LONDON'S GLOBAL UNIVERSITY

UCL

Using natural language processing to develop a pipeline to analyse media representation of people with disabilities in Web-based news articles

Collection and filtering of Web-based news articles, comparison of open-source sentiment models, and applications of the technology

Bagus Maulana[1]

MEng Computer Science

Catherine Holloway, Nicholas Firth

Submission date: 30th April 2018

[1]**Disclaimer:** This report is submitted as part requirement for the MEng Degree in Computer Science at UCL. It is substantially the result of my own work except where explicitly indicated in the text. *The report may be freely copied and distributed provided the source is explicitly acknowledged*

Abstract

Report Title: Using natural language processing to develop a pipeline to analyse media representation of people with disabilities in Web-based news articles: Collection and filtering of Web-based news articles, comparison of open-source sentiment models, and applications of the technology

Authors Name: Bagus Maulana

Supervisors Name: Catherine Holloway, Nicholas Firth

Date and Year of Submission: 30th April 2018

Analyses of news media, and representation of groups such as people with disabilities, is a popular research theme in various fields. A common approach is to manually analyse a small sample of a few hundred news articles and generalise an overall conclusion from that sample. Computational natural language processing (NLP) can be used to process articles much faster and vastly increase the sample size, which can uncover further information from the corpus, such as trends (i.e. how the conclusion varies over independent variables).

This project explores the feasibility of developing a computational pipeline that performs data collection (from online news sources), filtering, parsing, sentiment scoring, and statistical analysis. This pipeline is then used in an experiment to collect articles from three major British online news publishers, extract information, and show trends regarding how the sentiment of news articles varies over time and between different publishers.

Results indicate that for some metrics (such as moving average of sentiment score) and on certain keyword categories, minor trends over time and between different publishers are apparent, although inconsistent. While the approach showed promise in performing quantitative analysis upon large bodies of literature (and specifically in the media-representation domain), it is highly recommended that future approaches to this media-analysis problem would improve upon this work by training a domain-specific filter and sentiment scorer with labelled data to improve the accuracy (and thus consistency) of sentiment scores.

Contents

Chapter 1

Introduction

Natural language processing (NLP) encompasses a wide range of computational techniques for machine understanding of human (natural) language that are often used alongside each other. The review article [1] defines natural language processing as 'a theory-motivated range of computational techniques for the automatic analysis and representation of human language.' The techniques that fall under the umbrella natural language processing include word tokenisation, probabilistic language modelling, translation, part-of-speech parsing, sentiment analysis (or opinion mining), text classification/categorisation, and topic modelling, among other things. The computational models used in natural language processing range from simple rule-based models (e.g. splitting a sentence on whitespace to tokenise words) to statistical machine learning and deep learning models. Natural language processing is now applied for various everyday technologies, for example, information retrieval for search engines such as Google and Bing, and categorisation and topic modelling for recommendation engines used to suggest 'similar' articles.

The obvious advantage of natural language processing is that machines can process vast bodies of human-created literature (books, articles, posts, e-mails, messages, etc.) much faster than humans can, processing thousands or millions of documents per second. This allows for high-level quantitative analyses of all documents in a vast corpora for a given domain to be feasible, which can uncover information previously inaccessible by only reading and generalising from a small sample of documents. For example, this level of quantitative analysis can uncover trends and patterns within a given domain (e.g. how does the popularity of the term 'mentally ill' increase or decrease year-on-year in English news media?).

Various other studies have attempted to utilise natural language processing to perform high-level analyses in the domain of news media. The research done in [2] assembled a vast corpus of regional newspapers in the United Kingdom spanning 150 years to detect long-term patterns of cultural change (e.g. increase of female representation in the news, or when trains overtook horses for transportation) by analysing n-gram trends and named entities. More specifically, in the domain of media representation of specified groups of people, studies such as [3] has attempted to use features based on natural language processing (such as n-grams and part-of-speech tags) to classify racist and sexist posts in social media, although there is still a research gap in this area (especially for news articles, and/or relating to specific groups, such as people with (specific) disabilities or mental illnesses).

Applying natural language processing to perform meta-analyses over large text corpora has various interesting potential applications in improving our understanding of the human world - for example, to detect macroscopic cultural shifts as in [2]. In particular, the representation of specific groups, such as people with disabilities, has been a popular research theme for psychologists, sociologists, and others. For example, the paper [4] analysed a sample of 600 print articles relating to mental illnesses in New Zealand and categorised them to positive and negative depictions, and the predominant themes thereof (e.g. criminality (negative), educational accomplishments (positive)). Applying natural language processing to this area of research would allow the possibility of discovering higher-level trends, by computationally analysing a much larger sample of articles and identify trends by varying independent variables such as year of publication and publisher. In this research, a sample of 305,185 news articles (48,967 after filtering off-topic articles) from British online news sources are used. However, challenges remain as syntax-based statistical natural language processing approaches tend to be more limited in scope and is prone to false positives and negatives, and mitigating these factors is currently an open area of research.

The aim of this project is to utilise these computational natural language processing techniques in order to perform a high-level meta-analysis of literature available in the public news media found online. More specifically, to gather online news articles relating to people with disabilities in British media, and perform natural language processing analyses at scale to identify trends such as term popularity (e.g. 'suffer from ...' vs 'with ...') and variation in positive/negative sentiment.

The goal of this project is to develop a computational pipeline capable of performing this analysis of online media end-to-end. Given a list of topics consisting of keywords (or key phrases) and query terms, this pipeline covers the task of web crawling and scraping to collect a dataset of news articles; filtering relevant articles for the given keywords; matching relevant sentences referring to a keyword; performing sentiment analysis (using open-source libraries) on these sentences; and producing relevant visualisations to show trends as evidence of applications of the technology. This pipeline will be available open source on GitHub (`https://github.com/bmaulana/nlp-media`)

This project was carried out in a step-by-step approach. The pipeline was developed as individual components: a web scraper and crawler for data collection given a list of queries; a filter to remove irrelevant articles given a list of key terms; a parser to pattern-match relevant sentences given key terms; a sentiment scorer (and results analysis/comparison of different open-source scorers on this domain); and a script to perform statistical analysis on the results and produce relevant plots. A main pipeline script connects these components together by calling them in order for each topic and Web news source (Daily Mail, Daily Express, Guardian). Each component's output is piped to the next component's input by saving its output to a JSON file and having the next component read the previous component's output file, which ensures computation can be 'resumed' without re-running the previous component.

The body of this report is subdivided into four chapters:

- Chapter 2 covers related work on the domain of news media analysis (especially in the context of people with disabilities) and natural language processing, and background information regarding the relevant tools and techniques researched for this project.
- Chapter 3 defines a structured list of requirements, goals, and expectations for this project.
- Chapter 4 documents the design and implementation of the computational pipeline and its components that were used to carry out the data analytics experiment and achieve the stated

goals.

- Chapter 5 discusses the results of the experiment in diagrams, graphs, and tables; and their implications.

Finally, the conclusion (Chapter 6) evaluates the project, summarises key achievements and take-aways, provides recommendation on how this work could be expanded upon, and sets guidelines for further work in this field. Furthermore, the bibliography lists sources that were used as references in this project, and the Appendix section contains the raw results of the experiment (i.e graphs and other data not in Chapter 5).

Chapter 2

Context

2.1 Background

Analyses of news media in its various forms (print, online, etc.) has been a consistent research theme. The news media provides a quantifiable depiction of the prevailing society's popular conceptions or views regarding a theme or topic. This is especially applicable with regards to conceptions on particular groups of people, in which the language used in the media reflect on popular views, and has been shown to differ (with statistical significance) in different societies. For example, it was shown that the Canadian press was more likely to name individuals with disability and use appropriate labelling than the Israeli press in 1998 [5]. Furthermore, there is evidence to suggest that news media sources contribute to shape and reinforce beliefs among the society, such as misconceptions and stigma [6].

Public awareness of disability is also a popular research theme. While not related to news media, a review in 2011 [7] found 75 articles and 68 studies that passed a selective inclusion criteria with regards to intellectual disabilities, published in English between 1990 and mid-2011. The topics brought up include the public's knowledge, attitudes, and beliefs about intellectual disability; and varying for socio-demographic characteristics, cross-cultural comparisons, and the effects of interventions.

Analyses of news or other media are primarily carried out by taking a small, statistically representative sample of documents (news articles) from a text corpora (for example, all news articles published in England for a certain period) and analysing them manually. There has also been various such studies within the domain of disability awareness in the media:

- In a 2002 study [4], researchers analysed a sample of 600 print articles relating to mental health or mental illness that was collected by a commercial clipping bureau. The articles were then categorised into positive and negative depictions, then further into sub-samples such as danger to others, criminality, vulnerability, etc. The study found that at the time, in New Zealand, negative themes predominate about 3 to 1 (with 27% being positive). However, given the paper's scope, this conclusion cannot be generalised to learn trends, or how the conclusion varies given certain variables (e.g. time, location).
- A study conducted in 1998 [5] were replicated in 2008 [8] to assess change in representations of disability and persons with disability in the Canadian news media. This study sampled

196 news articles in 1998 and 166 news articles in 2008. It found an increase in the usage of 'person-first' terminology (e.g. person with disabilities) and a decrease in 'disabling language' (e.g. disabled person). This is an attempt to identify trends with regards to media representation of disability, however only provides two data points (1998 and 2008) with relatively small sample size.

