work in

all cases and it sometimes did not retrieve the newest articles. After processing it could put these articles into the News Hunter system for storage in the graph before analysis.

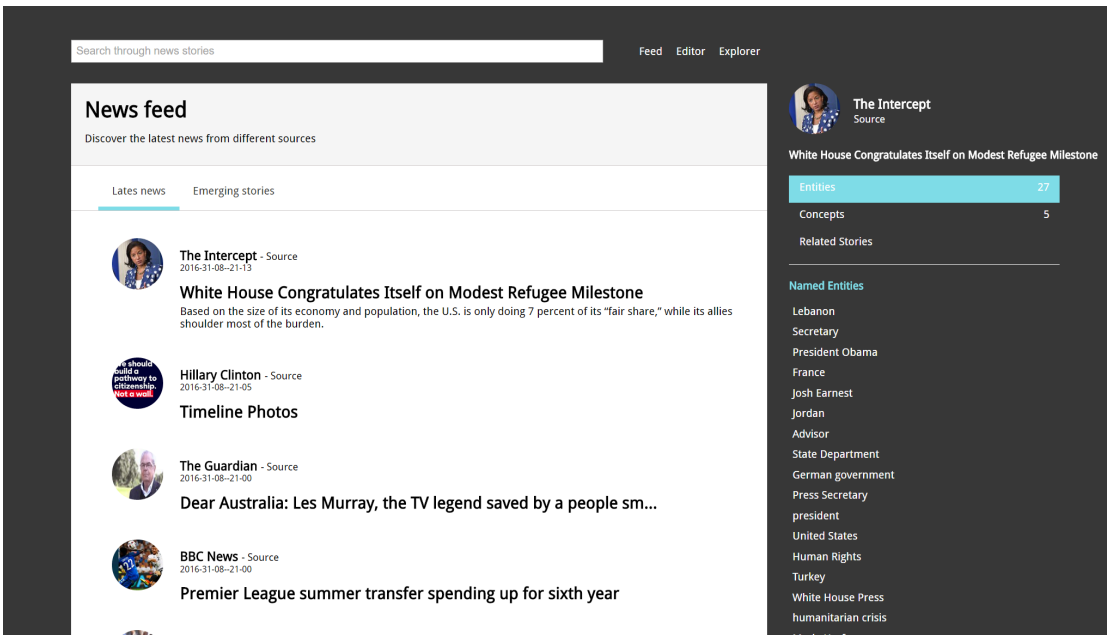

Figure 4.7: Front-end news feed with the ability to inspect stories

After the newly improved upon front-end design was implemented with HTML and CSS, the new functionality could be added. Figure 4.7 shows the new front-end featuring news articles from different sources that have been gathered by the web crawler, but also including one of the Facebook posts from Hillary Clinton's Facebook page.

4.4.8 Iteration 8: Multilingual translation services

Being able to make the system understand different languages would provide the journalists with more information to work with. Languages are constructed differently. Therefore, the algorithms used for keyword and entity extraction will have problems with the output quality on other languages than the ones they are built for. A solution for this problem is to translate articles into a common language before its analyzed. We chose English as the main language for our system.

Goals

One of the difficulties we faced was analyzing text in different languages. Our goal for this iteration was to explore the possibility of including automatic translation into the pipeline so that we could cover multilingual stories.

1. Translate multilingual text to English.

Tools and technologies

Microsoft Translate API Microsoft offers an API for multilingual machine translation.

Microservices Microservices is an architectural style where the application is divided into services that have their own responsibility and is loosely coupled.

Development process

Some of the services we have implemented support multiple languages, but the amount of supported languages is limited. To handle this problem, we decided to implement translation of the text to English with the Microsoft Translation API before analysis. The reason for translating to English as a main language was because most of the models for NER are tailored towards that language.

In addition, we decided to restructure the architecture as a microservice to make it more scalable and maintainable due to the ever-growing system complexity.

Validation

To validate this iteration, we ran stories written in multiple languages through the translation process, and then fed it to the News Hunter system. We then compared the results with similar news stories written in English. The languages used for the testing were English, Norwegian, and Spanish. If some of the most important named entities and keywords found in the English version of the story, were also found in the in the translated version, the process could be viewed as a success. two news events from distinct categories were chosen. For each of the news events, three news articles were picked out that matched the prerequisite language choices.

We chose two stories to include in our evaluation, each with a different topic. For each of the articles the top 3 highest scoring keywords and named entities have been included in the table for comparison.
The two news events we chose were:

1. Apple unveils red iPhone 4.1

2. Barcelona vs PSG football match 4.2

Results

The first event was the unveiling of a new red colored iPhone from Apple on 21/03/2017. The English language news article was written by The Telegraph, the Norwegian article by DN (Dagens Næringsliv), and the Spanish language article by El Mundo.

Table 4.1: Story: Apple unveiling new red iPhone

Red iPhone	English	Norwegian	Spanish
Keywords	• red apple • red iphone • red product	• apple apple • apple ipad • product red edition	• red iphone • apple watch bracelet • apple product
Named Entities	• Apple (company) • red (color) • Iphone (informationAppliance)	• Apple (company) • GB • Product Red (company)	• Iphone (informationAppliance) • Apple (company) • Red (color)

All the news articles had the Apple company correctly labeled as an important named entity, while only the English and Spanish had the iPhone product, and the color red in the top 3 named entities. The translated Norwegian article had GB (Gigabyte) and a company called Product Red as the other two top entities. For keywords, the English and Spanish article both had red iPhone as an important keyword.

The second event was a football match between Barcelona and PSG where the score ended with 4-0 to Barcelona. The English article was from The Week, Spanish language article from AS, and the Norwegian article from NRK.

Table 4.2: Story: Barcelona - PSG Football match

Barcelona - PSG	English	Norwegian	Spanish
Keywords	Barcelon coach luis enrique, barcelona miracle, barcelona net	Barcelona coach luis enrique, barcelona, coach enrique	Epic goal, second goal, minute suarez
Named Entities	Barcelona (soccerClub), PSG (soccerClub), Champions League	Barcelona (soccerClub), PSG (soccerClub), 5	Camp Nou (stadium), Neymar (soccerPlayer), Barcelona (soccerClub)

The three news articles had Barcelona mentioned as a named entity, the English and Norwegian article also had PSG as an important named entity. All the other important named entities listed is unique for each article. English and Norwegian both had ”Barcelona coach Luiz Enrique” as keywords, none of the other keywords matched.

4.4.9 Iteration 9: News harvesting with RSS

The web scraper approach had several limitations. One of which was the amount of non-news related articles that would end up in the system. This added a level of noise to the news feed that made it more challenging to work with.

Goals

Our goal for this iteration was to increase the consistency of the articles that should be displayed in the news feed. To achieve this goal, we looked into getting more consistent data with the use of RSS streams.

1. Retrieve articles from RSS streams

Tools and technologies

RSS Format that enables users to get updates from web feeds in a computer-readable format.

Feedparser(Library) A Python library that has the capability of parsing feeds, i.e. RSS.

Development process

We started looking for a library we could use for this task that would fit into the existing pipeline. This meant that we had to find a Python library that could be implemented for this task. The Feedparser library was chosen due to its popularity and stability. RSS feed streams were added manually to a list with the addition of a news category attribute.

Validation

We discarded all data from the database that had been retrieved with the previous methods, then the messages gathered from RSS feeds were sent to the system for processing. If the new messages would show up in the front-end the iteration could be considered a success.

1. Check if the results from RSS feed end up in front-end.

Results

The new and improved service for retrieving and parsing data gave a more consistent stream of news articles that had additional information available due to the structure of RSS feeds. The news stories gathered with RSS streams were sent to the front-end successfully and the goal for the iterations has therefore been met. The Newspaper library was still used to download and parse the articles as the RSS stream did not always provide the complete article text content which meant that the parsing limitations of Newspaper were still present.

4.4.10 Iteration 10: The news editor

So far, the system's only concern has been to retrieve and analyze textual data from external sources. This is good for the journalists to read up on current events and make use of the built-in features if they want to go deeper. One of the most important aspects of a journalist's job is to produce stories. Since this is a system aimed at news professionals a writing tool should be included. The text written in the writing tool will be monitored continuously and background information will be presented to the journalist during writing. When a written article is stored in the graph, it will automatically be connected to related stories.

Goals

In the next iteration, we focused on allowing the user to write news stories directly into the user interface and then have it analyzed in real-time. We wanted the analysis to provide a list of named entities and keywords mentioned in the story-in-progress, and provide related stories and up to date background information about the content.

1. Create a text editor in the front-end

2. Analyze the written text and present the journalist with the results

Development process

The first step towards this new feature was to further refine the user interface, and create a text editor where the end user can write news stories. During the writing process, the text can be directly sent to Spacy, it was then analyzed and keywords and named entities is returned. The inspector was changed to fit the text editor page and displays the keywords and named entities that have been automatically picked up. These results were then used to find related content, such as related stories, in the graph based on the named entities. This functionality could also prevent multiple journalists from working on the same story. According to Wolftech, this can be a problem in large news organizations working from separate locations, and in different time zones. Since the system was not implemented in a live environment with multiple computers connected to the same server, this last feature was not implemented further than a front-end visualization of how it could be displayed.

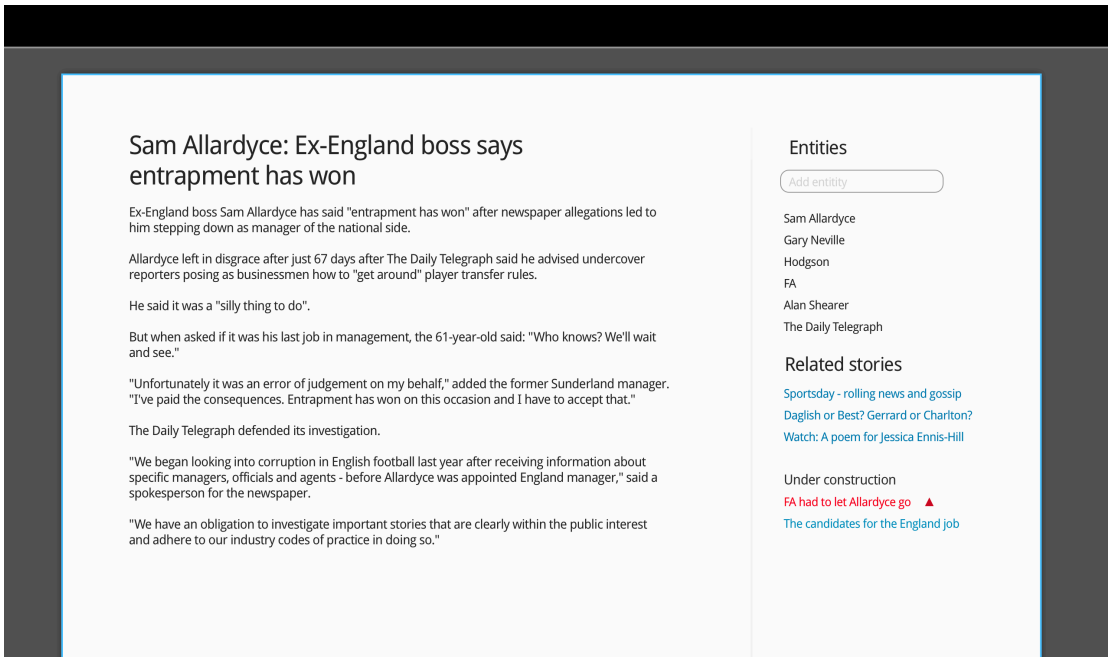

Figure 4.8: News editor concept created in Sketch

Before the editor was implemented it was designed in Sketch. Figure 4.8 shows how it looked when a story was in progress. The main area is where a journalist would do the writing, and the right area would be populated with relevant information. In addition to the automatically extracted named entities, a function for adding missing entities was considered. If multiple journalists were working on the same story, this would be highlighted at the bottom of the right section.

Validation

To find out if the goals had been reached named entities and keywords from the writing tool should show up in the inspect menu.

1. See if named entities and keywords are present in the inspect menu.

Results

At this stage, the end-user was able to write a news story, and have it analyzed and the results presented in the user interface inspector area. When a keyword or named entity is selected in the inspector, the keyword or named entity was highlighted in the text editor.

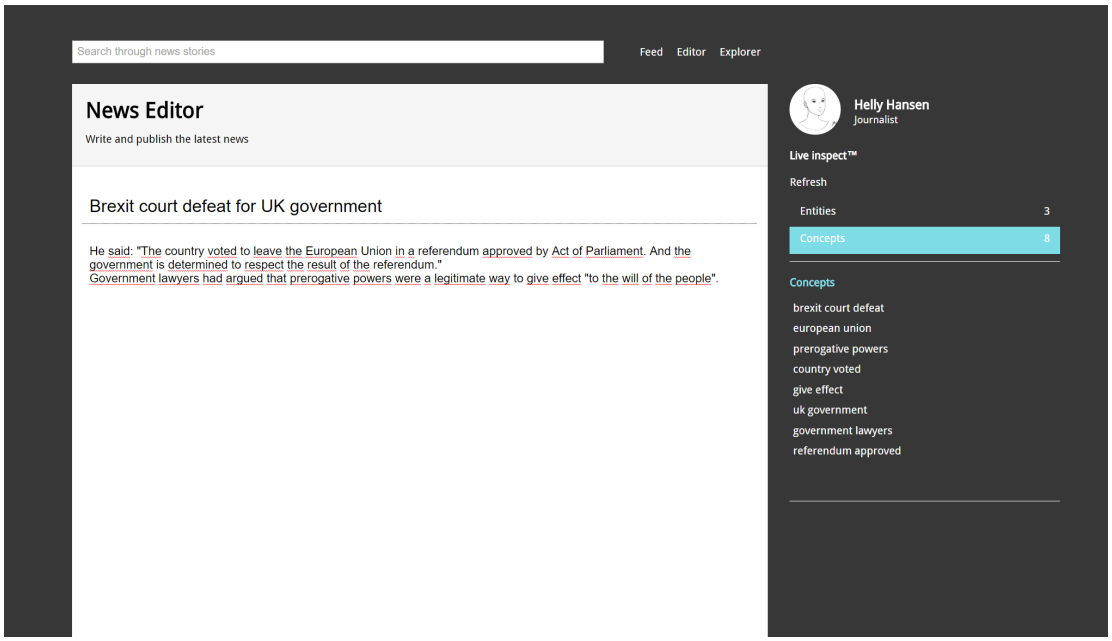

Figure 4.9: News editor with live updated background named entities and keywords

The figure 4.9 shows the first implemented version of the news editor. The main part of the window shows how the headline and body of the text would be written. The right part shows the text after it was analyzed by the microservice responsible for analyzing textual data. The labeled concepts here are keywords gathered from the RAKE algorithm and the 3 named entities found come from the Spacy analysis.

4.4.11 Iteration 11: Newsfeeder

Up to this point it has not been possible to read the full text of an article without following the articles link. Having the text integrated into the application will help the journalist save time, and enable them to faster read up on the incoming articles.

Goals

In this iteration, the focus was to create a component that could receive or retrieve messages and send them to the analyzer service. The goal for this component was to improve the architecture to better conform to a microservice structure. One system limitation was that the full length of messages was not stored in the graph. This new service would tackle this problem and store the full message text in a database so that the end-user could read full articles and compare the results from the analysis with the actual text content.

1. Create a service that can store data

Development process

The first steps towards this goal was to identify which parts of the existing system had to be taken out into a new service. The parts that were taken out were put into a new service called Newsfeeder. This component is responsible for taking in new stories and sending them to the message analyzer API for storing in the graph and further analysis of the content. The front-end would have to be altered to accommodate the additional textual data that each message brings.

Validation

For the iteration to be considered a success it should be possible to read the full text of an article inside of the application.

1. Have the full text of a story available inside of the application

Results

The parts of the system that handled retrieving and receiving of messages were successfully extracted into the Newsfeeder service. The full-length messages were in parallel with the analysis process stored in a SQL database. In the user interface, it was possible to interact with each story in the news feed to get the full text of each article.

4.4.12 Iteration 12: More precise related stories and text summaries

Up to this point the related stories have been calculated from the amount of shared named entities. Longer articles will often have a greater number of named entities included within the text, and that will increase the possibility to encounter related stories. An alternative way to use named entities in search for related stories, could be to only compare the entities that are considered important for each story. This can filter out the non-essential named entities found in the longer stories, and give better grounds for comparison.