- A study in 2005 [9] analysed 1,515 articles relating to autism in Australian news media. All articles were read by two research assistants to ensure they are on-topic and then coded as either 'negative' or 'positive' in overall focus, and then coded into themes (e.g. funding, education, etc.).

By applying natural language processing and computational techniques in analysing text articles, it is possible to develop a computational pipeline that could analyse and extract quantitative information from these articles at a much faster rate, enabling the analyses of a much larger scale of documents within a realistic time frame. While a sample of few hundred or thousand documents is usually enough to provide statistically significant conclusions, by providing an analysis of the full corpora (or a much larger sample), it is possible to uncover additional information from the data set. For example, higher-level trends (such as how the conclusion varies by year, location, publisher, etc.) can be discovered from a quantitative analysis of the larger dataset, by 'splitting' the result set into smaller subsets based on independent variables (e.g. year, location, publisher, etc.) and performing statistical comparisons of dependent variables (e.g. term frequency) between each subset.

Additionally, data collection is much more feasible in scale, cost and time using computational techniques. An automated script can be used to collect news articles published on the Internet at a rate of roughly one article per second, or thousands of articles per hour (varies on Web source, hardware, Internet connection, etc.), a vast improvement over contracting a commercial clipping bureau to provide 600 articles as in [4].

Several studies has taken advantage of this approach to carry a more complete analysis of textual corpora. For example, [2] assembled a corpus of 35.9 million news articles from 120 publishers in the United Kingdom between 1800 and 1950, representing 14% of all news articles published in the United Kingdom over that period. With this approach, the researchers were able to extract quantitative time-series information with regards to cultural trends as present in 35.9 million British news articles over the 150-year period. This amount of large time-series information (represented as n-grams and named entities) allowed them to discover macroscopic cultural trends. By analysing and comparing word (n-gram) trends across various topics, the researchers were able to identify trends that reflect cultural shifts e.g. 'train' increasing in popularity and overtaking 'horse' around 1900, or 'labour party' overtaking 'conservative party' and 'liberal party' in news coverage from the 1920s onward. Additionally, they also used entity recognition to extract named entities from articles and considered trends based on known information about these named entities, such as the proportion of female vs male entities, categories of entities (e.g. politicians, writers, etc.), and age of these entities. They also considered the geographical location of the publication to see how word usage trends (e.g. 'british' vs 'english') differ based on location.

[2]'s study was based on prior discussions and studies on the potential of exploiting large text corpora to detect macroscopic, long-term cultural changes. [10] was one of the first studies to suggest this approach. In this seminal study, a corpus of 5 million digitised English-language books

published over 200 years (or about 4% of all books ever published), provided by Google's effort to digitise books, were analysed to extract how often a given n-gram was used over time. (This data is available on `http://www.culturomics.org/`) This information is then used to analyse trends in language: the size of the English lexicon, regularisation of English verbs (from 'irregular' suffixes to '-ed'), or how quickly years (e.g. '1950') decline in use. Influenced by [10], several other studies has been published adopting a similar approach:

- an analysis of 1.7 million Victorian-era books [11]
- an analysis of 17,094 US Billboard Hot 100 songs between 1960 and 2010 [12]
- an analysis of a 3.9 million news article sample from the Summary of World Broadcasts (SWB) collection [13]
- an analysis of 2.5 million English-language million news articles from 498 online news outlets from 99 countries [14]

However, there were also criticism of this approach, such that it ignored semantics and context, thus for example 'thirteen hundred words of gibberish and the Declaration of Independence are digitally equivalent' [15], issues with OCR quality and duplicate editions [15], or that the selection of digitised books are biased [16].

There has been some progress in applying natural language processing techniques more specifically in the domain of how (specific) groups of people are represented in the media:

- A study has attempted to extract features (such as n-grams and part-of-speech tags) using natural language processing to classify racist and sexist posts in social media, providing annotations to 6,909 tweets [3].
- Another study explored potential linguistic markers of schizophrenia in social media from a dataset of 174 users with self-reported schizophrenia and up to 3,200 tweets per user, and control users in a 50:50 split. They used a support vector machine (SVM) approach over features based on lexicon-based approaches (i.e. a list of mental health related keywords), latent dirichlet allocation (LDA), Brown clustering, character n-grams, and perplexity [17].

However, there is still a visible research gap in this area, for using computational natural language processing specifically for news articles and/or with regards to specific groups, such as people with disabilities (or a specific disability).

To date, advances in natural language processing has made this approach much more feasible, efficient, and effective, even given limited time and hardware constraints. Various free and open-source tools for computational natural language processing and statistical analysis has been developed by the research community, such as nltk [18] and StanfordNLP [19] for natural language processing, scikit-learn [20] for statistical analysis, and matplotlib [21] for plotting. Depending on the type of data analytics performed and the hardware used, these computational tools are able to analyse news articles at a rate of multiple documents per second. Section 2.3 contains a further listing and discussion of these tools, alongside the specific libraries and natural language processing techniques used for this research project.

2.2 Research Methodology and Sources

Background research were carried out by investigating papers from public sources (such as Google Scholar). Research articles were gathered from a list of important topics and query terms related

to the technique (natural language processing) and/or the domain (disability and news media): for example, 'natural language processing review', 'news media' AND 'disability', 'natural language processing' AND ('news media' OR 'cultural trends'), and 'natural language processing' AND 'disability'. Highly-cited research articles are prioritised as examples, as they are deemed to be more 'important' papers or studies within its topic. Additionally, the author looked for highly-cited 'key' papers and review articles within each specific topic, and then looked at its list of citations (older papers), and research articles that cites the review article (newer papers), to expand the list of relevant research papers and examples.

Technical research, on the other hand, were carried out as necessary. After the requirements and components for the pipeline has been decided, research were carried out to find relevant techniques, formulae, algorithms, tools, libraries/packages, and existing implementations that would be useful to implement each component (or sub-tasks within a component). The research were carried out in an iterative approach alongside software development, where initial planning and research would show an initial implementation plan, then implementation would uncover feasibility of these approaches and possible alternatives/refinements to be researched, then further research may reveal new options/refinements to be implemented, and so on. Research or work done on other components/tasks may also reveal possible improvements or alternatives for another task, which may require further research and implementation. Again, more popular tools and libraries are prioritised, although several approaches and implementations were considered for most components and sub-tasks, to be compared for suitability, runtime, results, etc.

The sources used for this literature review are:

- Google Scholar, often used to find research articles to act as 'entry points' towards a research topic, to find other studies similar to another research article within a specified topic, or to retrieve citation information of a given research paper, book, or popular Python package.
- GitHub topics, used to find repositories that are relevant to a specific task or component, find similar GitHub repositories, gather information about a given repository, and also as a benchmark for topic popularity and range of solutions in a given programming language. Several GitHub pages also curate a list of repositories within a specific topic [22]–[24].
- Python package repositories such as PyPi [25] and Anaconda Cloud [26], which list all available Python packages and shows general information regarding them, and also provides a search function; useful to find relevant packages for a component/task and gather information about a given Python package.
- Official web sites and documentation of Python packages, which list and define capabilities (functions and parameters) of the package, useful to explore the functionality of a given package and its capacity to solve a specific task, and to understand the technologies/approaches used by the package's implementation (e.g. how is a sentiment model implemented?). Often (especially with more popular packages), citation information regarding the package would also be available in its web site.

2.3 Technical Context

As mentioned earlier, computational natural language processing techniques are utilised to extract features from collected news articles (text documents) at scale. For this project, the requirements

for the computational pipeline can be subdivided to five main components: data collection, filtering, sentence matching, sentiment scoring, and statistical analysis and data visualisation. Among these components, natural language processing techniques are necessary for filtering, sentence matching, and sentiment scoring. On the other hand, collection is performed using established, general-purpose web-scraping tools. Similarly, statistical analysis and data visualisation is performed using general-purpose statistical tools and metrics. This section will cover the technical tools researched and used for all listed components regardless.

Natural language processing covers three main 'curves' or areas: syntax, semantics, and pragmatics (narratives, understanding). Syntax specifies the way symbols (words, terms, tokens, or n-grams) and groups of symbols are arranged and whether they are well-formed in an expression, whereas semantics specifies what these expressions mean, and pragmatics specifies contextual information [1]. Contemporary (or 'traditional') approaches to natural language processing mainly focus on syntactic analysis, due to the relative ease of extracting syntactic features of text such as term frequency, word co-occurrence, and part-of-speech tags, compared to extracting logical expressions and networks necessary for semantic analysis. However, syntactic analysis is much more limited as it often misses information such as the (semantic) context of a word (e.g. "one" in "there's no one there" (referring to a person) vs "we have only one car" (referring to a quantity)). This paper will focus on mainly syntactic techniques and features, as these are more relevant to this domain of high-level topic matching and sentiment analysis that is feasible with current technology at this scale.

Python was chosen as the main programming language used for this project. The primary reason for this choice is the wide availability and range of existing tools for natural language processing, sentiment analysis, statistical analysis, data visualisation, web scraping and parsing, etc. in Python. A study in 2016 showed that Python is the most popular language for machine learning and data science [27], which correlates to the amount of available tools developers have created for the language. A GitHub search for the topic 'nlp' as of 18 April 2018 reported 1,397 Python and 470 Jupyter (Python interactive 'notebook') repositories with the tag 'natural language processing', compared to the second most popular programming language being Java with only 251 repositories tagged 'nlp' [28]. Furthermore, Python is also an ideal language for quick experimentation due to relatively high-level and low verbosity of the code, such that it is relatively easier to make small changes on the fly. Additionally, the Anaconda distribution of Python [29] is used for its suitability to set up and manage Python environments and packages for data science projects.

2.3.1 Data Collection

For data collection, general-purpose tools for sending HTTP requests (to 'open' web addresses and store HTML web pages programmatically) and parsing HTML code (to parse article text and metadata from 'raw' HTML code) are sufficient. The Requests library [30] is a popular Python tool (with 400,000+ daily downloads) for sending HTTP/1.1 requests simply. A HTTP GET request will retrieve the HTML code (and other information) associated with a given URL from a web server, similar to opening the page on a web browser, stored as a Python object by Requests. Once the HTML code of a web page (given an article's URL) has been stored, the BeautifulSoup library

[31] provides simple methods to navigate and search a parse tree (such as HTML code). Given that web pages from the same source/publisher tend to follow a similar structure, BeautifulSoup can be used to parse article text and relevant metadata (e.g. headline, date of publication, outgoing links in a search page) by searching for specific tags and attributes within the HTML code.

2.3.2 Filtering

In this project, filtering off-topic articles is done somewhat crudely via ranking term frequency independently for each document. Term frequency is a simple and commonly-used metric for natural language processing, with various existing tools that can compute this metric for thousands of text documents within seconds. To put simply, the term frequency of a term (i.e. a word or token) in a document is the number of times that the term occurs in the document. The popular scikit-learn library [20] provides a tool to measure term frequency of text documents, handling both tokenisation (converting a text document into a list of tokens (terms/words)) and counting word occurrence. It also provides the option to ignore stop words (i.e. common, meaningless words in English, such as 'the' or 'a'), which can often skew the results due to its relative high frequency. Additionally, the nltk library [18] is used for stemming – to reduce all words in the document to its word stem (such that e.g. 'walk', 'walks', 'walked', and 'walking' are equivalent).

In literature related to natural language processing, several more complex approaches has been proposed and used for the task of text classification and filtering off-topic articles. A conventional approach is by calculating term frequency – inverse document frequency (tf-idf) [32], [33], a metric that builds on term frequency by taking into account the relative importance of each word. Inverse document frequency (idf) is calculated by counting the number of documents in a corpus where a term appears – if a term appears more frequently (e.g. common words such as 'the'), it is deemed to be less important and assigned a lower score. However, this approach is unsuitable for this project, given the selective nature of the dataset (where only articles containing certain query terms are collected), thus the idf values of these query terms would be flawed (as they would appear in every document).

Another proposed approach is by using a supervised machine learning model to classify documents into pre-defined categories (the text classification problem). Various approaches were proposed to solve text classification, including Support Vector Matrices (SVM), Naïve Bayes (NB), and k-nearest neighbour (kNN) models [34]. However, this approach is infeasible for this project due to a lack of labelled data (i.e. a dataset of articles and category labels, or in this case 'is this article relevant?' boolean labels).

Topic models, which compute the proportion of abstract 'topics' in a document, has also been proposed. Latent dirichlet allocation (LDA) [35] represents documents as random mixtures over latent topics, where each topic is characterized by a distribution over words. However, as these topics are abstract and characterized generatively (i.e. each topic's distribution over words are generated by the model, rather than pre-defined), it is not very useful for the task of classifying whether a document matches pre-defined topics/keywords. Additionally, LDA is significantly more computationally expensive than tf or tf-idf.

2.3.3 Text Parsing and Sentiment Analysis

Sentiment scoring of articles is sub-divided into two components: a component to find sentences relevant to a topic (given a list of key terms) in a text document (sentence matching), and another component that performs sentiment analysis on these sentences and transforms it into a real-valued score of each sentence. SpaCy [36] is a popular tool for general natural-language processing tasks, using pre-trained convolutional neural network models for tasks such as tagging, parsing, and entity recognition, and is benchmarked to be the fastest and among the most accurate syntactic parser, able to parse 13,965 words per second in 2015 [37]. Among the information SpaCy extracts from text are lemmas (root words) of terms (e.g. 'mentally' $\rightarrow$ 'mental') and features a rule-based matching engine (retrieve a list of sequences of tokens within a document that matches a given pattern, e.g. tokens with a specified lemma), both which are useful for the task of finding sentences relevant to a topic in a document.

For sentiment scoring of sentences, a variety of open-source tools and pre-trained models dedicated to sentiment analysis were researched for the purpose of comparison. Sentiment analysis, or opinion mining, is defined as "the field of study that analyzes people's opinions, sentiments, evaluations, attitudes, and emotions" from natural (human) language [38]. A GitHub search for the topic 'sentiment-analysis' as of 18 April 2018 reported 450 Python and 222 Jupyter repositories with the tag 'sentiment-analysis' [39]. Furthermore, a community-curated list of sentiment analysis methods and implementations exist [22], which served as a useful starting point to explore open-source sentiment analysis implementations.

The particular sentiment analysis implementations that have been explored in this paper are:

- VADER is a parsimonious rule-based model that scores the sentiment of a given sentence, based on the presence of 'sentiment lexicons' (a list of lexical features common to sentiment expression, such as 'good' and 'bad', including slang words, emoticons, and acronyms), and rules such as negations (e.g. 'not good') and increased sentiment intensity given emphasis such as punctuation, capitalisation, and degree modifiers (e.g. 'very') [40].
- vivekn's 'sentiment' repository implements a supervised machine learning approach to sentiment classification trained on an enhanced Naïve Bayes classifier, trained on a publicly available dataset of 25,000 highly-polar movie reviews from the Internet Movie Database (IMDb) using bigrams and trigrams as features [41].
- xiaohan2012's 'twitter-sent-dnn' repository trains a convolutional neural network model with dynamic k-max pooling (DCNN) for modelling (real-valued sentiment scores of) sentences. The sentence model properties considered are the word and n-gram order, and induced feature graph (generated by the DCNN). It was trained on a dataset of 1.6 million tweets with emoticon-based labels. [42]
- kevincobain2000's 'sentiment_classifier' repository trains a supervised machine learning model based on a Naïve Bayes and Maximum Entropy Classifier to transform a sentence to positive and negative (real-valued) sentiment scores. It uses bigrams as features, implements Word Sense Disambiguation using wordnet [43] to tranform bigrams to 'senses', and considers word occurrence statistics from nltk's movie review corpus. The training data is a mixture of nltk's movie review corpus, Twitter posts, and Amazon customer reviews data. [44]
- OpenAI's 'generating-reviews-discovering-sentiment' repository provides a pre-trained single-

layer multiplicative LSTM recurrent neural network model with 4096 units (a relatively simple model optimised for training/convergence time) to generate (real-valued) sentiment scores of input sentences. It was trained on a dataset of over 82 million Amazon product reviews from May 1996 to July 2014, substantially larger than previous work (and taking one month across four NVIDIA Pascal GPUs to train), and outperforms state-of-the-art models when tested on similar-domain corpora such as Rotten Tomatoes and IMDb reviews. Sentences are represented as a sequence of UTF-8 encoded bytes where for each byte, the model updates its hidden state and predicts a probability distribution over the next possible byte. [45]

- Stanford CoreNLP [19] provides a set of linguistic analysis tools, including sentiment analysis, given input text, while running in a local web server. Its sentiment analysis tool uses a recursive neural network model, representing text as parse trees, and trained on a Sentiment Treebank of fully-labelled parse trees for 215,154 unique phrases and 11,855 sentences from the Rotten Tomatoes movie review corpus; and classifies sentences into five sentiment classes, from 'very negative' to 'very positive' [46]. Although Stanford CoreNLP was written in Java, several packages exist that allow a Stanford CoreNLP local server to be started and queried programmatically in Python [47].
- TextBlob is a general-purpose natural language processing library similar to nltk, SpaCy, or CoreNLP. It provides two sentiment analysis models: PatternAnalyzer, a rule-based classifier based on part-of-speech pattern matching, and NaiveBayesAnalyzer, a Naïve Bayes classifier trained on a dataset of movie reviews. [48]

Several other repositories has also been explored, however deemed unsuitable for this project either due to requiring to be re-trained using labelled training data (which was unavailable for the domain of news articles), or the implementations are broken or infeasible.