After the inclusion of the full text of articles the front-end would sometimes appear to be cluttered. This was due to the length of some articles, in addition to the formatting being

removed. The result was arguably that stories were less readable. To counter this, we looked at implementing automatic text summarization.

Goals

In this iteration, we tried to make some changes to our algorithm for finding related stories in the search for more precise results.
Automatic summarization of text was also a goal for this iteration.

1. Make changes to the related stories algorithm
2. Implement automatic text summarization

Tools and technologies

TextRank Library for automatic keyword and sentence extraction (summarization).

Development process

In order to improve on this, we decided to create a simple ranking system for named entities in text and only look for other stories where the same named entities were considered important. This would help us eliminate some of the problems with articles that would appear as related to too many articles due to it having many named entities.

The ranking system worked by giving a score to each entity based on its order in the text, and the number of times it was mentioned. The basis for this was that many news articles mention the most important aspects of the story early in the text, or in the title, so we could assume those were the most important to the story.

For automatic text summarization, different available libraries were looked at. As we had previously tested TextRank for keyword extraction, this could also be applied for text summarization. The method this library is using is called an extractive approach as explained in section 2.5.3.

Validation

In this context related is a vague and relative term. Two stories could seem related for one user, but not for another. If one story is about a football match in which Brazil plays, and

the other is about foreign politics in Brazil, some could say that they should be considered related since they are both about Brazil. But other than that, they don't share many similarities, and some could argue that they are not related. The ability to look for stories that contain a certain named entity is already included in the application if a named entity is clicked upon.

The goal for this iteration was to make changes to the algorithm that looks for related stories. The problem with the previous algorithm was the overwhelming number of related stories found if the story mentions a common named entity. For the iteration to be considered a success it should reduce the number of relation stories considerably.

To validate the automatic text summarization, longer texts should be inserted into the system, and the summarized version should show up in the front-end.

1. Limit the number of related stories

2. Summarize longer texts

Results

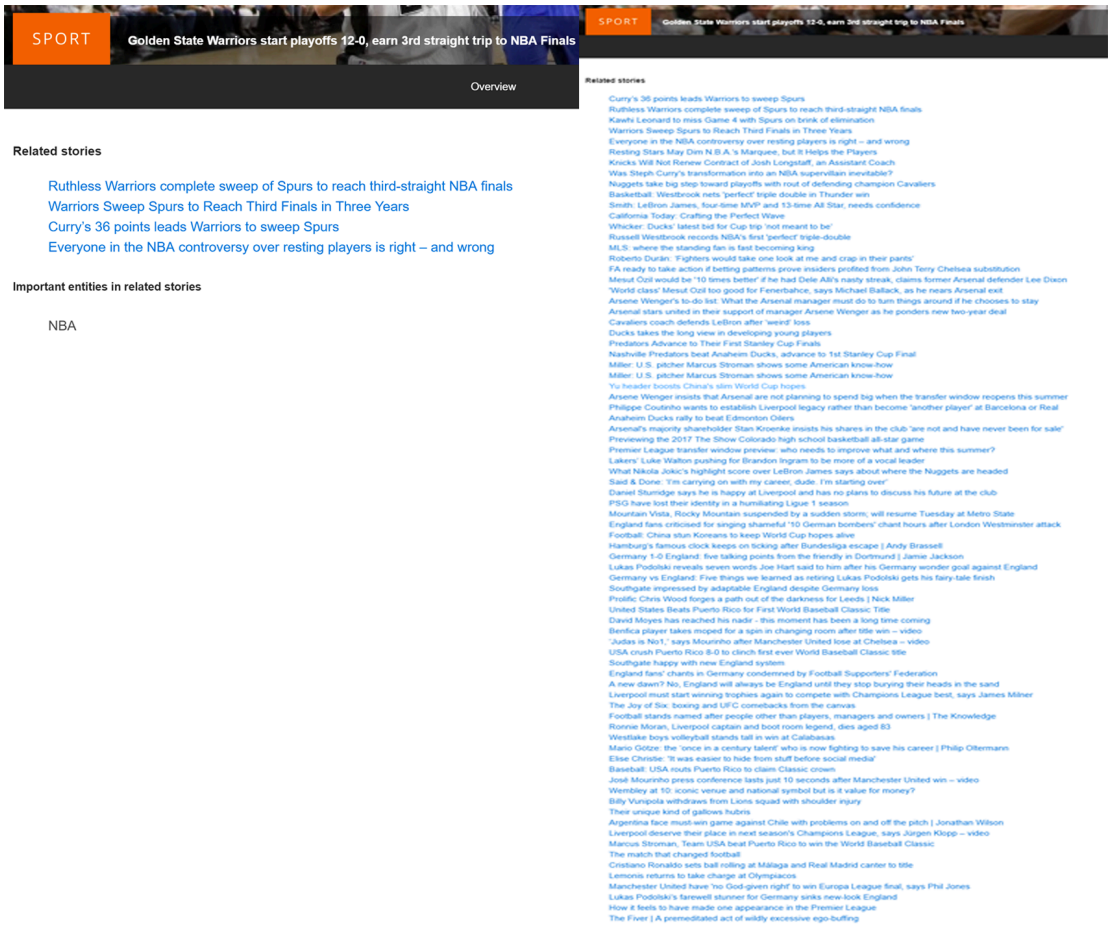

Figure 4.10: After, and before the change in the related stories algorithm

In figure 4.10 it is possible to see the issues that were present when looking for related stories. After the changes, the list has shrunk down to only four related articles. A feature for finding important articles in related stories, that is not included in the original story, is also visible.

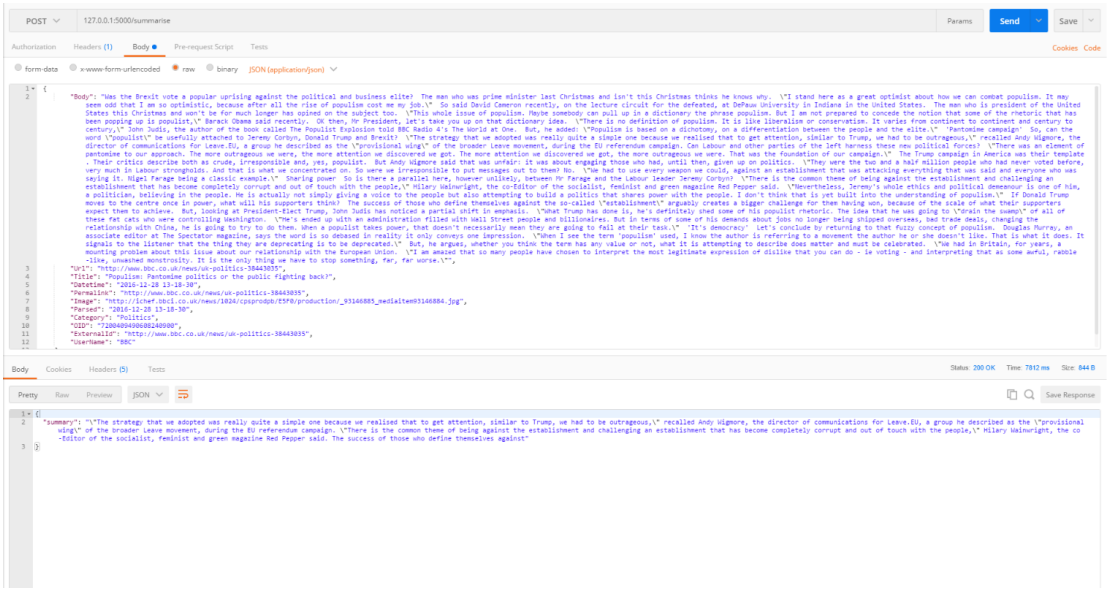

Figure 4.11: Summarization of text

Figure 4.11 is a screenshot of the postman tool used for testing endpoints. An article text (top part of the image) had been sent to the summarization service, and the summary was returned (bottom part of the image).

4.4.13 Iteration 13: News classification

Classification of documents is the process of assigning labels based on its contents. It is common for news agencies to divide news articles into distinct categories, e.g., sports, weather, culture etc. These categories are useful for consumers who are interested in reading about certain topics. For News Hunter, classifying news articles into predefined categories to add more metadata to the graph could aid the system in finding related stories. In addition, it could help journalists do searches and analysis by being selective about which categories to include.

The workflow for building a news classifier was based on earlier work done in the field by Kaur and Bajaj (2016) using well known and tested methods.
The categories were chosen with the BBC Insight dataset as a base, with a couple of added categories to capture a wider variety of streams and news content. The BBC Insight dataset contains 2225 documents from BBC news which are split into 5 different topical categories. The categories or labels are: sport, business, entertainment, politics, and tech (Greene and Cunningham, 2006).

Goals

For this iteration, the goal was to test a simple classification system using RSS data streams and single-label classification methods based on machine learning algorithms.

1. Gather and organize RSS data according to their news category
2. Create a pipeline for classifying news articles

Tools and technologies

Scikit-learn Python library built for machine learning purposes. It contains efficient tools for data mining and data analysis. Classification-, regression- and clustering algorithms are some of the features that are included in the library.

Development process

The starting point was to find RSS feeds that already contained some labeling such as technology, politics, or sports. Streams were gathered and put into a document if the news outlets had a RSS feeds corresponding to one of these categories. For the classification, Scikit-learn was chosen as the library of choice due to its power and simplicity. Collecting from diverse sources allows for the model to train on a wider variety of texts and make better predictions on news articles from various sources.

In addition to these the following categories were also added through RSS feeds due to their prevalence in a variety of news outlets: health, science, environment, and crime. The predefined categories that are defined in the RSS stream metadata from the online newspapers ensures that all the news articles will be put into the correct categories.

Pre-processing of data is important to reduce the dimensionality of data, which in text documents can be high. Cleaning up the text was the first step after the data had been gathered. Each article was sent through Spacy for POS tagging which removed stop words, emails, pronouns, conjugations, and lemmatized the words to their base form. After cleaning the textual input an Scikit-learn pipeline was built. The pipeline consisted of a CountVectorizer for tokenizing and creating a matrix of documents. TF-IDF was implemented for feature selection to assign higher weight to potentially important words in a text. The last step was

to add a classifier to the pipeline. The two chosen classifiers for this task were SVM and MLP.

After the data had been processed and used to train the model, it was ported over to the Flask API and made available for classifying.

Validation

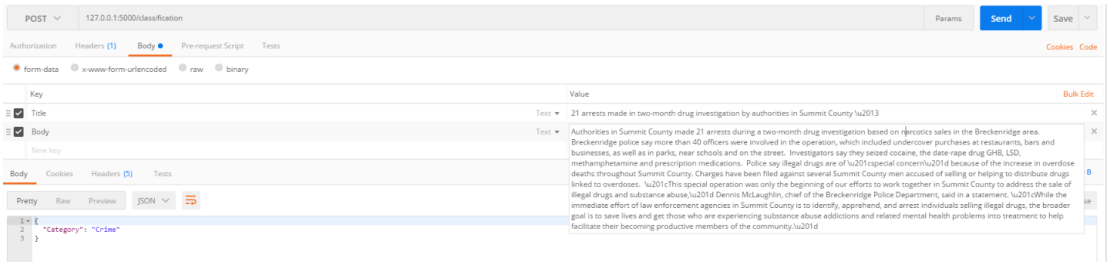

Figure 4.12: Automatic classification of news categories show in Postman

The figure 4.12 shows how the text on the right side of the picture has been classified into the crime category. Further results from the classifier will be posted in the evaluation and results chapter.

4.4.14 Iteration 14: Using Agents and social media

In recent years, social media have flourished, and the content provided by the users varies from trivial things about the author's day, to content that have many similarities with traditional news. The use of smartphones has also rocketed in the past decade, and may be one of the reasons behind the success of modern social media. A large part of the population is equipped with a pocket device capable of taking high quality photos and videos, author a few lines of text, and then just seconds later make it available for the entire world. This capability to instantly share a moment can potentially enable every smartphone user to serve as a news reporter. But this does not mean that professional journalists become obsolete. It provides them with new opportunities; as they gain access to a larger number of sources of information. But for a journalist in pursuit of the next story, it can be difficult to analyze the massive amount of user-generated content. News if often referred to as perishable, so it is crucial to be on the lookout for the next story.

Figuring out what is important in the vast amounts of data form social media can be intimidating. A way of filtering the content, and get potentially newsworthy data could prove useful.

Goals

The goal for this iteration was to filter social media data, and analyze its content. Due to restrictions in data retrieval from sites such as Twitter, Facebook etc. the goal was slightly altered to automatically get data from social media accounts of people mentioned in the news.

1. Automatically detect people from Named Entities

2. Find their social media accounts

3. Query their social media feed

Development process

With the use of the results from DBpedia Spotlight when analyzing named entities, the system was altered to detect when a named entity had the type Person. If an entity received the type of person, another query was made to look for social media handles with the use of the Wikidata service. The entity was then added as an Agent in the graph in addition to its original entity type.

```
                    var twitterHandle = QueryForTwitterHandle(link);

                    if (twitterHandle != null)
                    {
                        var localAgent = new Models.Agent
                        {
                            Homepage = "twitter.com/" + twitterHandle,
                            Label = entity.Label,
                            Source = "twitter",
                            Id = twitterHandle
                        };
                        var lifter = new MessageLifting();
                        lifter.CheckAgent(localAgent, db);
                        entity.Agents.Add(db.Agents.First(c => c.Id.Equals(localAgent.Id)));
                        db.SaveChanges();
                    }
                }
            }
        }
    }
    return true;
}

public static string QueryForTwitterHandle(string entity)
{
    var id = Regex.Split(entity, "entity/")[1];
    SparqlRemoteEndpoint endpoint = new SparqlRemoteEndpoint(uri);
    SparqlResultSet results = endpoint.QueryWithResultSet(
        "  SELECT ?twitter WHERE { " +
        "wd:" + id + " wdt:P2002 ?twitter . " +
        " }");
```

Figure 4.13: Screenshot of the code responsible for assigning Twitter handles to agents

The code sample in figure 4.13 shows the process of querying Wikidata for Twitter handles.

Validation

To validate the efforts, we checked that some of the named entities with the type of person were now also considered an agent, and that they had Twitter handles.

1. Find agents with Twitter handles within the system.

Results

It was possible to find agents with Twitter handles within the system, unfortunately the functionality of including their social media feed when inspecting entities, and a separate feed for social media next to the feed from news outlets were never implemented.

4.4.15 Iteration 15: Clustering of news events

If the patterns show that multiple articles are written about the same event or topic, one could argue that the event may be of some significance and should be brought to the journal-

ist's attention. Some of the sources will often have different views of the event, and provide unique insight, even though the main message is the same, but some of the articles can be very similar. Presenting redundant news stories to the journalist could slow down his or her process. Some of the most popular news aggregator systems, such as Google News and Yahoo News have solved this task in their feed. The articles that are centered around the same event or topic are grouped together. This allows the feed to be less polluted by redundant articles. EventRegistry and NewsMap are two other services that have developed methods for grouping articles together.

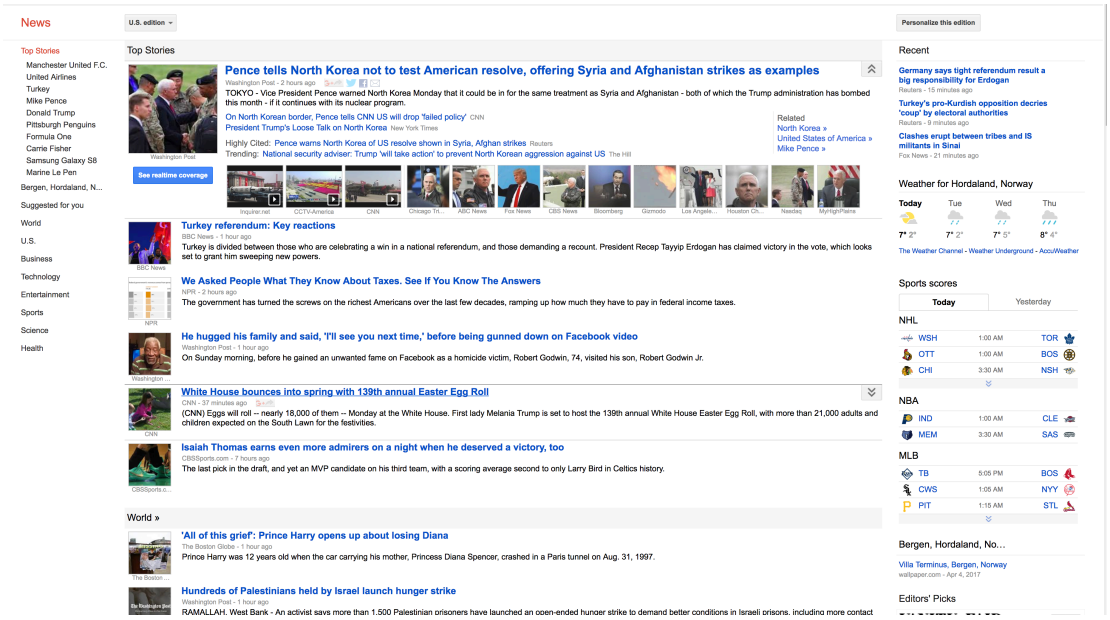

Figure 4.14: Screenshot of the Google News interface.

In figure 4.14 the top aggregated stories are shown in the news feed and in the left side menu. When one of the stories is clicked, the sources reporting the story is shown.

Goals

The goal for this iteration was to explore possibilities for grouping together newsworthy stories, implement a method, and present it in the front-end.

1. Group together stories that describe the same event
2. Add the grouped stories to the graph
3. Build a front-end solution for the aggregated stories

Development process

To reach the goals set for this iteration the development process was divided into three phases. The first phase was to find an algorithm that could group together textual data that seemingly contains much of the same content. We decided to go with Python for the machine learning part, due to its extensive variety of libraries that could fit the problem. One of the libraries that contained many algorithms was Scikit-learn. Previous research suggests that clustering algorithms may be a viable option for tackling this kind of problems. When clustering algorithms are used on a set of documents, the algorithm will try to figure out how similar each document is to every other document. If the algorithm decides that some documents are similar they will be grouped together in a cluster. There are a variety of different clustering algorithms available, in the scikit-learn package alone there are nine different clustering algorithms that each has its own strengths and weaknesses.

Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-Means	number of clusters	Very large `n_samples`, medium `n_clusters` with MiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters	Distances between points
Affinity propagation	damping, sample preference	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	bandwidth	Not scalable with `n_samples`	Many clusters, uneven cluster size, non-flat geometry	Distances between points
Spectral clustering	number of clusters	Medium `n_samples`, small `n_clusters`	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints	Distances between points
Agglomerative clustering	number of clusters, linkage type, distance	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints, non Euclidean distances	Any pairwise distance
DBSCAN	neighborhood size	Very large `n_samples`, medium `n_clusters`	Non-flat geometry, uneven cluster sizes	Distances between nearest points
Gaussian mixtures	many	Not scalable	Flat geometry, good for density estimation	Mahalanobis distances to centers
Birch	branching factor, threshold, optional global clusterer.	Large `n_clusters` and `n_samples`	Large dataset, outlier removal, data reduction.	Euclidean distance between points

Figure 4.15: Screenshot of the comparison of the different cluster algorithms in scikit-learn, taken from Scikit-learn.org.

The table shown in figure 4.15 shows a comparison of the different algorithms available in the scikit-learn library.

When working with news data, there will be an unknown number of documents that the algorithms must go through. The content will in most cases also vary over time. Having good scalability is important to meet the required demands, the algorithm must work on small

datasets, as well as large datasets. This alone will eliminate a lot of clustering algorithms. Clusters will vary in size, and in most cases, it will be hard to predict the number of clusters that are appropriate for the dataset. One widely used clustering algorithm is K-Means. The algorithm is described in detail in section 2.4.1. It is often used for document clustering, but with the variety of textual data in news it has too many limitations. When the algorithm is initialized the number of clusters must be set. Therefore, the choice fell on the DBSCAN algorithm which matched all of the criteria. The algorithm is described in detail in section 2.4.1.

To run data through the DBSCAN algorithm, new functionality in the .NET back-end had to be built to send and retrieve results. When called upon, it sent news stories from x days to the Python endpoint. The textual data was preprocessed and turned into tf-idf as explained in section2.5.5. The results did then go through the DBSCAN algorithm. The DBSCAN algorithm was built to return an array containing the different clusters, x-1, x0, x..n. X-1 contained the stories that did not fit the parameters required to be included in a cluster. These outliers were removed before the array was returned to the .NET module.

When the .NET back-end received the results from the clustering process it stored the results in the graph.

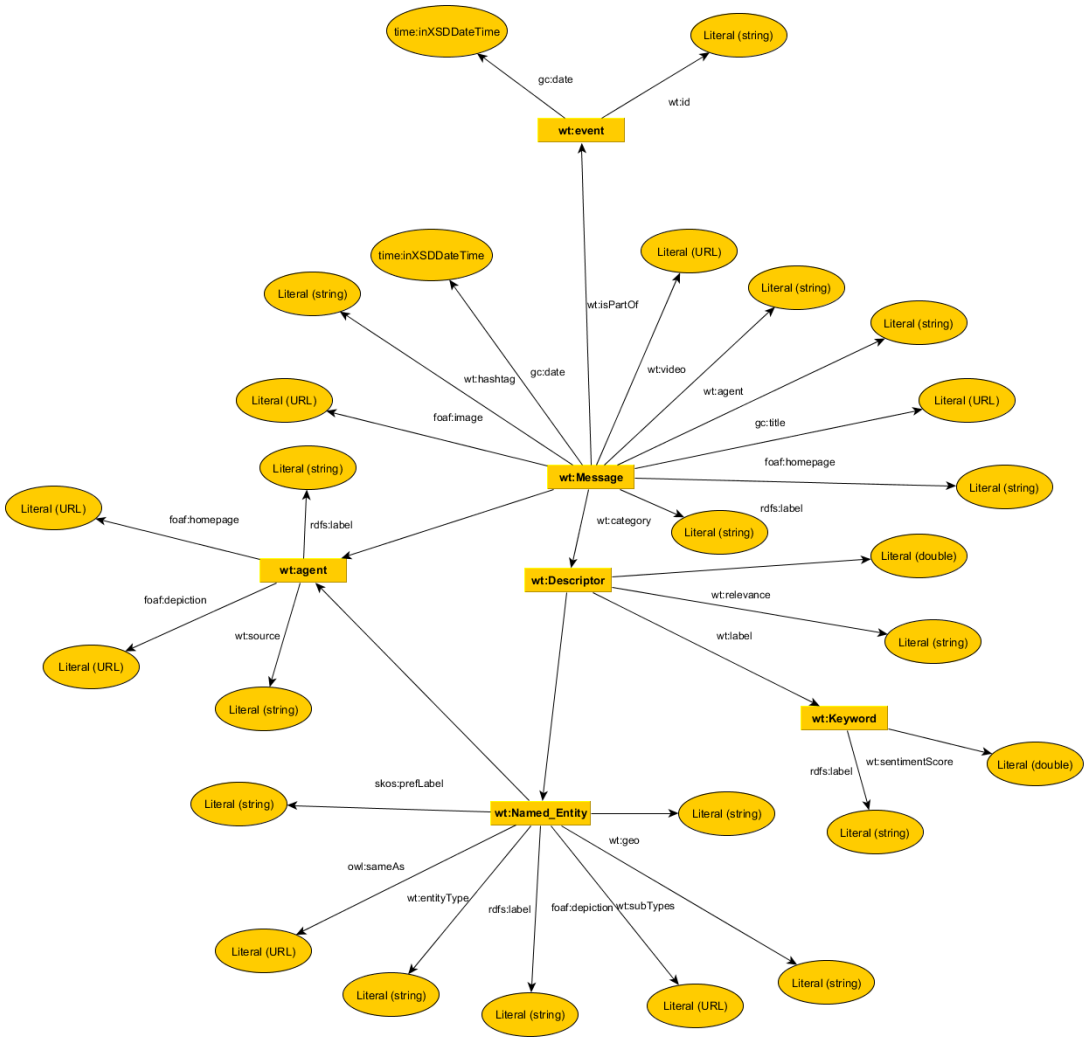

Figure 4.16: The next version of the News Hunter ontology

In figure 4.16 the graph has evolved to include an event. One event can include messages, and messages can be part of an event.

Alterations to the front-end were made to get this feature included. In the news feed, a possibility for finding top stories was added. When this button was clicked on, the most recent group of clusters was gathered from the graph database. Each cluster was shown as a single story. When a cluster was inspected, the additional data was on a cluster level, as opposed to the document level data that was shown when a single story was inspected.

Validation

To test that the feature works as intended the results were compared to news aggregators available for the public. The clusters were also inspected to validate that each cluster was

clustered correctly.

1. Inspect clusters and see if they contain documents about the same event.

2. Compare the results with Google News and Event Registry to see if the clusters are shown as a top story in their system.

Results

The clustering algorithm analyzed 1292 articles from a variety of newspapers gathered with our Newsfeeder system. The articles were in these categories: Business, entertainment, politics, sport, technology.

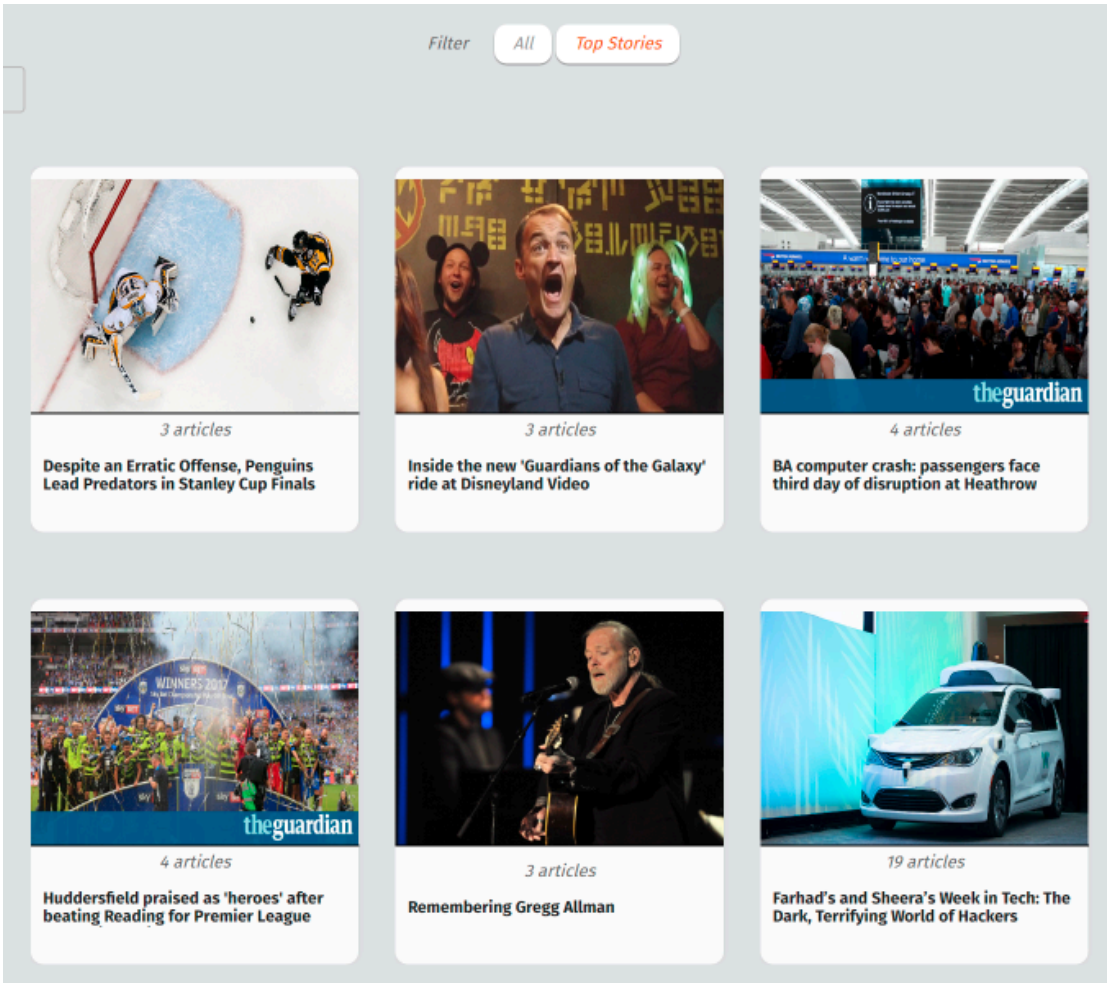

Figure 4.17: Screenshot of the clusters found.

Figure 4.17 shows the six clusters found. Four of the clusters contains articles about the same event, while the other two are clustered incorrectly.

Figure 4.18: Screenshot of a correctly labeled cluster.

In 4.18 all stories are about a IT crash that have affected British Airways passengers.

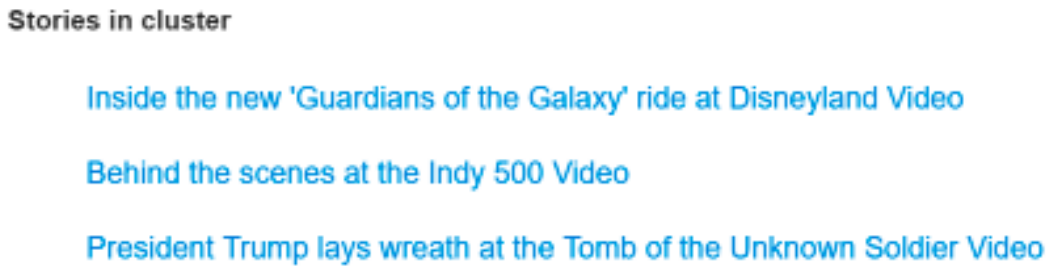

Figure 4.19: Screenshot of a cluster that is wrong.

In 4.19 all stories originate from abc news and they are videos, not articles.

4.4.16 Iteration 16: Multi-label news classification

Following the work done on the previous classifier a more general and standardized way of categorizing was required that would also allow multiple labels or topics to be captured. The International Press Telecommunication Council (IPTC) news codes are part of an architecture to model metadata information for the news industry (Troncy, 2008; IPTC, c). The shift toward a multi-label and hierarchical structure should be able to better represent the content of an article and be more useful for journalists.

IPTC is also attempting to solve this news classification problem through their IPTC Extra rule-based classifier (IPTC, a). This will be an open source multilingual project that will allow newsrooms to tag content automatically with IPTC Media Topics(IPTC, b).

Goals

News classification using IPTC newscodes and data set from The Guardian.

1. Gather data from The Guardian API
2. Map Guardian tags to IPTC equivalents
3. Create pipeline for classifying news articles

Tools and technologies

Keras Keras is a Python library for high-level neural networks and deep learning using Theano or TensorFlow as a backend.

Development process

The first part of this iteration was finding a suitable news dataset to use as a basis for multi-label classification. For this task, The Guardian was chosen as a basis for creating a dataset due to its availability through an API with a free developer key for non-commercial application and having a tagging system that had at least some categories that were equivalent to IPTC news codes categories.

The tags labeled by The Guardian was then used to search for matches in the IPTC newscodes(IPTC, d). In addition, the use of Wordnet synsets was applied to each of the categories to see if there were more matches found in the synsets. While the process was meant to be fully automatic without human intervention, to ensure that there was some level of quality during testing all the label mappings were looked through manually afterwards. Some of the mappings were too broad i.e. The Guardian technology tag containing subcategories that spanned multiple different top-level newscodes. Due to this problem with higher level tags being too broad, most of the chosen tags were specific to lower level categories. The hierarchical nature of the IPTC structure allow for very specific topics to be captured on the lower levels of the hierarchy. For the experiment of this model, only the top categories will be used due to the complexity and the numerous categories there are on the lower levels.

Guardian tag	IPTC mapping	IPTC top level mapping
politics/localelections	local elections	Politics
world/capitalpunishment	capital punishment	Crime, law and justice
world/earthquakes	earthquake	Environment
sport/triathlon	triathlon	Sports

After the mapping had enough tags for training a classifier, the files were converted to a standardized format with the top-level IPTC tags present. In addition to using Scikit-learn for building a classifier, Keras was also chosen as it presented the options of using deep neural networks in addition to SVM in Scikit-learn. As with the last classifier, the documents were run through the same pipeline as in section 4.4.13, with modifications to allow multi-label classifications.