2.3.4 Statistical Analysis and Data Visualisation

For statistical analysis and data visualisation, the conventionally used libraries in Python are numpy [49], scipy [50], scikit-learn [20], and matplotlib [21]. Numpy provides a powerful and efficient n-dimensional array object (often used as requirement for other libraries), and functions to perform mathematical operations over real values, vectors (1-dimensional arrays), and matrices (2-dimensional arrays) such as scalar/vector/matrix addition, multiplication, extracting columns of a matrix to a vector, and boolean filtering [49]. Scipy is a library that extends numpy to provide additional domain-specific functions, providing tools such as sparse matrices and implementations of statistical equations such as estimating distributions [50]. Scikit-learn provides implementations of algorithms for data analysis, feature extraction, and machine learning, such as the CountVectoriser used to compute term frequencies [20]. Matplotlib is a 2D plotting library that produces visual graphs from lists/arrays [21]. It provides the capability to generate various types of plots, such as scatterplots, line plots, histograms, and box-and-whisker plots; modify the plot parameters (such as colours, labels, and bounds), create a grid of axes and plot multiple graphs in the same axes, generate a legend or colorbar, among other features.

The types of plots and statistical analysis metrics that are deemed relevant for this analysis are:

- Scatter plot, used to show the distribution of data within two variables (e.g. year published and sentiment score), and colour could be added to show a third variable (e.g. source/publisher of article). Available on Matplotlib.
- Histogram (and 2D histogram), used to show the distribution of articles relative to variables such as publisher, year of publication, and sentiment score ranges. Available on Matplotlib.
- Box-and-whiskers [51] and violin plot [52], also used to show and compare the distribution of dependent variables (e.g. sentiment score) within different subsets of the data separated by independent variables (e.g. publisher). Available on Matplotlib.
- Line graph, used to show trends in a dependent variable (e.g. sentiment score) over an independent variable (e.g. year of publication). Available on Matplotlib.
- Mean and standard deviation, to quantify the distribution of articles within different subsets of the data separated by independent variables (e.g. publisher and year of publication) and provide a quantitative measure to compare different subsets. Available on SciPy.
- Mann-Whitney U Test [53], a statistical metric that measures whether it is true that given a randomly-selected value from a distribution, and another randomly-selected value from another distribution, the first value is equally likely to be less than or greater than the second value (i.e. there are no statistically significant difference between the two distributions), to show if the difference between two subsets are statistically significant. Available on SciPy.

Chapter 3

Requirements and Analysis

3.1 Problem Statement

The primary aim of this project is to utilise available natural language processing techniques in order to perform a high-level meta-analysis of online news articles relating to people with disabilities available in the online British news media, with the goal of revealing trends by varying for independent variables such as source and year published. To achieve this aim, the solution would need to collect and scrape online news articles from the Internet, filter only relevant articles, use available natural language processing tools to extract information from these articles, perform statistical analyses, and show visualisations of the resulting data to show trends. Of particular interest (as a dependent variable) is a sentiment (or opinion, polarity) index of articles (i.e. how positively does an article view people with disabilities?), and how it varies given independent variables such as source and year published. Thus, the general solution defined by this aim would involve the completion of several sub-problems, primarily data collection, filtering, parsing, sentiment scoring, and statistical analysis and visualisation.

The goal of this project is to develop a computational pipeline that implements the general solution. Before the project could be started, the topics relevant to this project has to be defined. Thus, a list of topics relating to the domain of people with disabilities would need to be compiled, each topic consisting of a list of keywords (or key phrases) and query terms related to a specific disability (or disabilities in general). Given this list of topics, this pipeline has to implement these following tasks:

- Web crawling and scraping web pages, using public APIs where possible, to collect a dataset of news articles, given the list of query terms for each topic.
- Filtering relevant articles from the dataset, given the list of keywords for each topic.
- Extracting relevant sentences that refers to a keyword, from the dataset of filtered articles
- Performing sentiment analysis (using open-source libraries and pre-trained models) on the dataset of relevant sentences
- Producing relevant statistical analyses and data visualisation to show trends over independent variables such as source and year published.

As the primary advantage of computational natural language processing is in its speed relative to human reading, the solution must be able to perform these computations quickly and at scale.

The total number of documents (online news articles) available (given Daily Mail, Daily Express, and Guardian as sources) is expected to number in the thousands to tens of thousands of documents per topic, or hundreds of thousands of documents in total across all topics. Given this scale, the solution must be able to compute the full pipeline within a feasible timeframe (i.e. not more than a few days), given available general-purpose hardware (Intel i7-6700HQ CPU @ 2.60GHz, NVIDIA GeForce GTX 1060 GPU) to show that the solution is feasible without specialised hardware.

3.2 Requirements

The solutions follows a component-based design, where each sub-problem must be implemented by a component that focuses only on the sub-problem. These components must be linked together via a 'pipeline' script that executes each component in order, and iterates through the list of topics and sources. This is ideal such that changes could be made to a component without affecting (or needing to re-write) code in other components. The list of required components are as follows: data collection, dataset filtering, rule-based sentence matching, sentiment scoring, statistical analysis and visualisation.

3.2.1 Data Collection

The data collection component must be able to find a list of articles related to each topic (a list of query terms) for each supported source (Daily Mail, Daily Express, and Guardian). For each article, the only information required at this stage is a working URL pointing to an online resource containing the article text and relevant metadata. Thus, the information that has to be provided by the component after this stage is a list of URLs pointing to relevant articles given a list of search queries.

Then, after the list of URLs pointing to Web-based news articles has been compiled, the component must be able to scrape these articles and extract the full article text and relevant metadata from each URL. Aside from the full article text, the relevant metadata that needs to be extracted are the article's headline, URL, date of publication, and publication source. The information is returned in the form of an array of JSON objects, with each JSON object containing information (text and metadata) about a single news article. This information must then be saved locally to a file where it will be read by the next component. The file and directory structure must be consistent given source and topic (such that the next component can find it programmatically).

As data collection is expected to consume the longest time (due to the necessity to submit a web request for each article) compared to other components, additional requirements regarding scalability are enforced. The data collection component must run in reasonable time (i.e. less than a day for each topic), given available hardware and a scale of up to tens of thousands of articles per topic. Additionally, it should be possible to resume progress on data collection, such that it is possible to re-run the program at a later date to add new articles without sending web requests for articles already in the collection. It would also help in cases where the program is interrupted, Internet connection is lost, the machine is shut down, etc. Thus, the component should be able to store already-parsed articles to an external file, and read from the external file upon starting to gather a list of already-parsed URLs, and avoid re-parsing existing URLs.

3.2.2 Dataset Filtering

The next component handles dataset filtering. Initially, this component must be able to load the dataset of parsed articles from the file saved by the data collection component. Each file contains a subset of all parsed articles for a specified topic and a specified source. For each article, it must decide whether it is relevant to the topic defined by the subset, given a list of keywords and key phrases unique to each topic. This decision should be based by the article's full text and headline, and should take all keywords and key phrases associated to the topic into account. It should also be able to show/print a sample of an arbitrary number of documents, which would be used to analyse and improve the accuracy of the filter.

Then, it must save all articles deemed on-topic (relevant to the topic) to a new output file, containing an array of JSON objects in the same format and information as in the data collection component's output file. Articles deemed off-topic (not relevant to the topic) must not be saved to the output file. The file and directory structure of the output file must be consistent given source and topic (such that the next component can find it programmatically).

3.2.3 Text Parsing and Rule-based Sentence Matching

The next component parses the article text using open-source natural language processing tools to extract relevant information required by the sentiment scoring and statistical analysis components. Given an input file which is the output file of the dataset filtering component, this component should be able to load the article text and metadata of all saved articles. Then, information should be extracted by syntactically parsing each article's text and headline. The sentiment scoring component would require all sentences relating to a keyword or key phrase (containing a keyword, key phrase, or an equivalent term) to be extracted from each article. Additionally, the component should also extract other information as required by the sentiment scoring and statistical analysis components, including the total number of sentences and the number of relevant sentences in the document, and the term frequency of each keyword and key phrase.

After these information has been extracted from the article, it must save the information to a new output file, in the form of an array of JSON objects where each JSON object contains all the information about a single news article (with a key for each 'feature' e.g. relevant sentences). The metadata of each article (headline, date of publication, and source) should also be saved to the new output file, as it would be required as independent variables for the statistical analysis component to analyse trends in the dataset. The file and directory structure of the output file must be consistent given source and topic (such that the next component can find it programmatically).

3.2.4 Sentiment Scoring

The sentiment scoring component computes a real-valued score for each relevant sentence extracted by the previous component, which must correspond to the perceived 'sentiment' of the sentence towards a disability, disabilities, a person with disabilit(y/ies), or people with disabilit(y/ies), referred by the keyword or key phrase, with sufficient accuracy. Given an input file which is the output file of the dataset filtering component, this component should be able to load the relevant sentences and other data for all saved articles.

3.2.4.1 Comparison of open-source implementations

Then, two iterations of the sentiment scorer component must be developed. The first iteration of the sentiment scorer component is used to perform a comparison between several open-source sentiment analysis implementations for this sentiment scoring task. It is not used in the final pipeline

This iteration of the component must select a sample of sentences from the dataset, with a sample size arbitrarily defined by the user. Also, it must implement all open-source sentiment analysis implementations ('sentiment scorer') that were listed in section 2.3.3. All sentences in the sample must be analysed and given a score by each sentiment scorer, and the component should also allow the user to manually label these sentences as positive, neutral, or negative. With this information, the component must be able to determine the accuracy of each sentiment scorer (i.e. show the proportion of true positives and true negatives, and show a confusion matrix), and store the total, per-sentence, and per-article runtime of each sentiment scorer.