Validation

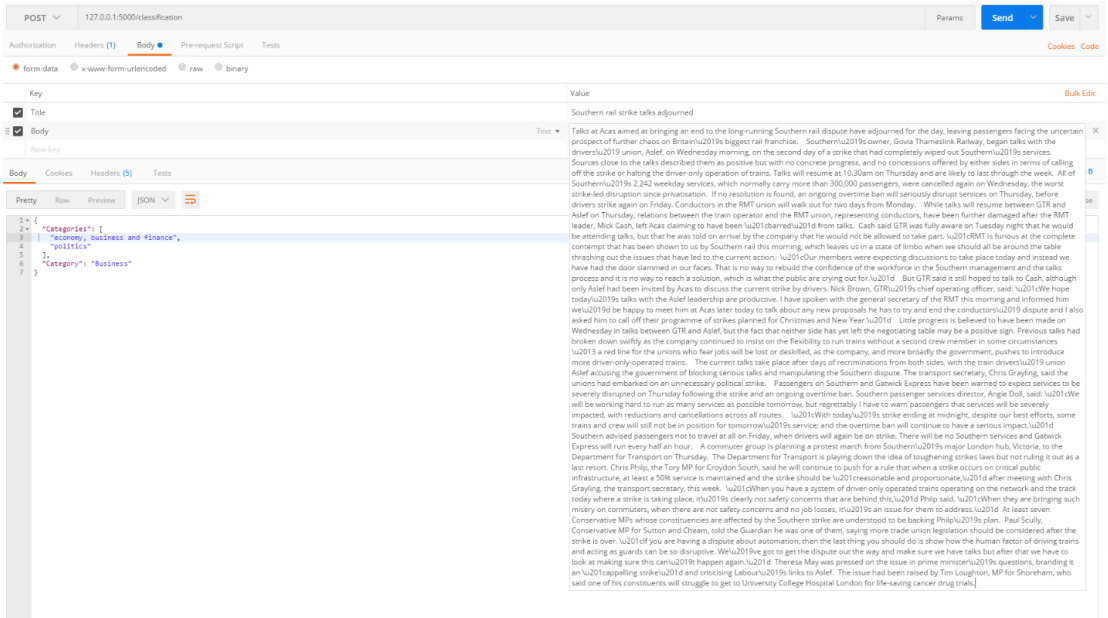

Figure 4.20: Postman showing text being categorized using two different classifiers.

The figure 4.20 shows the results from both the multi-label and single-label classification. While the single-label classifier guessed it to be in business, the multi-label approach picked both business and politics. Further results will be included in the evaluation and results chapter.

4.4.17 Iteration 17: Front-end optimization

In most of the iterations new features were added, or existing features were improved upon and/or changed. We had issues fitting all the functionality into the user interface used up to this point. especially the functionality dealing with gathering of external data when interacting with named entities. To deal with this problem we decided to make a new user interface with more space for our features.

Goals

Having an user interface that is easy to use is important, the Technology Acceptance Model 2.8 it describes how ease of use can have an impact on the usefulness of a system. The goal for this iteration was to create an easy to use interface that can fit the functionality.

1. Design a user interface that can fit all the functionality
2. Implement the design

Development process

Before the interface was designed in Sketch, the core features were listed to make sure that they were included. The problem with the previous design was that all the functionality when inspecting a story was cramped in a column on the right side of the screen. More space was needed to better make use of the functionality, so it was decided that having a modal window that can overlay the application could provide the space needed. Putting all the showcased functionality into a modal window made the sidebar excessive, the whole window was therefore redesigned to make better use of the space. The feed no came in three columns instead of just one.

Validation

To validate the work done in this iteration we could see if the functionality was still intact after the transition into the newly implemented user interface.

1. Front-end has new design
2. The functionality is intact

Results

The user-interface was successfully implemented, and the functionality were still intact after the transition. Some screenshots of how the final user-interface looked like is presented below.

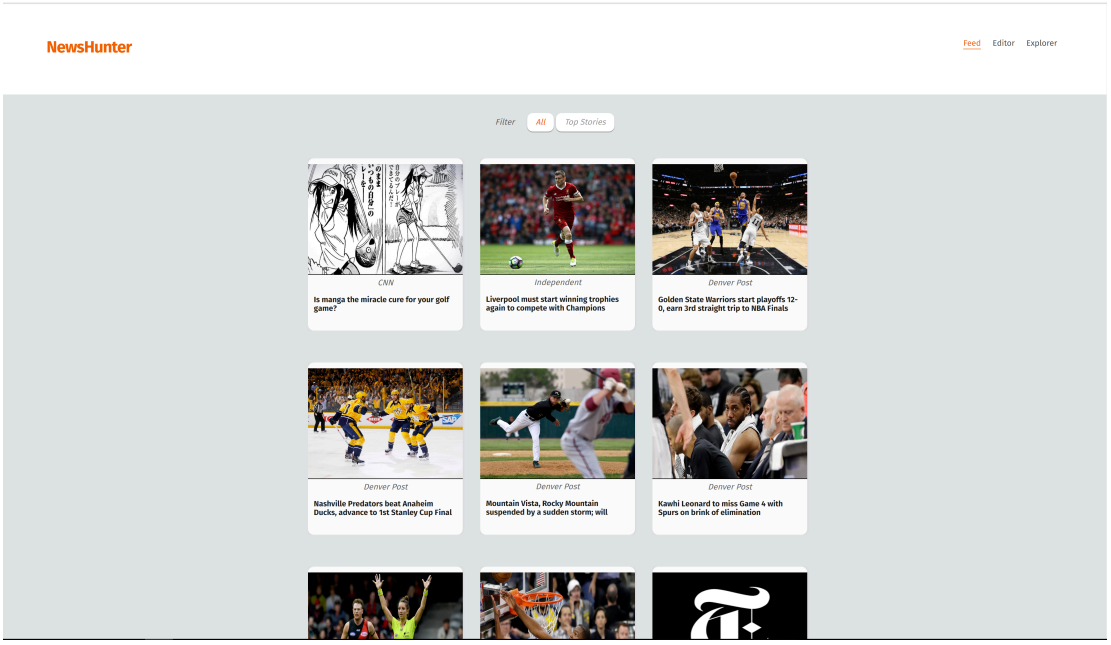

Figure 4.21: Screenshot of the News Hunter feed view

Figure 4.21 shows how the news feed looked like in the final version of the user interface. The view had been simplified with less menus. and the feed now came in multiple columns to make better use of the space.

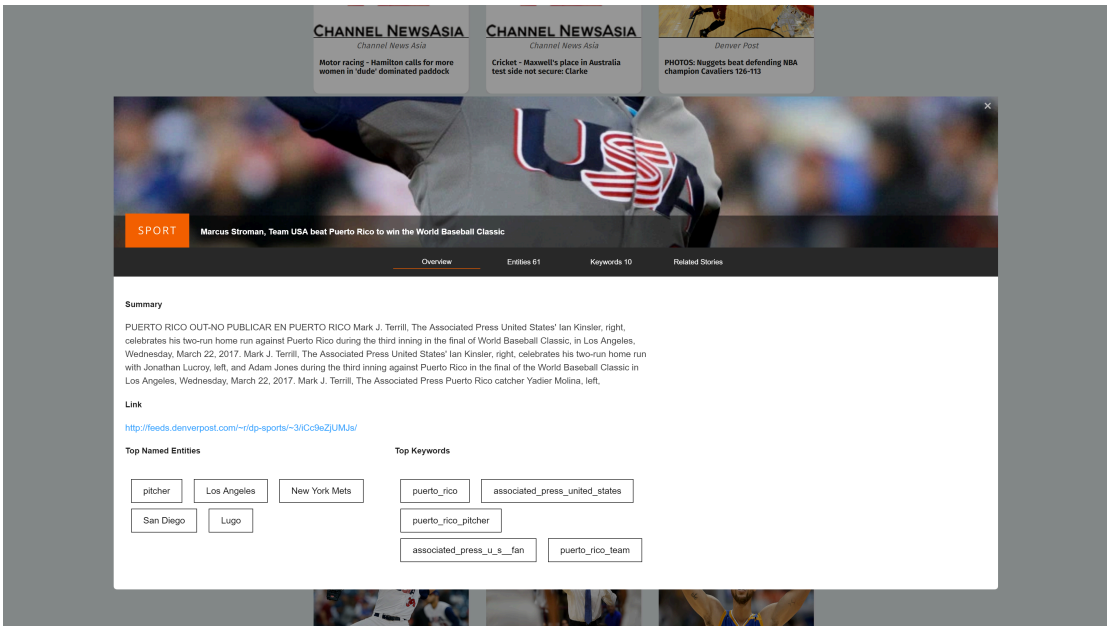

Figure 4.22: Screenshot of the news inspector

When one of the articles in the feed view was clicked upon, the user was presented with an overview of the story, as shown in figure 4.22. In the overview, the user was presented with the article category, the title, a summary of the article text, a link to the original source and the top named entities and keywords found in the article. The top entities and keywords were chosen via their relevance score. In the empty white space on the right side we wanted to include word-cloud from topic modeling and/or an interactive map.

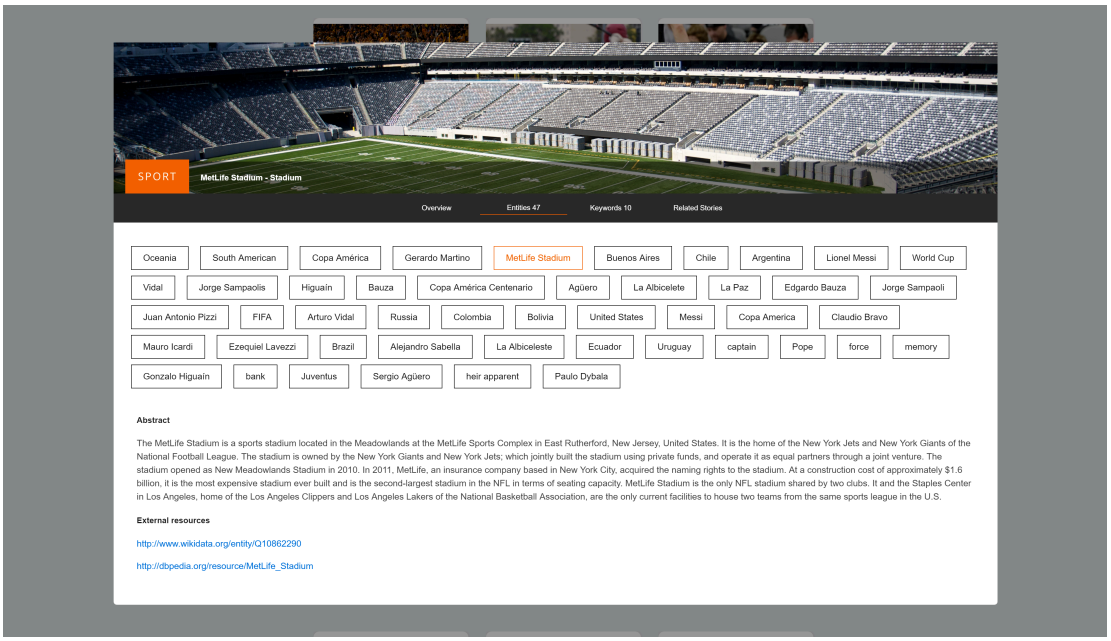

Figure 4.23: Screenshot of the entity inspector

Named entities are widely used within our system for multiple purposes. One of the features we found interesting was the gathering of external data with the use of named entities. In figure 4.23 a named entity is selected, ”MetLife Stadium”. When a named entity was selected, the entity type would present itself, for ”MetLife Stadium” the type of the entity was stadium. The top image was also gathered from external resources, as well as the text describing the entity, and the links to external resources. The text was gathered from DBpedia.

Chapter 5

Evaluation and Results

In design science research, it is important to evaluate an artifact using rigorous evaluation methods.
In this chapter, we will present the evaluations we conducted on different parts of the system.

5.1 Objectives

The main goal of the evaluations was to find out how useful the system was for journalists, and get an estimation of how good the output was. In addition to getting qualitative answers to these questions from journalists and domain experts, we also tested some of the core features with quantitative methods.

5.2 User evaluations

5.2.1 Protocol

The user evaluations were done by combining demonstration, observation, questions, and assessing output quality.

Participants

Journalists and domain experts who work daily with the development of tools for newsrooms were included in the evaluation. The criteria for choosing participants was that they had professional experience as journalists, or were familiar with tools and methods within the domain. In total, there were six people that took part of the user evaluations. Two of them

with background as journalists, and four of them can be considered domain expert due to their extensive experience with newsroom systems.

Introduction, demonstration and observation

During user evaluations, each session started with a brief introduction where the purpose of the evaluation was explained. The participants were then given a short demonstration of the News Hunter system, before they got to try the system for themselves. They were instructed to click on different news articles and try the different features incorporated in the system. After they had familiarized themselves with a feature, they were asked questions about the feature's usability.

Questions

The questions used during the interview were inspired or taken directly from TAM and TAM 2. What we wanted to find out with these questions was whether the information and possible use cases of the system presented felt useful, and which features the participants could find interesting. The question we used can be found in appendix A.

Output quality

Testing the output quality of the system in a qualitative way is important to see which of the implemented features that could produce high quality results that were useful, and which needed refinement.
Named entity recognition quality was assessed by looking at the outputs which had been identified by Spotlight and Spacy and seeing if they had made any obvious faults or entirely missed some named entities. After those tasks had been completed they were asked whether the categories presented were accurate of the content.

5.2.2 Interview results

Inspecting messages

What is useful about this feature? All participants found the feature useful for getting an overview of a news story.

What is missing from or problematic with this feature? One of the participants would like an option to skip the inspect menu to go straight to an articles original source. The second participant would also like to have the option to see if the story has been updated by the source, and then also get the update history. Participant 3 would like related stories to be displayed more prominently than named entities and keywords. One of the domain experts would like the top named entities in the overview section to be displayed as a word-cloud instead of a list.

Is the feature useful? All of them found the feature to be useful.

Named entities

What is useful about this feature? Five of the participants found the feature helpful for getting external data. Two of the participants noted that getting all the information directly in the application was better than having to do multiple searches on the web. The second participant added that each entity button represented one tab in a web-browser. Using the feature to get more context was helpful for understanding a story.

What is missing from or problematic with this feature? The first participant found the term ”named entities” to be confusing initially. The feature could benefit from having more filtering and information from additional sources. The second participant found that some entities that were present in the articles were missing. The entity images could scale better in the user-interface. The third participant wanted the external data to come from verified sources. Two of the domain experts would like to have entities that represent locations to be displayed on a map. Having the option to sort entities by type in addition to relevance was also requested by two of the domain experts. One of them wanted to have the type represented visually next to the entity-name.

Is the feature useful? Both journalists found the feature to be useful. Three of the domain experts found the feature to be useful, the last one would find it useful given more reliable data sources.

Keywords

What is useful about this feature? One of the journalists, and one of the domain experts said that it was an efficient way to save time when figuring out what a story is about.

Two of the participants, one journalists and one domain expert, thought of using the keywords for meta-tagging and for searching related content.

What is missing from or problematic with this feature? Five of the participants noted that the quality of the keywords could be better. The second participant would like an option to remove faulty keywords. One of the domain experts noted that word disambiguation could be a problem. Another noted that keywords could include high-level topics.

Is the feature useful? Both journalists found the feature to be useful. The domain experts were however divided, two of them found it useful, two of them did not.

Summaries

What is useful about this feature? One of the journalists would like to have the feature implemented in their system, to get condensed stories that they could be published regularly. This would enable them to publish content more frequently. Two of the domain experts noted that it was an effective way of getting a quick overview and that it was potentially time-saving when figuring out the main content of a story.

What is missing from or problematic with this feature? The first participant found the quality of the summary to be faulty. Some sentences did not make any sense, and some were cut short. The second participant found no obvious relation between the headline and the summary in some cases.

Is the feature useful? The participants were divided on the usability. Only the second and fourth participant found the feature to be useful. Participant 1 and 3 would rather have the title and lead paragraph of a story to be shown instead of a summary. The others were unsure about the usefulness.

Top stories

What is useful about this feature? Two of the participants saw the use for this feature if there were multiple articles about the same event. Four of the participants stated that the feature could be used for finding out what is happening and what needs to be written about.