This iteration of the component will not be used in the final pipeline, but only used as a tool to compare existing sentiment scorer implementations for the domain of sentences in news articles relating to disabilities or people with disabilities.

3.2.4.2 Final implementation

The second iteration of the sentiment scorer component is used in the final pipeline. This component must perform sentiment analysis for all relevant sentences in the dataset (instead of only a limited sample of sentences). Instead of computing multiple sentiment scores of every sentence using all implemented sentiment scorers, it must only compute one sentiment score using the best-performing sentiment scorer that runs in reasonable time, as shown by the first iteration of this component. It should also compute the average sentiment score for each article, given information of the sentiment scores for all relevant sentences in the article. Furthermore, the component must run in reasonable time (i.e. less than a day for each topic), given available hardware and a scale of up to tens of thousands of articles per topic.

After the sentiment scorer has scored all relevant sentences in the dataset, it must then save information about all articles, with score labels appended to each sentence, to a new output file. The JSON object format of each article in the output file should be equivalent to the input file's format (i.e. no information from previous components are lost), with the exception of an additional 'sentiment score' key-value pair within each sentence's JSON object. The file and directory structure of the output file must be consistent given source and topic (such that the next component can find it programmatically).

3.2.5 Statistical Analysis and Visualisation

The last component performs statistical analysis and visualisation. This component must read the sentiment scoring component's output files as input files, to load the dataset of news articles and features extracted by previous components about each article. At this point, information extracted about each article by previous components must include: source, publication date, and sentiment score, alongside other information. To combine information about articles from different sources (represented by different files), this component should be capable of reading input from

several different input files in a single run, and collate the information about every unique article in each file to a single dataset.

This component must be able to show trends, visualisations, statistical metrics that show how dependent variables (e.g. sentiment score) differ relative to the independent variables (e.g. publication year and source). The plot types and statistical metrics relevant to this analysis was defined in the end of section 2.3.4. At a minimum, this component must show how the sentiment score varies relative to publication year and source, and how subsets divided by publication year and source differ in distribution of sentiment scores, using the plots and metrics as defined in section 2.3.4. Additionally, the component should also show visualisations based on other information extracted by previous components as relevant, such as trends in the term frequency of each keyword and key phrase (as a dependent variable) over time (publication year, as an independent variable).

3.3 Analysis of Requirements

The implementation design (and additional requirements) of this project are highly influenced by the core requirements (i.e. data collection, filtering, parsing, sentiment scoring, and statistical analysis and visualisation). The overall design of the pipeline, with isolated components for each sub-problem, stemmed from having largely independent sub-problems that was required in order to perform the data analysis 'end-to-end' (i.e. from data collection to analysis and visualisation) Also, initial prototypes (and initial 'runs' collecting only a limited number of news articles) showed that data collection took the majority of runtime in the pipeline. For this reason, the requirement where it should be possible to resume progress on data collection (and not repeat the process for existing data) was added.

Furthermore, the format of JSON objects of each output file were largely based on the information extracted by each component, as defined by the requirements. For example, the format of a JSON object in the output file for the data collection component is as follows (each JSON object represents one article in the dataset):

```
{
        "https://www.express.co.uk/comment/columnists/...": {
                "source": "Daily Express",
                "title": "Will she grow out of her stutter?",
                "datetime": "2008-02-12T00:00:00+00:00",
                "section": "comment",
                "subsection": "columnists",
                "text": "..."
        }
}
```

Where the URL and 'source', 'title', 'datetime', and 'text' fields correspond to the requirements of what information needs to be extracted by the data collection component.

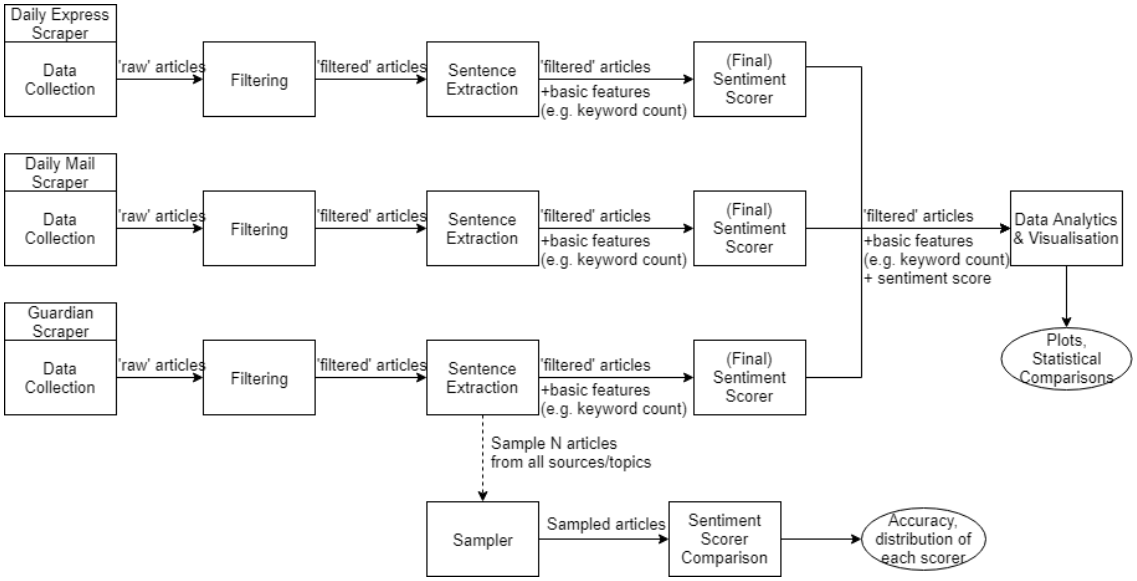
Chapter 4

Design and Implementation

4.1 Overall Design

To fulfil the stated requirements in chapter 3, the solution that was decided is a computational pipeline consisting of five main components. This solution was used to carry out the experiment to collect and analyse online news articles and visualise potential trends in sentiment/opinion. The results of this experiment is shown in chapter 5.

The high-level design of the pipeline and its main components are defined as follows:



In this experiment, the pipeline is ran once for each source and topic, generating a dataset of all articles for the specified topic from the source with all extracted features, stored in a file. The 'plot' component then loads these files and combines the dataset of all articles for every source within the same topic, and produces various plots to show trends within the topic, and statistical metrics for each possible subset within the topic.

Components pass datasets to each other by storing all information it extracts in a JSON format. The output file of an earlier component is read as the input file for the next component, given the same source and topic. The format of the output/input files is a collection of JSON

objects, where each line consists of a single JSON object, and each JSON object represents a single article. The JSON object contains a key-value pair for each extracted feature, and each component preserves the previous component's information (key-value pairs) while appending the object with additional key-value pairs for each feature it extracts. This design ensures that in case computation is interrupted for any reason, the dataset generated by the previous component has already been saved to a file, and computation can restart from the interrupted component (without having to re-run previous components).

4.2 Dataset Description

4.2.1 Sources

The Internet is an increasingly common medium for news publishers to publish news articles and for consumers to read these articles. As of 2017, 64% of British individuals read and/or download online news, newspapers or magazines, a sharp increase from 20% in 2007 [54].

The Daily Mail (`dailymail.co.uk`) and the Guardian (`theguardian.com`) are the two most visited news publisher's web sites as of 2016, with a monthly viewership of 11.85 and 10.05 million respectively [55], and are the main subjects of this experiment. Additonally, The Daily Express (`express.co.uk`), a slightly smaller newspaper with a monthly viewership of 2.67 million in 2016 [55], is also added as a news source in this experiment, to control for source size and explore how the experiment performs with smaller sample sizes. Furthermore, the Guardian provides an API [56] and the Daily Mail and Daily Express provide advanced search tools [57], [58] to query news articles from their websites, which prove useful in their respective data collection component's implementation, although the pipeline would work with any online news source that provides an internal search tool. Thus, these three news publisher sources (Daily Mail, Guardian, Daily Express) were selected the main subjects of this experiment.