What is missing from or problematic with this feature? The first participant said that if the top story inspector shows named entities and keywords about the whole group, only the ones that are shared among all articles should be displayed and there should not be too many of them. The second participant wanted to be able to search through groups to find specific articles that mention certain things. When inspecting a group, it could be beneficial to have the possibility of excluding certain sources. Being able to see why articles are grouped together would be a welcoming feature. Two of the participants said that it could be problematic if the feature takes too much focus. Smaller, but also important, stories could easily be forgotten. One of the domain experts would like a slider for finding top-stories within a certain time-frame. Two of the participants said that categories could help each top story give more information, better results and could be used for filtering.

Is the feature useful? All participants found the feature to be useful, but they all wanted additional functionality.

Related stories

What is useful about this feature? All the participants found the feature to be useful when looking for similar news stories.

What is missing from or problematic with this feature? Two of the participants would like additional details, like i.e. date and source for each related story. The second participant noted that the related story algorithm could be problematic if two words had separate meanings. I.e., the word ”Oscar”, could be the name of a person, or the name of the famous statuette given to prize winning actors. Two of the domain experts noted that having a feature to see if anybody is working on a new story about the message could be useful in large newsrooms.

Is the feature useful? Yes, all the participants found the feature to be useful.

Editor

What is useful about this feature? The first participant noted that getting keywords automatically from the text was useful. Three of the participants stated that the entities and keywords could be used as meta-data and tagging.

What is missing from or problematic with this feature? All the participants stated that the editor could have more features. Getting related information when writing, and being able to publish the article with related stories was a feature most the participants wanted. The second participant noted that the use of search engines to find earlier articles were often used in the workflow, thus getting earlier articles directly to the editor would be beneficial, this was also a feature three of the domain experts wanted. Having the possibility to remove unwanted keywords and named entities was also requested by three of the participants. Getting warning when multiple journalists are writing about the same story was a feature requested by two of the domain experts.

Is the feature useful? All the participants said that the editor could be useful if more features were included.

5.2.3 Named entity quality results

We tested the output of the named entities in our system by giving the users a list of which entities were found and what types of entity they were. Users were then given time to look over the list and assess whether the named entities were correct. Unlike the named entity view in News Hunter, the list only contained named entities that had background information available. This was done to limit the amount of entities for the participants to look through.

Article: 1

The first article was a story from Reuters titled: ”Trump gives nod to Republican tax-credit proposal on Obamacare”. This article had 11 named entities listed by entity extraction process. Tom Miller (Heroes character) was picked up on. White House entity was confusing to some, as it pointed to the presidency and not the building. One noted that health insurance should not have been included as an entity.
Overall the users had issues with 3 out of the 11 named entities listed. They had no problems with 73% of the entities.

Article: 2

The second article was a story from BBC titled: ”UKIP burka ban policy ’misguided’ says party’s MEP”. This article had 15 named entities retrieved by the named entity algorithm.

Commonwealth and the EU were marked as countries, which most of the users argued was incorrect. Someone also noticed that the June General Election pointed to a Greek election instead of the British one.
Overall the users had issues with 3 out of the 15 named entities listed. They had no problems with 80% of the entities.

Article: 3

The third article was a story from CNN titled: ”Paul Pogba: Is he worth $120 million?”. This article had 27 named entities listed. All the users noticed that Player ROI as a person was incorrectly labeled. Some of them also noticed that ”current tv” should not have been marked as an entity since it was not referring to the TV station. Many were also skeptical or unsure about Awash River as an entity in this context.
Overall the users had issues with 3 out of the 27 named entities listed. They were satisfied with 89% of the entities.

Notes

Some users felt that the entity types should have been more specific overall, such as having soccer players marked as type soccer player and not just person.

5.2.4 Keyword quality results

The keyword quality was tested by having the interview subjects read through medium length news articles. The articles can be found in appendix B. After reading the articles they were asked to come up with suggestions of keywords or phrases that would describe the contents of the article. After they had provided their keywords they were presented with two lists of automatically generated keywords or key-phrases produced by the system. The participants were then prompted to choose the list they preferred. TextRank and RAKE were the algorithms chosen for keyword extraction.

The results from the preferred algorithm and the user provided keywords were calculated using precision, recall and F1. For the calculation, true positives were keywords that were found in both the algorithm keyword set, and the keyword set provided by the user. The false

positives were the keywords found in the algorithmic keyword set, but not in the user provided set. The false negatives were the words provided by the user, that had not been picked up by the algorithms.

Article: 1

The first article was a story from Reuters titled: ”Trump gives nod to Republican tax-credit proposal on Obamacare”.

Algorithmic keywords	
Algorithm	Keyword
TextRank	house republican plan, house republican proposal, health insurance, medicaid coverage, republican tax, credit proposal, tax credit, trump, health,state Governor.
RAKE	democratic president barrack obamas signature 2010 healthcare law, president donald trump backed, age-based monthly tax credit, expanded health savings accounts, reconcile house republican plans, government health insurance program, people purchase health insurance, draft republican replacement,congressional republican ranks, american enterprise institute.

In the table above the results from the automatically generated keywords extraction algorithms are shown. Of the participants in this study three of them preferred the keywords provided by the TextRank algorithm. One of them preferred the results from RAKE. The others could not decide, they preferred TextRank for search and generality, and RAKE for context.

User provided keywords	
Participants	Keyword
All participants	- Donald Trump (5). - Obamacare (5) . - Tax credit (5). - Health insurance (3). - Medicaid (3). - Barrack Obama (2). - Between state (2). - Republican (1). - Democrat (1). - Medicare (1). - Poor people (1). - Congress (1).

The keywords provided by the users are displayed in the table above. It shows the normalized version of each keyword and how many users that used it. The names of persons were normalized to their full name.

Since the participants in the study were asked to provide keywords, their answers were incompatible with the output from RAKE, which often produces key-phrases, rather than keywords. The TextRank keywords were also the preferred algorithm by the participants and was therefore compared with the set provided by users.

Before the two sets were compared, the keywords mention by only one person were removed. If the keyword consisted of multiple words, 50% had to be correct for it to be a match. This means that i.e. ”Donald Trump” and ”trump” was considered a match. If there were multiple partial matches or it was a full match, it only counted as one match.

$$P = \frac{4}{4+6} = .4 \tag{5.1}$$

$$R = \frac{4}{4+3} = .571 \tag{5.2}$$

$$F1 = 2\,\frac{0.4 \times 0.57}{0.4 + 0.57} = .470 \tag{5.3}$$

Article: 2

The second article was a story from BBC titled: ”UKIP burka ban policy ’misguided’ says party’s MEP”.

Algorithmic keywords	
Algorithm	Keyword
TextRank	ukip burka ban policy, ukip leader, mr nuttall, muslim leadership, mr whittle, mr carver, ukip, british people, british muslims, foreign affairs spokesman.
Rake	- mr whittle told bbc radio 4s today programme ukip, device media caption ukip leader paul nuttall, ukip deputy leader peter whittle defended, britains secretary general harun khan condemned, full-face veils hinder integration mr carver, mr nuttalls so-called integration strategy, west midlands mep james carver, leader paul nuttalls plans, ukip donor arron banks, mr nuttall sparked controversy.

After evaluation the results from this article, we found that they were consistent with the results from article 1. Of the participants three of them preferred the keywords provided by TextRank. One of them preferred the results from RAKE. The others could not decide, they are useful for different purposes.

User provided keywords	
Participants	Keyword
All participants	- Burka ban (4). - Paul Nuttall (3) . - Muslim (3). - Brexit (2). - UKIP (2). - Female genital mutilation (2). - Britain (1). - British (1). - Liberty (1). - Cultural (1). - Face cover (1). - James Carver (1). - EU (1). - BBC (1).

$$P = \frac{4}{4+6} = .4 \tag{5.4}$$

$$R = \frac{4}{4+2} = .666 \tag{5.5}$$

$$F1 = 2\,\frac{0.4 \times 0.57}{0.4 + 0.57} = .498 \tag{5.6}$$

Article: 3

The third article was a story from CNN titled: ”Paul Pogba: Is he worth $120 million?”.

Algorithmic keywords	
Algorithm	Keyword
TextRank	manchester united player, premier league player, manchester united midfielder, premier league goal, english premier league, premier league interception, player roi, manchester united, premier league, investment united.
Rake	man united midfield partners juan mata, current tv deal reportedly worth, live on-field performance indicators, manchester united defenders chris smalling, made 24 premier league interceptions, manchester united midfielder paul pogba, winning 41 aerial challenges, performed 39 successful dribbles, twenty summer signings included, interceptions, player roi

The results from the third article was also in favor of TextRank. Four of the participants said that they preferred it to RAKE. The others were indecisive for the same reasons as article 1 and 2.

User provided keywords	
Participants	Keyword
All participants	- Player ROI (5). - Manchester United (4). - Paul Pogba (4). - Premier League (4). - Juventus (1). - Successful (1). - Assists (1). - Passes (1). - Dribbles (1). - Worth (1). - Football (1).

$$P = \frac{4}{4+6} = .3 \tag{5.7}$$

$$R = \frac{3}{3+1} = .75 \tag{5.8}$$

$$F1 = 2\,\frac{0.3 \times 0.75}{0.3 + 0.75} = .428 \tag{5.9}$$

5.2.5 Category quality results

Each of the articles presented to the users during the named entity and keyword tests also showed a predicted news category from both the single- and multi-label classifiers. Unfortunately, only one category prediction was given by the multi-label classifier. This is likely due to the lengths of the articles, as longer news articles contain more features for the classifier to use.
All the users agreed that while the categories were correct, they were too general and having more precise and descriptive categories would be better.

5.3 Algorithm evaluations

In this section, we will look at quantitative evaluations done on some of the implemented features.

5.3.1 News classification

The news article classifiers are split into two main groups. There is the multi class classifier built using data gathered through RSS streams as explained in section 4.4.13. The second set of classifiers were multi-label using data from The Guardian with mappings between their tagging system and IPTC standards.

Datasets

The single label RSS dataset used for the single label classification was gathered between December 2016 to May 2017.

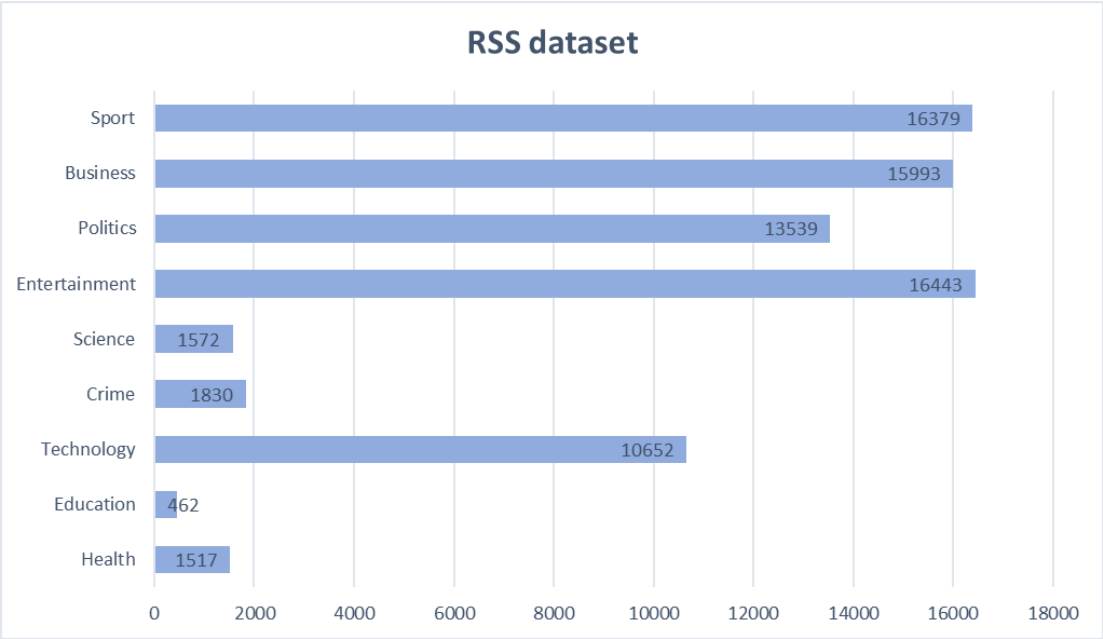

Figure 5.1: The distribution of news articles in the automatically created dataset based on RSS

The dataset that was used to train the single-label classifiers is shown in and ended up being unbalanced. 5 of the categories ended up having a way more samples than the other 4.

The multi-label set was created using data from The Guardian by retrieving it from their API. This meant that unlike the RSS dataset, there was less noise leftover from the parsing process in the set. 544 820 articles were downloaded with the start date set at January 1.st 2010.

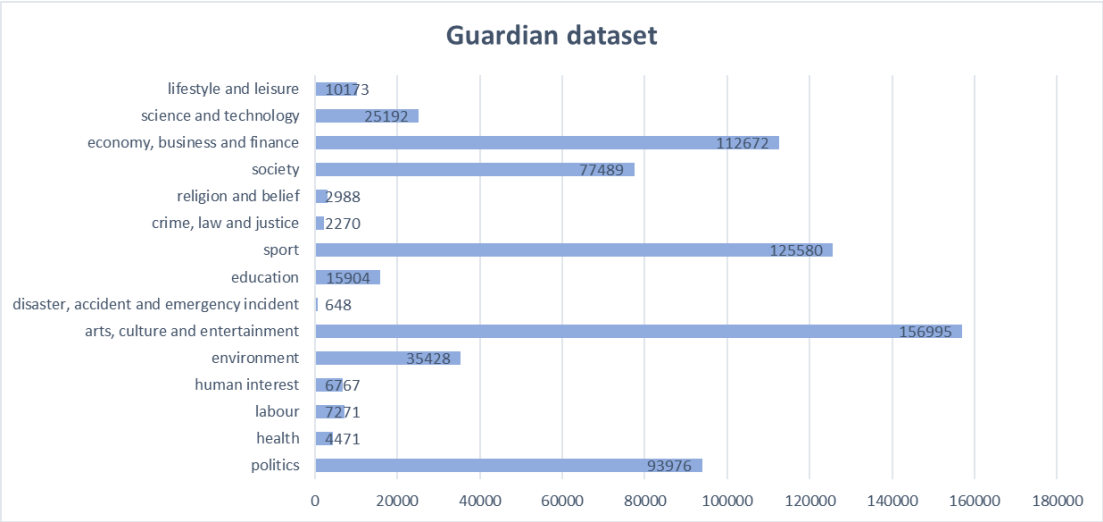

Figure 5.2: The distribution of news articles in the dataset gathered from The Guardian

Figure 5.2 shows how many times a category is present in a news article in the dataset. Both of the datasets were split into 70% training and 30% validation.

Classifiers

There are a lot of classifiers that can be used to solve text classification problems. For this experiment we used a SVM with a linear kernel and A multi-layer-perceptron (MLP) type neural network was used. The nerual network used a relu activation layer and an adam optimizer.

For the multi-label dataset, SVM was used from scikit-learn and a neural network was built using Keras. The neural network contained one hidden layer with a dropout function to reduce overfitting. It used relu activation for the input layer and softmax for the hidden layer with an adam optimizer.