4.2.2 Topics, keywords, key phrases, and query terms

The final list of topics relevant to disabilities and people with disabilities, and the keywords, key phrases, and query terms deemed relevant to each topic, are:

Topic	Keywords and Key Phrases	Query Terms
'disabled'	'disabled', 'disability', 'handicapped', 'cripple', 'invalid', 'accessible', 'ablism', 'ableism', 'differently abled'	'disabled', 'disability', 'ablism', 'ableism', 'differently abled'
'autism'	'autism', 'autistic', 'asperger\'s', 'ASD'	'autism', 'autistic', 'asperger\'s', 'ASD'
'blind'	'blind', 'blindness', 'blindism', 'visual impairment', 'partially sighted', 'vision loss'	'blind', 'blindness', 'visual impairment', 'partially sighted', 'visually impaired'

'cerebral palsy'	'cerebral palsy', 'spastic'	'cerebral palsy', 'spastic'
'deaf'	'deaf', 'deafness', 'hearing impaired', 'hard of hearing', 'hearing loss'	'deaf', 'deafness', 'hearing impairment', 'hard of hearing', 'hearing impaired'
'developmental delay'	'developmental delay', 'developmental disability', 'developmental disorder', 'learning disability', 'slow learner', 'retardation', 'intellectual disability'	'developmental delay', 'developmental disability', 'developmental disorder', 'learning disability'
'dyslexia'	'dyslexia', 'dyslexic'	'dyslexia', 'dyslexic'
'epilepsy'	"epilepsy', 'epileptic', 'seizure'	'epilepsy', 'epileptic'
'mental illness'	'mental illness', 'mental health', 'mental disability', 'mental disorder', 'mental issue', 'brain injured', 'brain injury', 'brain damaged', 'psychological', 'psychiatric', 'emotional disorder', 'behavioural disorder', 'retardation', 'intellectual disability', 'mentally ill', 'mentally disabled', 'mentally handicapped'	'mental illness', 'mental health', 'mental disorder', 'mental disability', 'mentally ill', 'mentally disabled', 'mentally handicapped'
'mute'	'mute', 'muteness', 'mutism', 'cannot speak', 'difficulty speaking', 'synthetic speech', 'non-vocal', 'non-verbal'	'mute', 'muteness', 'mutism', 'non-verbal'
'paralysis'	'paraplegic', 'quadriplegic', 'spinal cord', 'paraplegia', 'quadriplegia', 'paralysed', 'paralyzed', 'paralysis', 'crippled', 'leg braces', 'wheelchair'	'paraplegic', 'quadriplegic', 'paraplegia', 'quadriplegia', 'paralysis'

'speech impairment'	'speech impairment', 'stutter', 'speech disability', 'speech disorder', 'communication disability', 'difficulty speaking', 'language impairment', 'language disorder', 'language disability', 'speech impediment', 'stammer'	'speech impairment', 'stutter', 'speech disorder', 'speech impediment'

This list of key terms was roughly based on guidelines from the Californian [59] and UK [60] governments (ignoring whether the term is considered 'appropriate' or 'inappropriate', as terms labelled 'inappropriate' are often still commonly used in the news, and relevant to consider when deciding whether an article is on-topic), with a few additions based on other commonly-used terms found on sampled articles from Daily Express, Daily Mail, and Guardian. The list of query terms were based on the list of key words, with words/phrases that could have other meanings (or combinations of multiple common words) removed (unless the word/phrase is very commonly used to refer to the disability, such as 'blind' and 'mute').

4.2.3 Dataset Size

The dataset is comprised of all articles found online given the query terms defined in section 4.2.2, published between 2000 to (approximately) late-March/early-April 2018. The size of the initial collected dataset (i.e. all articles collected by the Data Collection component, prior to any further processing) in number of articles, for each source and topic, were:

Topic	Daily Express	Daily Mail	Guardian	Total
Disabled	16,818	24,768	30,598	72,184
Autism	988	6,035	5,780	12,803
Blind	9,467	23,616	32,307	65,390
Cerebral Palsy	509	1,995	1,569	4,073
Deaf	6,795	20,686	8,163	35,644
Developmental Delay	965	3,529	1,517	6,011
Dyslexia	283	938	1,980	3,201
Epilepsy	700	2,924	2,368	5,992
Mental Illness	8,102	38,273	29,831	76,206
Mute	1,541	2,312	4,764	8,617
Paralysis	711	4,346	3,879	8,936
Speech Impairment	1,777	2,994	1,357	6,128
Total	48,656	132,416	124,113	305,185

After filtering, the size of the dataset that was plotted, in number of articles for each source and topic, were:

Topic	Daily Express	Daily Mail	Guardian	Total
Disabled	1,852	6,035	8,524	16,411
Autism	128	1,755	1,278	3,161
Blind	775	3,008	3,894	7,677
Cerebral Palsy	25	253	75	353
Deaf	114	747	901	1,762
Developmental Delay	5	130	447	582
Dyslexia	19	110	281	410
Epilepsy	58	740	374	1,172
Mental Illness	398	6,794	8,137	15,329
Mute	24	147	285	456
Paralysis	53	1,003	405	1,461
Speech Impairment	57	73	85	215
Total	3,508	20,813	24,646	48,967

For the results evaluation, the focus will be on the 'disabled' topic, as it has the highest amount of news articles within its subset, and is the most generalisable topic on disabilities (as it refers to the general theme of disabilities and people with disabilities, rather than a specific topic).

4.2.4 Limitations

As shown in section 4.2.3, the Daily Express has significantly less articles for any given topic than the Daily Mail or the Guardian, sometimes much less so. Some topics, such as cerebral palsy, developmental delay, dyslexia, mute, and speech impairment, are also severely lacking in sample size of articles (post-filter). In particular, cases where there are less than ~200 articles for a source within a topic were problematic to plot or form statistically-significant conclusions regarding trends (e.g. to compare with other sources), as the distribution of the data is too varied. In section 5.4, we show that it was difficult to obtain statistically significant conclusions when comparing to subsets with less than ~200 articles. This means that it is difficult to analyse or compare the Daily Express's subset for topics other than general 'disabled', 'blind', and 'mental illness', due to its relative lack of sample size.

Another limitation with this experiment is the (limited) length of time each news source (Daily Express, Daily Mail, and Guardian) retain articles for in their online archive. Our dataset indicates that by March/April 2018 (when this experiment took place), the Daily Express only retains articles from after ~2007, the Daily Mail only retains articles from after ~2010, and the Guardian only retains articles from after ~2000. This raises an issue when analysing year-on-year trends, as the proportion of articles' sources within a topic are different in each year, and year-on-year differences may be better explained due to this difference in proportion, rather than an actual trend (see also: section 5.3). For this reason, we consider trends over time to be significant only when the trend is consistently repeated for each source's subset (instead of the full dataset for that topic), for all sources with statistically-significant sample size in that topic.

4.3 Components

4.3.1 Data Collection

The data collection component is composed of two sub-parts. The first part is a scraper object unique to each supported source (see section 4.2.1) that defines two functions unique to each source:

- A function to return a list of URLs pertaining to news articles related to a specified query term. If there are multiple pages, the scraper needs to parse how many search pages exist for the query from the first page, and request each search page.
- A function that, given a URL pertaining to an article, parses the article to return the article text and metadata: headline, publication date/time, and publisher.

These two functions require separate implementations for each source, as each website has a different HTML structure (which tends to be consistent within the same source). Thus, the implementations required to parse a list of URLs from the search page and parse text from the article page are different for each source. The scraper object creates a wrapper for the implementations of these two functions to be injected into the main data collection script, such that the main script will work for any source, with the source-specific implementations of these functions abstracted away.

This design allows the use of a single data collection script (the second part of this component) for multiple sources, by having the methods that require separate implementations for each source (finding articles, scraping text and metadata) injected as a dependency to the script. Furthermore, as defined in the requirements (section 3.2.1), it should be possible to re-run the program at a later date to add new articles without sending web requests for articles already in the collection, such that it is possible to 'update' the collection and recover collected information in case computation is interrupted.

The data collection script first loads the existing output file for the source and category (if it exists) and loads a list of already-saved URLs. Then, for each query term in the category, it queries the search page(s) to gather a list of URLs. It then combines the URLs for all search terms in the category into one list. For each URL, if the URL is not saved already in the output file, it queries the URL for the article text and metadata, and stores it as a JSON object to an appended line in the output file. This design allows data collection to be resumed in case computation is interrupted, but with the drawback of technically violating the JSON standard for a valid JSON text to have only one top-level object per file [61].

Among all components in the final pipeline (i.e. not including sentiment-scorer comparison), data collection took the longest to compute. Results from preliminary runs of the experiments indicate that data collection took approximately 0.61 seconds per article for the Daily Express, 0.70 seconds per article for the Daily Mail, and 0.20 seconds per article for the Guardian. The full dataset of 305,185 articles took roughly a week to collect using a single general-purpose computer (Intel i7-6700HQ CPU @ 2.60GHz, NVIDIA GeForce GTX 1060 GPU) on a 200 Mbit/s (download speed) internet connection.

4.3.2 Dataset Filtering

Dataset filtering was implemented by measuring 'term frequency ranking', which measures how often does a key term (i.e. keyword or key phrase) appears in a text document relative to other terms. To compute this metric, the term frequency of every token (word) in the document has to be measured first. This is implemented using scikit-learn's [20] CountVectoriser, which tokenises a document and converts it into an array of term-frequency values for each token in the document. This array is then sorted in descending order, and then the position of the keyword in the sorted array (plus one) is recorded as the keyword rank of the document. A keyword rank of 5, for example, shows that the keyword is the 5^{th} most popular word in the document.

To use keyword rank for filtering documents, a constant threshold value is used to determine whether documents are on-topic or not. A document is on-topic if the keyword's rank is lower than the threshold value (i.e. the keyword is one of the most frequently used terms in the document), and is off-topic if the keyword's rank is higher than the threshold value (i.e. the keyword is rarely used in the document, relative to other terms). As the term frequency of the keyword is measured relative to other terms in the document, this metric also normalises for document length.

As in this experiment, each category can contain multiple key terms, slight modifications has to be made during pre-processing (before tokenisation). All mentions of any keyword or key term in the article are replaced by 'KEYWORD_TOKEN', and the filter measures the rank of 'KEYWORD_TOKEN' instead of any specific term, such that it doesn't matter which term the article uses. Additionally, stemming is also performed for every word in the document to reduce all words to its root forms, regardless of its word form or tense (e.g. 'disability' and 'disabilities' are both treated as the same word) using the nltk [18] library. Furthermore, CountVectoriser also ignores 'stop words', or meaningless common words in the English language (such as 'the' and 'a').

To evaluate the results of this filtering, the filtering component can be set to print an arbitrarily-sized sample of N articles alongside its predicted label (on-topic / off-topic) and keyword rank. Trivial evaluation (with a sample size of 10 articles per topic) indicated that for most topics, this simple metric works well in distinguishing between on-topic and off-topic articles. Manual reading of sampled articles indicated that:

- There exists a good of correlation between keyword rank and on-topic/off-topic articles (although the sample size is too low to make statistically-significant conclusions, as increasing the sample size would mean manually reading more articles which was time-consuming)
- Articles predicted to be 'on-topic' are on-topic, and articles predicted to be 'off-topic' are off-topic.

However, there are several topics where this filtering mechanism does not perform well, and as a result many off-topic articles remain in the filtered dataset. These topics are 'blind', 'mute', 'paralysis' ('paralysed'), and 'speech impairment' ('stutter'), where the keywords can have other meanings irrelevant to the context of people with disabilities. For example, consider these sampled sentences from articles that were mislabelled on-topic:

- When Harry met Meghan: How Prince Charles's family friend set up blind date.
- Kate Garraway gets flustered as she struggles to mute her ringing phone live on air.
- Snooker: Higgins stutters then stages another late comeback.

As term frequency does not distinguish between multiple meanings of words, it is unlikely that

this limitation could be solved using any term frequency based approach, or a similar syntactic approach (such as tf-idf and word vectors).

Several other alternatives were considered to term-frequency ranking. The conventional approach is by calculating term frequency – inverse document frequency (tf-idf) [32], [33], which builds on term frequency by weighing more 'common' words with lower scores and vice versa. Using tf-idf, terms that appear more frequently in the corpus of all documents are deemed to be 'less important' and assigned a lower weight, and vice versa. However, this approach is unsuitable for this project, due to the selective nature of the dataset (where only articles containing certain query terms are collected). This will cause the idf weights of these query terms to be much lower than it should be, because at least one of the query terms would appear on every document in the corpus, highly increasing the document frequency of query terms. For this reason, limited evaluation showed that tf-idf ranking performed worse than term-frequency ranking.

A possible improvement to the term-frequency ranking model is by using a supervised machine learning model to classify documents to 'relevant' and 'irrelevant' (a boolean classification problem). There are many supervised approaches that are popular for text classification, including Support Vector Matrices (SVM), Naïve Bayes (NB), and k-nearest neighbour (kNN) models [34]. However, this approach would require labelled data (i.e. a collection of articles with 'relevant' and 'irrelevant' labels for each topic).

4.3.3 Rule-based Sentence Matching

The next component parses the article text using SpaCy [36] to extract relevant information required by the sentiment scoring and statistical analysis components. Initially, the headline is prepended to the article text to create the text document. SpaCy performs a syntactic analysis of the document, returning a list of 'enhanced' tokens (words) which contain additional information parsed by SpaCy, such as part-of-speech tags, syntactic parent/children, known entities, sentence start and end, and root word (lemma). The component iterates over this list of 'enhanced' tokens to extract information that would be useful for statistical analysis:

1. Sentences containing a word relating to a keyword or key phrase (using sentence matching), and the number of keywords / key phrases in each sentence.
2. For each token that has the same lemma as a keyword (or a sequence of tokens that has the same sequence of lemmas as a key phrase), the frequency of the token or sequence of tokens.
3. Number of relevant sentences in the document (i.e. number of occurrences of point 1).
4. Total number of sentences in the document.
5. Number of keyword occurrences in the document (i.e. number of occurrences of point 2).
6. Total number of tokens in the document.

Points 3-6 were used to measure a relevance score of each article, however that score is currently unused by the statistical analysis component. Future work could expand on this concept.

The sentiment scoring component requires all sentences relating to a keyword or key phrase to be extracted from each article. SpaCy provides a rule-based Matcher tool that returns the indices of tokens (or sequences of tokens) that fulfils a given 'rule'. This component uses the Matcher tool to find all indices of tokens that has the same lemma as a keyword (or sequences of tokens that has the same sequence of lemmas as a key phrase). Then, for each token, the component retrieves

the full sentence that contains the token index and stores them into a list of relevant sentences, along with the number of keywords / key phrases in each sentence. These relevant sentences are stored in an array of JSON objects, with each JSON object representing one sentence, such as:

```
"Don't be disabled in spirit as well as physically.\"": {
        "keyword_count": 1
}
```

This array is stored as a value within the parent article's JSON object.

4.3.4 Sentiment Scoring

The sentiment scoring component implements open-source libraries to measure the 'sentiment' (ideally, perceived view of the sentence towards a disability, disabilities, a person with disabilit(y/ies), or people with disabilit(y/ies) as referred by the keyword or key phrase) of news articles. Sentiment scores are real-valued scores (i.e. a score of 0.0 indicates that a sentence is neutral, a highly positive or highly negative score indicates the sentence has strong positive or negative opinions), capped between -1.0 and 1.0.

Two implementations of the component were developed:

- The first iteration of the sentiment scorer component is used to perform a comparison between several open-source sentiment analysis implementations for this sentiment scoring task. This iteration computes sentiment scores of all implemented scorers for every article, and is ran only on an arbitrarily-sized sample of the full dataset, as running all sentiment scorers without optimisation takes roughly ~1 minute per article which is infeasible for the full dataset.
- The second iteration of the sentiment scorer component is used in the final pipeline. This iteration only computes one sentiment score per each article, and is highly optimised. Two iterations of this component has been tried: one implementing OpenAI's [45] sentiment scorer, and another implementing VADER [40]. In both cases, the runtime of is lower than 0.2 seconds per article.

4.3.4.1 Comparison of open-source implementations

This component consist of two scripts: The first script is the sampler. For each topic and source in the parsed dataset, it loads a small sample of articles. It then loads all relevant sentences from the sample of articles to a combined list of relevant sentences from each source and topic. Then, it loads an arbitrarily-sized sample of relevant sentences for each source and topic (the experiment used 5 sentences per source and topic * 12 topics * 3 sources = 180 sentences).

Every sentence in the sampled are then scored using 7 open-source sentiment analysis implementations that were explored in section 2.3.3:

- VADER [40], a simple rule-based model that scores sentences based on the presence of 'sentiment lexicons' (a list of sentiment-related words, emoticons, and acronyms), and rules such as negations and emphasis (e.g. punctuation, capitalisation, 'very').
- vivekn's 'sentiment' repository, which provides an enhanced Naïve Bayes classifier model trained on a publicly available dataset of 25,000 highly-polar movie reviews from the Internet Movie Database (IMDb) using bigrams and trigrams as features, an implementation of [41].

- xiaohan2012's 'twitter-sent-dnn' repository, which provides a convolutional neural network model with dynamic k-max pooling (DCNN) using word and n-gram order as 'features', trained on a dataset of 1.6 million tweets with emoticon-based labels. It is an implementation of [42].
- kevincobain2000's 'sentiment_classifier' repository [44], which provides a supervised machine learning model based on a Naïve Bayes and Maximum Entropy Classifier, performs Word Sense Disambiguation [43], and uses bigrams as features (weighted by word occurrence statistics). Its training data is a mixture of nltk's movie review corpus, Twitter posts, and Amazon customer reviews data.
- OpenAI's 'generating-reviews-discovering-sentiment' repository provides a pre-trained single-layer multiplicative LSTM recurrent neural network model trained on a dataset of over 82 million Amazon product reviews, using bytes of a UTF-8 encoded sentence as 'features', an implementation of [45].
- Stanford CoreNLP [19] provides a recursive neural network model that converts text to parse trees, used as features. It was trained on a dataset of fully-labelled parse trees for 11,855 sentences from the Rotten Tomatoes movie review corpus. It is an implementation of [46]. Unlike other scorers in this list, it classifies sentences into five sentiment classes, from 'very negative' to 'very positive', instead of assigning a real-valued score. The stanfordcorenlp package [47] was used to start and query a Stanford CoreNLP local server in Python.
- TextBlob's PatternAnalyzer, a rule-based model based on part-of-speech pattern matching [48].
- TextBlob's NaiveBayesAnalyzer, a Naïve Bayes classifier model trained on a dataset of movie reviews (no information on features or dataset size) [48].

(Section 2.3.3 has more details on the implementation details of these repositories/libraries)

The scores of every sentence in the sample (of 180 sentences in the experiment) is stored in a JSON object similar to:


```
{
        "sentence": "The autism gender trap.",
        "label": "-",
        "sentiment_score_openai": -0.24175840616226196,
        "sentiment_score_vader": -0.3182,
        "sentiment_score_xiaohan": -0.9157504061450769,
        "sentiment_score_kcobain": -0.5,
        "sentiment_score_stanford": -0.5,
        "sentiment_score_textblob": 0.0,
        "sentiment_score_textblob_bayes": -0.9139802175212899
}
```


The JSON objects of the sample is then output to a text file, where a user can manually change the 'label' fields to either '+' (positive), '-' (negative), 'n' (neutral), or 'o' (off-topic) to be read by the second script. To ensure there is no bias in manual labelling, a second output file which contains only sentences and labels (without sentiment scores) were used.