Results

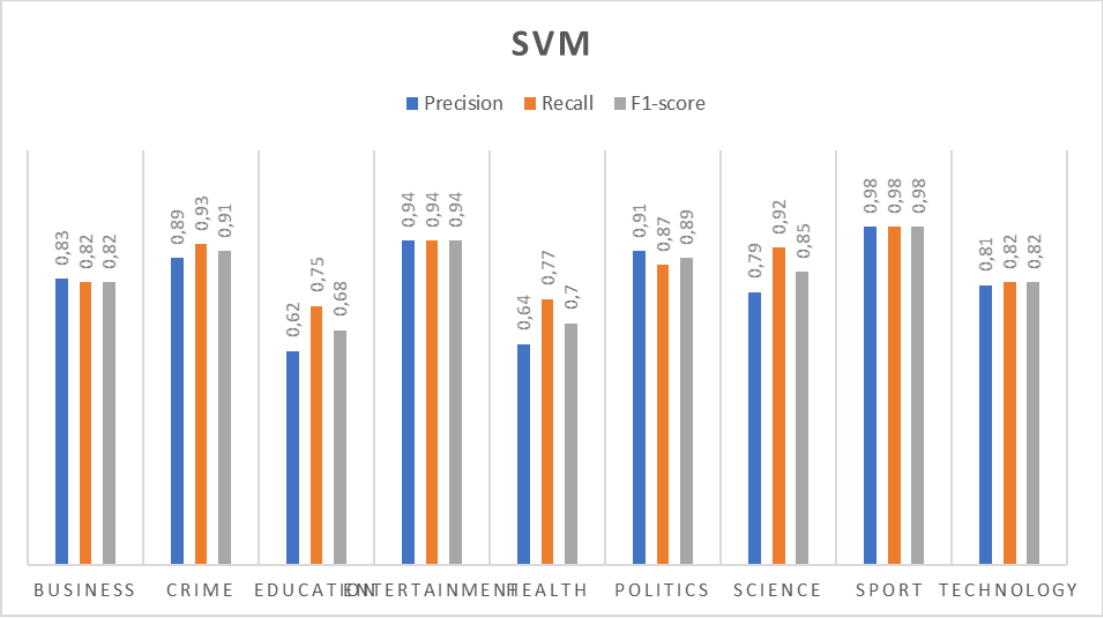

Figure 5.3: Precision, recall, and F1-score from the SVM

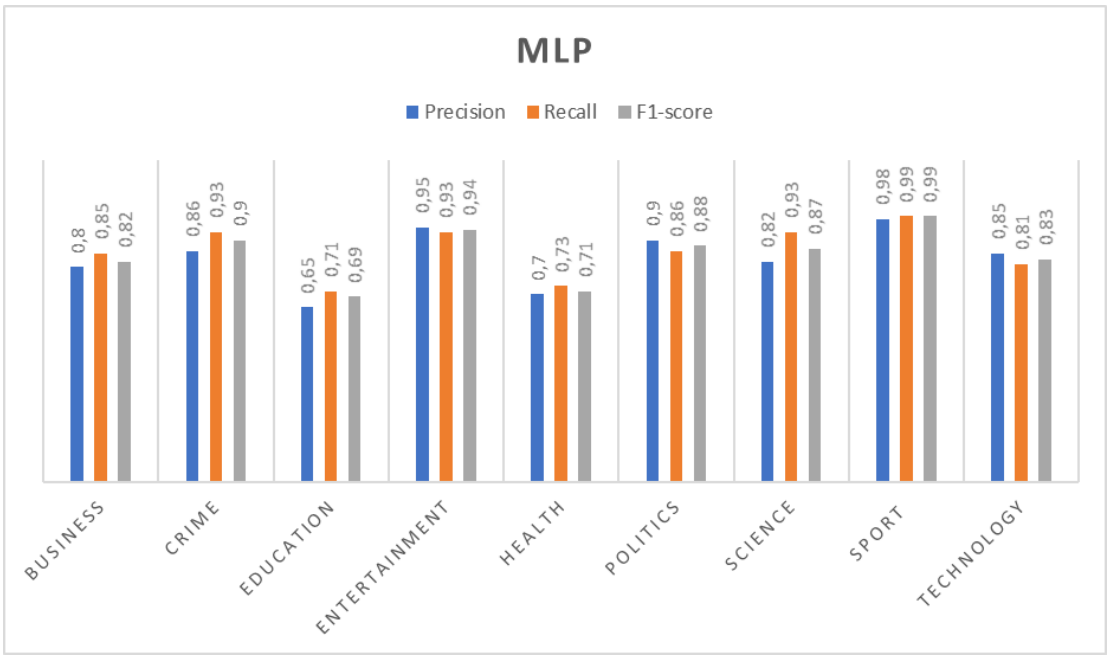

Figure 5.4: Precision, recall, and F1-score from the MLP

The results from the single-label classifiers were 0.89% accuracy. On figure 5.3 and 5.4 the precision, recall and F1-scores are listed for each classifier. The averages for precision, recall and F1-scores were all 0.89%.

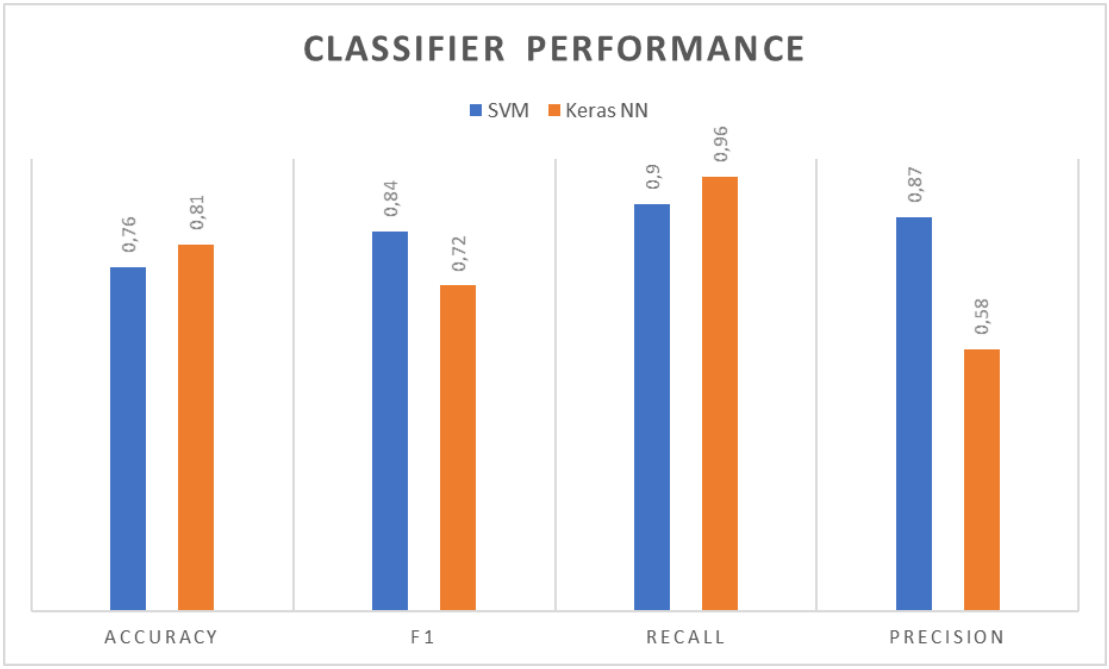

Figure 5.5: The results from the SVM multi-label classifier, precision, recall, and F1-score per category

The overall accuracy for the SVM was 0.76% while the Keras NN scored at 81%. The

F1-score, recall and precision can all be seen on figure 5.5

5.3.2 Clustering

Section 4.4.15 showed that some, but not all, of the clusters contained articles that were centered around specific events. As described in the introduction to the previously mentioned iteration, one of the goals with the clustering functionality was to pick up stories that potentially could be newsworthy.

Dataset

The articles used for testing the clustering process were gathered from a variety of diverse sources. For this test news from five distinct categories were chosen: Sport, politics, business, technology and entertainment. In the following tables the sources for each category is listed as well as how many articles that have been retrieved for each category. The number of articles is displayed in the parentheses next to the category name. Some of the sources are from the U.K and some of them are from the U.S.

Dataset	
Sport (172)	CNN, NY Times, Daily News, Channel News Asia, Independent, Denver Post, The Guardian
Politics (151)	Washington Post, BBC, CNN, Politico, CBS, NY Times, Denver Post, Washington Times, ABC, Reuters
Business (218)	Politico, Ny Times, CNN, The Guardian, Channel News Asia, Washington Post, Denver Post, BBC, ABC, Reuters
Technology (220)	Ny Times, CNN, The Guardian, CBS, BBC, ABC
Entertainment (531)	Washington Post, CNN, The Guardian, CBS, BBC, Denver Post, Independent, ABC

Evaluation

To evaluate how good this functionality was at detecting ”top stories” the results was compared to the top stories functionality from Google News. The Google News service did of course cover a wider variety of categories, and they had access to more sources within each

category. This evaluation did however give an indication of whether the functionality showed promise or not.

Results

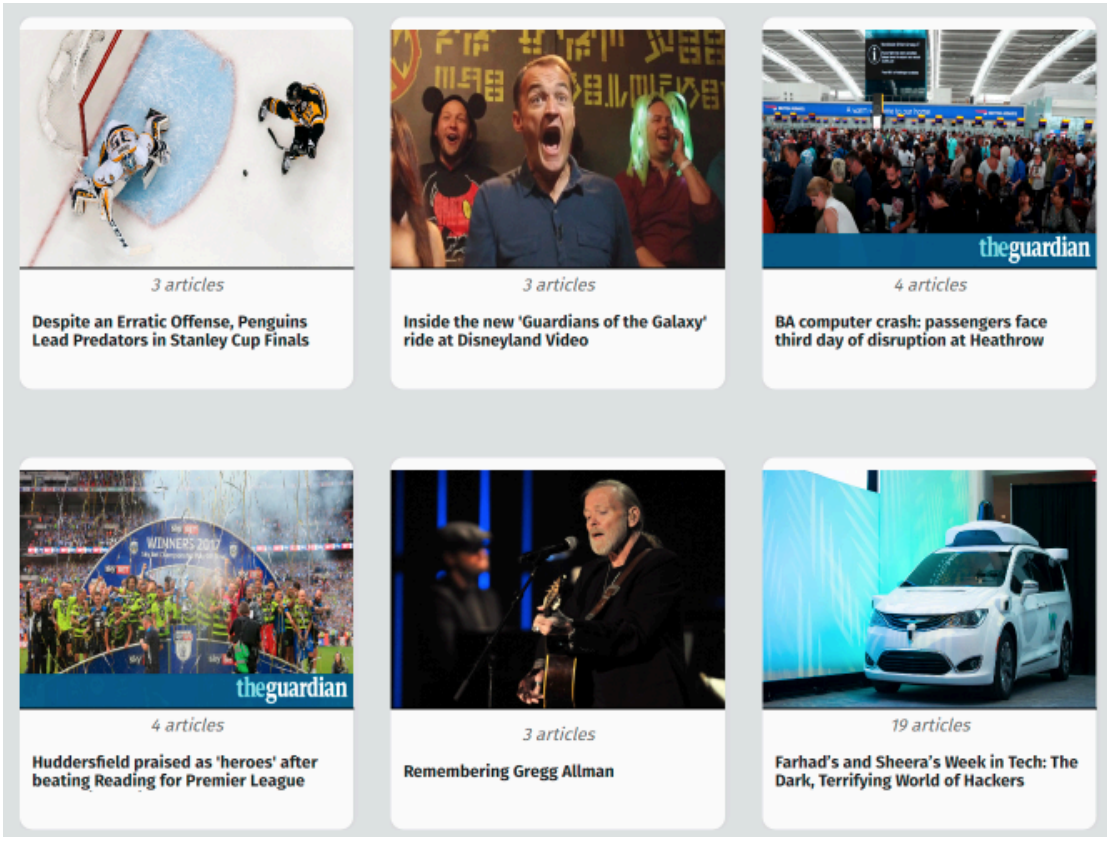

Figure 5.6: The six clusters returned from the clusters process

Figure 5.6 shows the retrieved clusters. Cluster number 2, and 6 are incorrectly clustered. The others contain articles about the same event. In the following figures screenshots from Google News will be presented. Since some of the articles are from the U.K and some of them are from the U.S, both nations versions of Google News have been looked at.

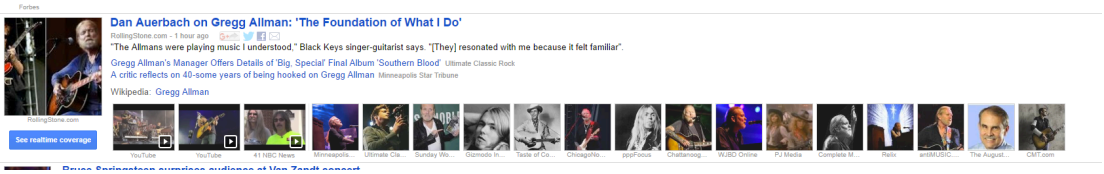

Figure 5.7: Screenshot from Google News: Story about Gregg Allman

Figure 5.8: Screenshot from Google News: British Airlines IT problems

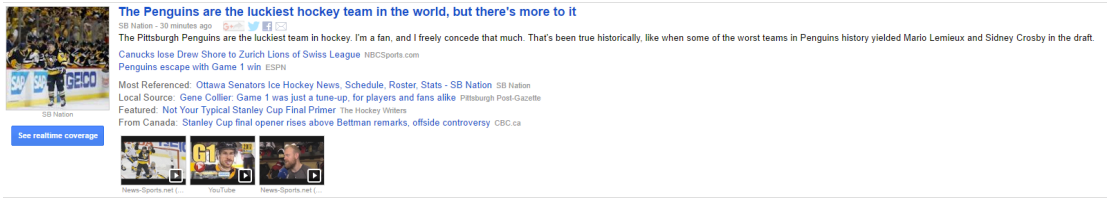

Figure 5.9: Screenshot from Google News: Hockey match with The Penguins

Figure 5.10: Screenshot from Google News: Story about Huddersfield Towns promotion to the Premier League

All of the correctly labeled clusters found in figure 5.6 was found in Google News. With the exception of the event shown in 5.10 all of them were found in the U.S version of the service, while the last one was found in the U.K version.

Chapter 6

Discussion

6.1 Research question 1

RQ1 - How can semantic web technologies be used to improve news aggregator systems?

Our development and evaluation suggests that semantic web technologies can indeed be used to create news aggregator systems with expanded functionality that journalists and domain experts consider an improvement over current solutions.

We found semantic web technologies useful for storing information extracted from text. The information was stored in a graph that expanded and connected to external sources every time a story was sent to the graph. Relationships between stories, named entities and keywords were created automatically when two articles had something in common. This provided interesting ways for searching through data. These features in addition to classification and clustering gave each story more ways to be inter-connected with both internal and external resources. Many of the included features would have been possible using other data-structures, but it would be harder to query. Expanding the graph gradually, and adding new connections to existing elements empowered the system. The semantic graph itself was therefore a necessity for News Hunter.

Named entities was one of the features that benefited most from the use of semantic technologies. DBpedia Spotlight allowed us to easily connect named entities in the graph with resources in DBpedia or Wikidata. This meant that we could populate the user interface with relevant text, images and access to open linked data 2.1.3. On one hand this proved very useful in News Hunter, but on the other it could be hard to trust these services in a full-scale

system due to their inconsistent updates of data. Investigating alternative sources for open linked data should be considered.

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.

Another feature that should be developed further is the implementation of an automatic translation service. The feature was developed at some stage as we described in section 4.4.8, but was not included in the final version of News Hunter. Being able to combine news articles from various sources and languages, and then make the information available to the user could prove to be a valuable tool. Having a multilingual system can potentially be used to gather information from open linked data in non-English languages, thus giving the journalists new inputs.

EventRegistry is one of the publicly available news aggregators that have made their system multilingual (Leban et al., 2016).

6.2 Research question 2

RQ2 - How can NLP and machine learning be used to connect stories in semantic news aggregator systems?

The evaluation suggests that NLP and machine learning can be used to connect information in semantic news aggregators.

In this section, we discuss the techniques used to create connections in the semantic graph, and how well they performed. The performance was measured using quantitative and qualitative research methods.

6.2.1 Keywords

The keyword evaluation was separated into two distinct parts. Part one was trying to find out which algorithm the users preferred. Part two dealt with measuring the performance of one of the algorithms using standardized testing methods for information retrieval.

Our user testing showed that TextRank was preferred to RAKE for keyword/key-phrase extraction. This was mainly due to the participants seeing multiple use cases such as general overview and searching. Some of the users did however prefer RAKE in a few cases due to it giving more context than TextRank. Having the option to choose between these two algorithms depending on usage could be beneficial. Some of the users were not satisfied with the output quality of the keyword algorithms. One solution that could be explored is to use more aggressive normalization for filtering out key-phrases or combinations that include the same word when they are presented to the user. Due to some of the keywords being named entities, removing these from the keyword should also be explored.

For the performance scoring the results were arguably not good. Some of the problems are explained in the first part i.e. normalization. Because of a small sample size of users, the keyword set provided by the participants was small. A larger sample size could have resulted in a wider variety of keywords for the user provided set. This could impact the F1-measure of the algorithm. In addition, having a larger sample size could better indicate which keywords were more important than others. Gathering this data and creating a dataset could be useful for further development as they can be used for automatic assessment of future improvements to existing or new algorithms. Given higher quality output from the keyword extraction the feature could be used for multiple purposes i.e. related stories.

6.2.2 Named entities

Our named entity evaluation was done by having participants look at a list of automatically extracted named entities from the articles they had previously read for the keyword evaluation. We found that all the participants were satisfied with most of the outputs, with only a few mistakes per article. We deployed a filtering method that removed named entities without a specific type. This was to present a best-case scenario to the users. These results implied that more aggressive filtering of the results from DBpedia Spotlight and Spacy can give reasonable output. Some feature, like ”related stories” uses named entities for matching articles, thus improving on the output quality would increase the usefulness in many areas of the system.

6.2.3 Classification

Classifying news articles into predefined news categories can used for grouping articles with the same category. This will enable easy retrieval of articles with the same category from the triplestore. High output quality is needed to make sure that relevant articles are not filtered out.

In this project, several classifiers were trained and tested on news data. The results from the classifiers are comparable to previous work done in the domain (Asy'arie and Pribadi, 2009; Romero and Koochak, Romero and Koochak). Asy'arie and Pribadi (2009) used a collection of 250 news articles with a Naive Bayes classifier. Their average recall score was 92.87% and their precision was at 91.16% Liparas et al. (2014) showed an approach that also used visual features and could achieve higher results than using just textual features alone. Their accuracy was 84.4% when using Random Forest machine learning on textual features split into 4 categories on 1043 documents. Their use of images in combination with textual features increased their accuracy to 86.2%. Romero and Koochak (Romero and Koochak) used a dataset with 32,602 articles over 7 categories and their softmax regression technique scored 75.36%.

The single-label classifiers implemented for News Hunter scored 89% accuracy on a dataset consisting of 78387 news articles over 9 categories. The use of SVM and MLP neural networks have shown to be effective. Finding the optimal settings for classifiers is a challenging task. During the implementations, we tested by tuning a few of the parameters available and so the work in this regard has not been exhausted. Future version should be tested with different settings to get the best accuracy. Different classifiers should be tested more in depth in addition to possible improvements in cleaning and pre-processing.

One difference between the News Hunter implementation and other work is the choice of using lemmatization instead of stemming on words. Whether this has an influence on the score could be tested further.

One issue with the current RSS implementation is that the parsing process as described in 4.4.9, is not perfect, which means that there may be some noise in the news articles. This could have a negative impact on the classification accuracy. The balance of the dataset was also not optimal, attempting to balance this out may provide better results in the future.

Categories such as sport, entertainment, and crime have such high accuracy that further work could comfortably be done in including more specific categories using a hierarchical ap-

proach.

Yang et al. (2009) used multi-label classification on a Reuters and Yahoo News dataset and could get between 60% - 81% on their F1 scores. Irsan and Khodra (2016) found using a Calibrated Label Ranking approach combined with Naïve Bayes to be effective when implementing a multi-label hierarchical classifier. They also found SVM to provide unsatisfying performance for their use. This kind of work could prove useful when developing a hierarchical approach.
The results from the multi-label classifier was an accuracy 76% for the SVM and 81& for the NN approach. While the NN scored higher on accuracy, the precision, recall and F1 score was less stable than the SVM. Considering the previous work done, trying to improve on the neural network implementation with deeper networks or using a recurrent (RNN) or convolutional (CNN) model may provide advances.

The multi-label dataset had some problems. The tagging process may be incomplete because the automatic process. This means that there could be missing tags and some articles may appear in more categories than they are currently tagged with. Training it on an expanded dataset that includes news articles from other news outlets should be done to create a more robust classifier. In addition to a bigger dataset, implementing a hierarchical classifier is needed. As with the single-label classifier, the categories: entertainment and sport scored consistently high, which means that hierarchical classifiers in these categories could produce satisfactory results.

Having more precise tags on the news articles for filtering and searching could help improve other features that we built including top stories, related news, and help filtering and searching through the feed. User evaluation of the automatic categorization of the news articles during the user evaluation showed that while the categories assigned were correct, they were too broad. Having more specific categories for news articles could be useful for users. If the rule-based classifier from the IPTC Extra project is of high quality, there is less need for further development of this feature.

6.2.4 Clustering

In the search for ways of connecting news stories, clustering of articles with content that describes the same event can potentially give many opportunities. In the clustering part 5.3.2 of the evaluation and results chapter the results indicated that the feature was able to detect articles that were written about the same event. Some issues with the approach was also visible in the results as one of the articles were clustered based on the source they came from and the only content of each article was a video. Having articles with limited textual data, and more focus on multimedia is quite common, therefore there should be done work on filtering out these articles, or find better ways for them to be included.
When a cluster was created, it was added to the semantic graph as a cluster. This enables some interesting use cases that should be further investigated. With clusters being added regularly it could be possible to look at clusters over time, to see which stories that were most written about throughout a year. Since articles that are part of a clusters are linked to the cluster, the articles are semantically linked to other articles that contains similar elements in the text.

6.2.5 Research critique

One of the problems with our algorithmic evaluation was that we did not have enough user generated keywords. Having a larger number of participants in the user evaluation would have given a larger keyword set. This would likely have an impact on the metrics. The named entity recognition evaluation could have been done differently. One improvement that could be done in future evaluations of name entity quality is to have journalists pick out and rank entities from shorter texts. This way we could measure both the accuracy and ranking.
The news classification task is a difficult problem to solve. While the implementations created in this project performed reasonably well, only a few classifiers were tested. There could have been more work put into the datasets to create more balance and find out why some categories scored higher than other, and potential ways to fix these problems.
Ideally the clustering process should have been tested on a larger dataset, and be evaluated regularly over time. The inclusion of additional sources should also be considered, especially if the main goal is to use the feature in the search for top stories. With a limited number of articles, the results are not as comparable with other services as we would have liked.

6.3 Research question 3

RQ3 - How can a semantic news aggregator aid journalists during research and writing?

The user evaluations suggest that some news aggregator tools that take advantage of semantic technologies can assist journalists during research and writing.
During the development of News Hunter, we explored a variety of potential features that could be incorporated into a newsroom setting. Some of the most promising features were included in the user evaluations. The features we chose to evaluate were selected based on research into related work and by feedback from the industry. Each of the features will be discussed separately, before a brief discussion of the artifact.

6.3.1 Feature: Summary

The user evaluation of this feature showed that the users were divided on the usefulness of the automatically extracted summaries. Some of the participants saw the benefits of a summary, but others would rather have a lead paragraph. The interviews showed that the output quality of the algorithm was problematic. In some cases, the summary algorithm delivered satisfying results, and according to some of the interviewees it could save time when reading up on a story. In other cases, the algorithms output generated incomplete and/or broken sentences. In its current state, the summary feature could be useful given that the output quality was satisfying. The tests showed that the quality was not as consistent as it should be in a complete system, therefore more optimization of the algorithm is needed, or the exploration of different approaches for summarization. Giving the user an option between lead paragraph and summary could potentially satisfy more users as it would increase ease of use.

6.3.2 Feature: Named entities

The named entity feature was one of the features that showcased the strength of the semantic web. All the users found the feature to be useful, but one of the domain experts did not like the idea of using open linked data due to data reliability. Even though the results show that it is useful in its current state, the results also show that there is room for improvement. Some of the suggestions retrieved from the user evaluation process was centered around ease of use. The term ”named entities” was confusing for one of the journalists, and perhaps

a renaming of the feature could make it more accessible. Sorting of the entities by type and giving more filtering options are other potential improvements. Additional sources and the option to choose between them is also something that could add to the user experience. Overall the feature shows promise, and can provide the users with valuable information even in its current state.

6.3.3 Feature: Keywords

During the user evaluations, it was clear that keywords could be used for multiple purposes. Some of the participants viewed it as a feature for meta-tagging and searching, while others saw it as a tool for getting a quick understanding of the content. Most of the users noted that the output quality should be more consistent. Because of the different views, half of the domain experts did not find the feature useful, the other half and the journalists did. The possibility to edit, remove or add keywords could potentially add to the user experience according to some of the participants. Further improvements of the keyword extraction process should be explored to get consistent high-quality output.

6.3.4 Feature: Related stories

The related stories feature was one of the most promising features according to the participants, as all of them saw the usefulness of this feature. Even though the feature proved to be useful, some improvements can be made. Showing additional information about each related story, and the ability to sort or filter the stories would add to its ease of use, this would again positively impact usefulness.

6.3.5 Feature: Top story

In its current implementation, the feature tries to group articles that are written about the same event. In the presentation of the feature all the grouped articles are listed. The participants in the user evaluation process found this to be useful for getting an overview of all the articles written about an event and for finding out what is currently happening and needs to be written about. All the users found the top story to be useful, but they all wanted additional information when looking at a top story. Having the option of searching and filtering top stories based on categories, sources, time and place are some of the suggestions from the users. Some of them

also wanted the functionality when looking at a single story to be included. Incorporating this additional functionality could further add to the usefulness.

6.3.6 Feature: Editor

The other key features of the system are centered around articles gathered from external sources. The editor feature is therefore unique in this system, due to it enabling the creation of content. In the implementation presented during the user evaluations the editor‘s only capability was automatic extraction of named entities and keywords. Some users found this to be useful for tagging the article automatically. All the participants saw the potential usefulness of the feature, but they wanted additional functionality if they were to use it. Adding previously written articles, external articles, external background information, and ways to edit the extracted entities and keywords were some of the functionalities the users wanted. In its current state the feature has arguably little usefulness, implementation of the proposed functionality could make the feature useful according to the users.

6.3.7 Research critique

The main problem with our user evaluation was the low number of participants. Ideally, we would have liked to include a larger number of people in our study. This could have given more variety in the answers we received and we could have included a TAM 2 questionnaire to properly score the entire system.

Chapter 7

Conclusion

In this thesis, we have presented a semantic news aggregator called News Hunter. The project was part of a collaboration between UiB and Wolftech. The goal was to provide journalists in newsrooms with enhanced investigation tools. News Hunter can deliver relevant background information such as named entities, keywords, related stories, categories and the grouping of similar news articles. An additional feature that came out of our research, was a text editor that could give journalists access to relevant background information.
The News Hunter system was built using C# and ASP.NET as back-end while the front-end of the application was built with AngularJS. Additional tools for extraction of textual data were implemented in a Python environment that used Flask to create endpoints to make the services accessible. Using agile iterations in our design research we could continuously add new features into the system. The continuous development cycle and feedback from Wolftech allowed us to move the project in new and interesting ways.

Users evaluated the distinctive features built into News Hunter. Different metrics for measuring the output quality of various components were also used in the system evaluation.
Most of the features were generally well received and users found the system to be useful. However, the results show that further improvements can be made. The mediocre output quality in some of the feature was the main drawback according to the users, this is also supported by the results from our information retrieval metrics. While higher output quality is needed, the results demonstrate the capabilities of News Hunter and encourages further development.

In the future, more thoroughly conducted evaluations should be considered a priority. Finding the features that can add the most value to semantic news aggregators should give an indication on what future development and research should focus on.
One of the features that we found most promising and innovative was the text editor. While the implementation we made was limited, it provided exciting opportunities. Getting related background information directly into the editor window, in addition to automatically creating connections between related content in real-time could be valuable extensions to the editor. Doing a more in-depth study on this would be an interesting contribution to research.
The related stories feature was another promising addition to semantic news aggregator systems. A well-developed version of this feature could be a time saver in a newsroom environment when searching for related news stories.
Multilingual capabilities are another addition that should be investigated further in future work. This would give the users more power to find relevant and useful information from sources in a variety of languages.

NLP and machine learning are two fields that are constantly evolving with new innovations that can provide opportunities for semantic news aggregators. Therefore its important to stay up-to-date on the latest research within the field.
Further development of the News Hunter ontology is needed to make it more adaptable to changes in the news industry. In addition, connecting it to other news and media ontologies should be investigated.

Bibliography

Ambler, S. W. (2013). User Stories: An Agile Introduction. *AgileModeling*, 1–5. 19

Asy'arie, A. D. and A. W. Pribadi (2009). Automatic news articles classification in Indonesian language by using Naive Bayes Classifier method. *Proceedings of the 11th International Conference on Information Integration and Web-based Applications & Services - iiWAS '09*, 658. 102

Beck, K. (1999). *Extreme Programming Explained: Embrace Change*. Number c. 18

Berners-Lee, T. (2006). Linked data-design issues. 6

Bizer, C., T. Heath, and T. Berners-Lee (2009). Linked data-the story so far. 6

Blei, D. M., B. B. Edu, A. Y. Ng, A. S. Edu, M. I. Jordan, and J. B. Edu (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research 3*, 993–1022. 17

Byun, H. and S. Lee (2002). Applications of support vector machines for pattern recognition: A survey. *Pattern recognition with support vector machines*. 13

Davis, A. L. (2016). Refactoring. In *Modern Programming Made Easy*, pp. 57–60. Berkeley, CA: Apress. 18

Davis, F. D. (1986). *A technology acceptance model for empirically testing new end-user information systems: Theory and results*. Ph. D. thesis. 19

Davis, F. D. (1989). Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. *MIS Quarterly 13*(3), 319. 19, 20

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. xi, 20

Ester, M., H. P. Kriegel, J. Sander, and X. Xu (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pp. 226–231. 12

Fern, N., M. Sintek, A. Bernardi, M. Fuentes, A. Marrara, and Z. Ben-asher (2006). NEWS: Bringing Semantic Web Technologies into News Agencies. pp. 778–791. 9

Foundation, O. Facebook SDK Package. 31

Gentleman, R. and V. J. Carey (2008). Unsupervised Machine Learning. In *Bioconductor Case Studies*, pp. 137–157. 12

Greene, D. and P. Cunningham (2006). Practical solutions to the problem of diagonal dominance in kernel document clustering. *Proceedings of the 23rd International Conference on Machine Learning (ICML'06)*, 377–384. 61

Gross, J. and K. McInnis (2003). *Kanban made simple: demystifying and applying Toyota's legendary manufacturing process*. 18

Group, O. W. OWL - Semantic Web standards. 5

Group, R. W. RDF - Semantic Web Standards. 5

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition 5*(2), 199–220. 5

Gupta, V. and G. Lehal (2010). A survey of text summarization extractive techniques. *Journal of emerging technologies in web intelligence*. 16

Hecht-Nielsen, R. (1989). Theory of the backpropagation neural network. In *International Joint Conference on Neural Networks*, pp. 593–605 vol.1. 14

Hevner, A. R., S. T. March, J. Park, and S. Ram (2004). Design Science in Information Systems Research. *MIS Quarterly 28*(1), 75–105. 23, 24, 25

IPTC. IPTC EXTRA. 71

IPTC. IPTC's EXTRA Project: Update - IPTC. 71

IPTC. Media Topics - IPTC. 71

IPTC. NewsCodes Concept. 72

Irsan, I. C. and M. L. Khodra (2016, aug). Hierarchical multilabel classification for Indonesian news articles. In *2016 International Conference On Advanced Informatics: Concepts, Theory And Application (ICAICTA)*, pp. 1–6. IEEE. 103

Jin, X. and J. Han (2016). K-Means Clustering. pp. 1–12. 12

Kaur, G. and K. Bajaj (2016). News Classification and Its Techniques: A Review. *18*(1), 22–26. 61

Kobilarov, G., T. Scott, Y. Raimond, S. Oliver, C. Sizemore, M. Smethurst, C. Bizer, and R. Lee (2009). The Semantic Web: Research and Applications. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 5554*, 723–737. 10

Kotsiantis, S. B. (2007). Supervised machine learning: A review of classification techniques. *Informatica 31*, 249–268. 12

Krstajić, M., F. Mansmann, A. Stoffel, M. Atkinson, and D. A. Keim (2010). Processing online news streams for large-scale semantic analysis. In *Proceedings - International Conference on Data Engineering*, pp. 215–220. 10, 29

Kruchten, P., R. L. Nord, and I. Ozkaya (2012). Technical debt: From metaphor to theory and practice. 18

Larman, C. and V. Basili (2003). Iterative and incremental developments. a brief history. *Computer 36*(6), 47–56. 19

Leban, G., B. Fortuna, and G. Marko (2016). Using news articles for real-time cross-lingual event detection and filtering. 11, 29, 100

Liparas, D., Y. HaCohen-Kerner, A. Moumtzidou, S. Vrochidis, and I. Kompatsiaris (2014). News articles classification using random forests and weighted multimodal features. In *In Multidisciplinary Information Retrieval, Springer International Publishing*, pp. 63–75. 102

Liu, B. (2010). Sentiment Analysis and Subjectivity. *Handbook of Natural Language Processing* (1), 1–38. 16

Lott, B. (2012). Survey of keyword extraction techniques. *UNM Education*. 16

Makhoul, J., F. Kubala, R. Schwartz, and R. Weischedel (1999). Performance Measures For Information Extraction. *Proceedings of DARPA Broadcast News Workshop*, 249–252. 15

Manning, C. D., P. Raghavan, and H. Schütze (2008). Introduction to Information Retrieval. *Journal of the American Society for Information Science and Technology 1*, 496. 17

Mendes, P. N., M. Jakob, A. García-silva, and C. Bizer (2011). DBpedia Spotlight : Shedding Light on the Web of Documents. *Proceedings of the 7th International Conference on Semantic Systems (I-Semantics). 95*, 1–8. 16

Nielsen, F. Å. (2011). A new ANEW: Evaluation of a word list for sentiment analysis in microblogs. In *CEUR Workshop Proceedings*, Volume 718, pp. 93–98. 16

Oh, K.-S. and K. Jung (2004). GPU implementation of neural networks. 14

Prud'hommeaux, E. and A. Seaborne (2008). SPARQL Query Language for RDF. *W3C Recommendation 2009*(January), 1–106. 6

Punch, K. F. (2005). *Introduction to Social Research: Quantitative and Qualitative Approaches*, Volume Second. 19

Romero, F. and Z. Koochak. Assessing and Implementing Automated News Classification. *pdfs.semanticscholar.org*. 102

Sammut, C. and Geoffrey I. Webb (2010). Accuracy. In *Encyclopedia of Machine Learning*, pp. 9–10. Springer US. 13

Sammut, C. G. I. W. (2010). POS Tagging. In *Encyclopedia of Machine Learning*, pp. 776–779. Springer US. 17

Shadbolt, N., W. Hall, and T. Berners-Lee (2006). The semantic web revisited. 5

Sommerville, I. (2010). *Software Engineering*. 17

Tom, E., A. Aurum, and R. Vidgen (2013). An exploration of technical debt. *Journal of Systems and Software 86*(6), 1498–1516. 18

Troncy, R. (2008). Bringing the IPTC News Architecture into the Semantic Web. pp. 483–498. Springer, Berlin, Heidelberg. 71

Vaishnavi, V. and B. Kuechler (2004). Design Science Research in Information Systems Overview of Design Science Research. *AIS*, 45. 23, 24

Veloso, A., W. M. Jr, M. Cristo, M. Gonc, and M. Zaki (2006). Document Classification. *Encyclopedia of Machine Learning and Data Mining*, 218–227. 13

Venkatesh, V. and Davis (2000a). A Theoretical Extension of the Technology Acceptance Model: Four Longitudinal Field Studies. *Management Science 46*(2), 186–204. xi, 21

Venkatesh, V. and F. Davis (2000b). Theoretical extension of the Technology Acceptance Model: Four longitudinal field studies. *Management Science 46*(2). 21, 22

Vicknair, C., M. Macias, Z. Zhao, X. Nan, Y. Chen, and D. Wilkins (2010). A comparison of a graph database and a relational database. In *Proceedings of the 48th Annual Southeast Regional Conference on - ACM SE '10*, pp. 1. 6

W3C (2008). SPARQL Query Language for RDF. 6

Wang, S.-C. (2003). Artificial Neural Network. In *Interdisciplinary Computing in Java Programming*, pp. 81–100. 13

Webb, G. I., J. Fürnkranz, J. Fürnkranz, J. Fürnkranz, G. Hinton, C. Sammut, J. Sander, M. Vlachos, Y. W. Teh, Y. Yang, D. Mladeni, J. Brank, M. Grobelnik, Y. Zhao, G. Karypis, S. Craw, M. L. Puterman, and J. Patrick (2011). Dimensionality Reduction. In *Encyclopedia of Machine Learning*, pp. 274–279. Boston, MA: Springer US. 15

Wolftech. Wolftech Homepage. 7

Yang, B., J.-T. Sun, T. Wang, and Z. Chen (2009). Effective multi-label active learning for text classification. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '09*, New York, New York, USA, pp. 917. ACM Press. 103

Zhou, S., Q. Chen, and X. Wang (2013). Active deep learning method for semi-supervised sentiment classification. *Neurocomputing 120*, 536–546. 16

Zhou, Z.-H. and M.-L. Zhang (2016). Multi-label Learning. In *Encyclopedia of Machine Learning and Data Mining*, pp. 1–8. Boston, MA: Springer US. 13

Appendix A

Interview questions

News Hunter:

Inspecting messages

1. What is useful about this feature?
2. What is missing from or problematic with this feature?
3. Is the feature useful?

Named entities

1. What is useful about this feature?
2. What is missing from or problematic with this feature?
3. Is the feature useful?

Keywords

1. What is useful about this feature?
2. What is missing from or problematic with this feature?
3. Is the feature useful?

Summaries

1. What is useful about this feature?
2. What is missing from or problematic with this feature?

3. Is the feature useful?

Related stories

1. What is useful about this feature?
2. What is missing from or problematic with this feature?
3. Is the feature useful?

Top stories

1. What is useful about this feature?
2. What is missing from or problematic with this feature?
3. Is the feature useful?

News editor

1. What is useful about this feature?
2. What is missing from or problematic with this feature?
3. Is the feature useful?

Article 1:

User keywords:

Which were better?

Were any of them useful?

Named entities:

Categories:

Article 2:

User keywords:

Which were better?

Were any of them useful?

Named entities:

Categories:

Article 3:

User keywords:

Which were better?

Were any of them useful?

Named entities:

Categories:

Appendix B

Articles

Trump gives nod to Republican tax-credit proposal on Obamacare

WASHINGTON U.S. President Donald Trump backed the use of tax credits to help people purchase health insurance in a speech to Congress on Tuesday, the first time he signaled support for a key component of House Republican proposals to replace Obamacare. Republicans, who control the White House and Congress, are united in their opposition to former Democratic President Barack Obama's signature 2010 healthcare law, but have so far failed to agree on the details of how to replace it.

"We should help Americans purchase their own coverage, through the use of tax credits and expanded Health Savings Accounts,T̈rump said. B̈ut it must be the plan they want, not the plan forced on them by our government.D̈emocrats are ardently opposed to tampering with Obamacare, which provided coverage to millions of previously uninsured people. A draft Republican replacement for Obamacare would include an age-based monthly tax credit that Americans who do not get health insurance through their employer could use to buy coverage and take from job to job. Some Republicans have voiced resistance to that plan. The president's comments were also a nod to health insurers - whom Trump met with on Monday - who say tax credits are necessary to keep people in the market.

"The fact that he used the word tax credits is a signal to congressional Republican ranksẗhat he supports their proposals, said Tom Miller, a resident fellow in health policy at the American Enterprise Institute think tank. Trump also said Americans should be able to buy insur-

ance across state lines, a proposal favored by health insurers because it would enable them to offer plans in states with fewer regulatory hurdles. Trump said state governors should be given resources and flexibility on Medicaid, the government health insurance program for the poor, and ensure that n̈o one is left out.T̈hat appeared to be an attempt to ease concerns from the more than 30 governors who expanded Medicaid coverage under Obamacare. But Trump offered few details on how he would reconcile House Republican plans to unwind the expansion of Medicaid with promises to maintain coverage for those who gained health insurance under Obamacare. He also reaffirmed that those with pre-existing conditions should have access to coverage but did not say how that would be accomplished.

”Categories”: ”economy, business and finance”
”Category”: ”Politics”

Keywords 1:

1. ”house republican plan”
2. ”house republican proposal”
3. ”health insurance”
4. ”medicaid coverage”
5. ”republican tax”
6. ”credit proposal”
7. ”tax credit”
8. ”trump”
9. ”health”
10. ”state governor”

Keywords 2:

1. ”democratic president barack obamas signature 2010 healthcare law”,

2. ”president donald trump backed”,
3. ”age-based monthly tax credit”,
4. ”expanded health savings accounts”,
5. ”reconcile house republican plans”,
6. ”government health insurance program”,
7. ”people purchase health insurance”,
8. ”draft republican replacement”,
9. ”congressional republican ranks”,
10. ”american enterprise institute”,

- Republican (Republican Party) -> Political Party
- Washington, D.C -> City
- Donald Trump -> Person
- Congress (United-States-Congress) -> Legislature
- health insurance (Health-insurance-in-the-United-States) -> Legislature
- White House (Presidency-of-Barack-Obama) -> Office Holder
- Democratic (Democratic Party) -> Political Party
- Barack Obama -> Person
- Tom Miller -> Heroes character
- American Enterprise Institute -> Organization
- Americans -> Nationality

UKIP burka ban policy ’misguided’ says party’s MEP

UKIP’s foreign affairs spokesman has resigned his post in protest at leader Paul Nuttall’s

plans to ban the burka. Mr Nuttall sparked controversy over his proposals to outlaw the full-face veil - as well as forcing girls at risk of female genital mutilation (FGM) to face regular medical checks. Announcing his decision to quit, West Midlands MEP James Carver described the policies as ẅisguided.̈ But UKIP deputy leader Peter Whittle defended the policies. T̈he burka is not something in the Koran - it's not specified by the Koran - it's a cultural practice. FGM is a cultural practice,ḧe said. Mr Nuttall's so-called integration strategy has already been panned by former UKIP donor Arron Banks, who accused him of going to ẅarẅith Muslim communities. Mr Carver said he s̈trongly disagreedẅith the policies. Ï would be the first to condemn a ban on wearing a crucifix as an infringement of liberty,ḧe said. N̈o-one has the right to dictate what people should wear.

"When facial identification is necessary, such as at passport controls, or in a bank, then it is perfectly reasonable to order the removal of veils - as is the practice - but in a free and liberal society, people have a right to their religious beliefs and to dress as they see fit.M̈edia playback is unsupported on your device Media caption UKIP leader Paul Nuttall says full-face veils hinder integration Mr Carver said he stepped down as UKIP's foreign and Commonwealth affairs spokesman ẅith deep regret,̈ but he added: Ï feel this policy undermines my desire to represent all communities within the West Midlands, including the many British Muslims, who I know from first hand experience, voted to leave the EU in last year's referendum.

"I have consistently spoken of my desire for a truly global perspective for the United Kingdom, outside of the European Union, and I see this policy as being incompatible with that aim.B̈ut Mr Whittle told BBC Radio 4's Today programme UKIP had discussed these issues for many years, including banning face coverings, which the party believe are a b̈arrier to integration in our society.̈ However the Muslim Council of Britain's secretary general Harun Khan condemned the move, and is due to say at a Muslim leadership dinner: Ït is deeply regrettable that the UK Independence Party is singling out Muslims in the most negative terms. T̈his is not leadership - and it is certainly not becoming of a party seeking to represent British people. T̈ogether with Britons, Muslims will be looking for true leadership that unites our country during this time of uncertain transition and seek to represent all British people, and certainly not scapegoat some. Bigots must not be allowed to set the terms of the debate.M̈eanwhile, Mr Whittle said Mr Nuttall p̈robably wills̈tand in the 8 June general

election, after the leader sidestepped the question at an event on Monday.

”Categories” ”politics”

”Category”: ”Politics”

Keywords 1:

1. ”ukip burka ban policy”
2. ”ukip leader”
3. ”mr nuttall”
4. ”muslim leadership”
5. ”mr whittle”
6. ”mr carver
7. ”ukip”
8. ”british people”
9. ”british muslims”
10. ”foreign affair spokesman”

Keywords 2:

1. ”mr whittle told bbc radio 4s today programme ukip”
2. ”device media caption ukip leader paul nuttall”
3. ”ukip deputy leader peter whittle defended”
4. ”britains secretary general harun khan condemned”
5. “full-face veils hinder integration mr carver”
6. ”mr nuttalls so-called integration strategy”
7. ”west midlands mep james carver”
8. ”leader paul nuttalls plans”,
9. ”ukip donor arron banks”,

10. ”mr nuttall sparked controversy”,

    Named entities:

    - UKIP (UK-Independence-Party) -> Political Party
    - MEP (European Parliament) -> Legislature
    - Paul Nuttal -> Person
    - West Midlands -> Region
    - Peter Whittle -> Person
    - Commonwealth (Commonwealth of Nations) -> Country
    - EU (European Union) -> Country
    - United Kingdom -> Country
    - BBC Radio 4 -> Radio Station
    - Today programme -> Radio Programme
    - Muslims -> Religious group
    - June General Election (Greek-legislative-election,-June-2012)-> Election
    - James Carver -> Person
    - Arron Banks -> Person
    - Harun Kahn -> Person

    Paul Pogba: Is he worth $120 million?

As history’s most expensive footballer, Manchester United midfielder Paul Pogba has been the subject of intense scrutiny since returning to English football. What is player ROI? How it’s calculated The English Premier League is awash with money, with its current TV deal reportedly worth nearly $11 billion. But which Premier League players are delivering the best bang for the buck for their clubs? Tracking a number of live on-field performance indicators – such as goals, key passes and defensive actions like blocks and interceptions –

Player ROI, or return on investment, compares players' game performance to their cost to find out. The 23-year-old, paid a reported $360,000 a week, has just two Premier League goals to his name, and took until November to register his first league assist. Some have claimed the Frenchman's Old Trafford career since his $120 million transfer from Juventus has been more notable for his haircuts and dance moves than his in-game contributions but, at least statistically, much of the criticism has been wide of the mark. Here are six reasons why Pogba is beginning to pay back the investment United made: Four years after he left for Juventus without a Premier League start to his name, Pogba's second Manchester United debut was promising, with the Frenchman registering more touches and passes in the opposition's half than any other player. Pogba has created 25 goalscoring opportunities in the Premier League this season – more than any other summer signing or Manchester United player. Pogba has performed 39 successful dribbles – more than any other central midfielder, and as many as Man United midfield partners Juan Mata (12), Ander Herrera (14), Henrikh Mkhitaryan (7) and Marouane Fellaini (6) put together. Pogba has completed an average of 57.6 passes per game in the Premier League this season. Only Man City's Fernandinho (60 per game), Arsenal's Santi Carzola (62), and Liverpool's Jordan Henderson (75) make more. The 23-year-old has an average pass length of 17 meters, and 76% of his 776 passes have gone forward. Not known for his defensive prowess, Pogba has made 24 Premier League interceptions – the fourth most at Manchester United and just one fewer than Chelsea defensive midfielder, Nemanja Matic. He has also been strong in the air, winning 41 aerial challenges – the same number as Manchester United defenders Chris Smalling (18), Eric Bailly (8), Daley Blind (15) combined. The Frenchman has the best overall Performance Score (258) of all the twenty summer signings included in CNN Money's 'Return on Investment' interactive. However, he may be the world's most expensive player, but so far he's only placed 17th in the ROI rankings with a 0.89 rating.

”Categories”: ”sport”
”Category”: ”Sport”
Keywords 1:

1. ”manchester united player”
2. ”premier league player”
3. ”manchester united midfielder”

4. ”premier league goal”
5. ”english premier league”
6. ”premier league interception”
7. ”player roi”
8. ”manchester united”
9. ”premier league”
10. ”investment united”

Keywords 2:

1. ”man united midfield partners juan mata”,
2. ”current tv deal reportedly worth”,
3. ”live on-field performance indicators –”,
4. ”manchester united defenders chris smalling”,
5. ”made 24 premier league interceptions –”,
6. ”manchester united midfielder paul pogba”,
7. ”winning 41 aerial challenges –”,
8. ”performed 39 successful dribbles –”,
9. ”twenty summer signings included”,
10. ”interceptions – player roi”,

Named entities:

- Paul Pogba -> Person
- Football -> Sport
- Manchester United -> Soccer Club
- England -> Country

- Premier Leauge -> Soccer league
- Awash River -> World Heritage Site
- Current TV -> Television Station
- Old Trafford -> Stadium
- Juventus FC -> Soccer Club
- Juan Mata (Soccer player) -> Person
- Ander Herrera (Soccer player) -> Person
- Henrikh Mkhitaryan (Soccer player) -> Person
- Marouane Fellaini (Soccer player) -> Person
- Fernando Luiz Roza (Soccer player) -> person
- Arsenal FC -> Soccer Club
- Santi Cazorla (Soccer player) -> Person
- Liverpool FC -> Soccer Club
- Jordan Henderson (Soccer player) -> Person
- Chelsea FC -> Soccer Club
- Chris Smalling (Soccer player) -> Person
- Daley Blind (Soccer player) -> Person
- CNN -> Television Station
- Player ROI -> Person
- Frenchman -> Nationality
- Nemanja Matic -> Person
- Eric Bailly -> Person