The second script is the analyser. Given a dataset of labels and sentiment scores (with the

format shown above), it plots the sentiment score distribution of positive, negative, and neutral sentences in seven histograms (one for each sentiment scorer). Additionally, it also computes these statistics for each sentiment scorer:

- Mean positive: mean sentiment score for all positive-labelled sentences
- Mean neutral: mean sentiment score for all neutral-labelled sentences
- Mean negative: mean sentiment score for all negative-labelled sentences
- True positive: count of positive-labelled sentences with sentiment score > 0.0
- False positive: count of positive-labelled sentences with sentiment score ≤ 0.0
- True negative: count of negative-labelled sentences with sentiment score ≤ 0.0
- False negative: count of negative-labelled sentences with sentiment score > 0.0
- Accuracy: (True positive + false positive) / (count of all positive or negative sentences)

The results of this sentiment scorer comparison is documented in section 5.2.

4.3.4.2 Final implementation

The results of the sentiment scorer comparison, shown in section 5.2, show VADER [40] and OpenAI's [45] as the two best sentiment scoring tools for this domain (measuring perceived 'sentiment' of a sentence towards a disability, disabilities, a person with disabilit(y/ies), or people with disabilit(y/ies)).

Unlike the iteration used for sentiment scorer comparison, this iteration only computes one sentiment score per each article, and is highly optimised. Two versions of this iteration were developed: one implementing OpenAI's sentiment scores, and another implementing VADER sentiment scores. In the final pipeline used for the experiment, only the version using VADER was used, as VADER scores was shown to be better at displaying trends separating different subsets than OpenAI scores (section 5.2).

Within the pipeline, this component's task is to compute sentiment score information for every relevant sentence, and append that information on each sentence's JSON object:

```
"Don't be disabled in spirit as well as physically.\"": {
        "keyword_count": 1,
        "sentiment_score": 0.4215
}
```

Additionally, this component also computes the sentiment score of each article, which is defined as the weighted average of sentiment scores of relevant sentences contained within the article:

$$Article's\ sentiment\ score = \frac{\Sigma(sentiment\ score * number\ of\ key\ terms) for\ all\ sentences}{Total\ number\ of\ keyword\ occurrences\ in\ the\ article}$$

This iteration is highly optimised for runtime: it only computes one sentiment score for each sentence, using one sentiment model, instead of loading and analysing every sentence with all seven sentiment models as in section 5.2. For the version that implement OpenAI's sentiment model, it is further optimised by 'batching' the call to the sentiment model: instead of calling 'openai_model.transform()' for every sentence, it builds a corpus (list) of all sentences in all articles (from the loaded file – each file represents all articles from a source for a topic), and calls 'openai_model.transform()' once, passing the corpus as parameter. Results from running the experiment indicate that sentiment scoring took approximately 0.16 seconds per article using the

OpenAI model, and 0.0031 seconds using VADER's implementation, using a single general-purpose computer (Intel i7-6700HQ CPU @ 2.60GHz, NVIDIA GeForce GTX 1060 GPU).

4.3.5 Statistical Analysis and Visualisation

Chapter 5

Results Evaluation

5.1 Focused topic

Although the pipeline was run on a list of twelve disability-related topics (as defined in section 4.2.2), this results evaluation will mainly focus on the 'disabled' topic. The 'disabled' topic was chosen as it has the highest amount of news articles within its subset, and it is the most generalisable topic (as it refers to the general theme of disabilities and people with disabilities, rather than a specific topic) (see also: section 4.2.3). This focus on one topic for the whole of Chapter 5 helps keep the numbers and results being discussed consistent throughout Chapter 5. Furthermore, the full result plots for all topics will be available in the Appendix.

However, at several points in this chapter, results from other topics will also be discussed where it would add to the discussion. Information from other topics will be explicitly mentioned (e.g. 'for the ... topic') such that the reader understands where the data does not refer to the 'disabled' topic.

Year-on-year trends are also a consistent theme throughout parts of this chapter. Within the 'disabled' topic, the (post-filtering) sample size that was obtained for each year between 2000 and 2018 is as follows:

Year	Daily Express	Daily Mail	Guardian	Total
2000	0	0	504*	504
2001	0	0	294	294
2002	0	0	334	334
2003	0	2	330	332
2004	0	2	387	389
2005	0	0	381	381
2006	0	0	338	338
2007	51	0	424	475
2008	86	0	424	510
2009	175	0	352	527
2010	157	173	365	695
2011	166	370	607	1,143

2012	238	516	822	1,576
2013	133	473	638	1,244
2014	140	846	544	1,530
2015	177	1,095	526	1,798
2016	260	1,128	640	2,028
2017	220	1,094	506	1,820
2018**	49	336	108	493
Total	1,852	6,035	8,524	16,411

*'2000' also includes a small amount of articles published before the year 2000.

**2018 is incomplete and would only include articles up to (approximately) late-March/early-April.

5.2 Comparison of sentiment scorers

4.3.4.1

5.3 Sentiment score: plots and trends

5.4 Sentiment score: statistical comparison of different sources

5.5 Key terms: plots and trends

Chapter 6

Conclusions

6.1 Achievements

6.2 Evaluation

6.3 Future Work

Bibliography

[1] E. Cambria and B. White, "Jumping NLP curves: A review of natural language processing research," *IEEE Computational intelligence magazine*, vol. 9, no. 2, pp. 48–57, 2014.

[2] T. Lansdall-Welfare, S. Sudhahar, J. Thompson, J. Lewis, F. N. Team, N. Cristianini, A. Gregor, B. Low, T. Atkin-Wright, M. Dobson, *et al.*, "Content analysis of 150 years of British periodicals," *Proceedings of the National Academy of Sciences*, vol. 114, no. 4, E457–E465, 2017.

[3] Z. Waseem, "Are you a racist or am I seeing things? Annotator influence on hate speech detection on Twitter," in *Proceedings of the first workshop on NLP and computational social science*, 2016, pp. 138–142.

[4] J. Coverdale, R. Nairn, and D. Claasen, "Depictions of mental illness in print media: A prospective national sample," *Australian & New Zealand Journal of Psychiatry*, vol. 36, no. 5, pp. 697–700, 2002.

[5] N. Gold and G. K. Auslander, "Media reports on disability: A binational comparison of types and causes of disability as reported in major newspapers," *Disability and rehabilitation*, vol. 21, no. 9, pp. 420–431, 1999.

[6] O. F. Wahl, "Mass media images of mental illness: A review of the literature," *Journal of Community Psychology*, vol. 20, no. 4, pp. 343–352, 1992.

[7] K. Scior, "Public awareness, attitudes and beliefs regarding intellectual disability: A systematic review," *Research in Developmental Disabilities*, vol. 32, no. 6, pp. 2164–2182, 2011.

[8] K. Devotta, R. Wilton, and N. Yiannakoulias, "Representations of disability in the Canadian news media: A decade of change?" *Disability and rehabilitation*, vol. 35, no. 22, pp. 1859–1868, 2013.

[9] S. C. Jones and V. Harwood, "Representations of autism in Australian print media," *Disability & Society*, vol. 24, no. 1, pp. 5–18, 2009.

[10] J.-B. Michel, Y. K. Shen, A. P. Aiden, A. Veres, M. K. Gray, J. P. Pickett, D. Hoiberg, D. Clancy, P. Norvig, J. Orwant, *et al.*, "Quantitative analysis of culture using millions of digitized books," *science*, vol. 331, no. 6014, pp. 176–182, 2011.

[11] F. W. Gibbs and D. J. Cohen, "A conversation with data: Prospecting Victorian words and ideas," *Victorian Studies*, vol. 54, no. 1, pp. 69–77, 2011.

[12] M. Mauch, R. M. MacCallum, M. Levy, and A. M. Leroi, "The evolution of popular music: USA 1960–2010," *Royal Society open science*, vol. 2, no. 5, p. 150 081, 2015.

[13] K. Leetaru, "Culturomics 2.0: Forecasting large-scale human behavior using global news media tone in time and space," *First Monday*, vol. 16, no. 9, 2011.

[14] I. Flaounas, O. Ali, T. Lansdall-Welfare, T. De Bie, N. Mosdell, J. Lewis, and N. Cristianini, "Research methods in the age of digital journalism: Massive-scale automated analysis of news-content—topics, style and gender," *Digital Journalism*, vol. 1, no. 1, pp. 102–116, 2013.

[15] P. Gooding, "Mass digitization and the garbage dump: The conflicting needs of quantitative and qualitative methods," *Literary and linguistic computing*, vol. 28, no. 3, pp. 425–431, 2013.

[16] T. Schwartz, "Culturomics: Periodicals gauge culture's pulse," *Science*, vol. 332, no. 6025, pp. 35–36, 2011.

[17] M. Mitchell, K. Hollingshead, and G. Coppersmith, "Quantifying the language of schizophrenia in social media," in *Proceedings of the 2nd workshop on Computational linguistics and clinical psychology: From linguistic signal to clinical reality*, 2015, pp. 11–20.

[18] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[19] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *Association for Computational Linguistics (ACL) System Demonstrations*, 2014, pp. 55–60. [Online]. Available: `http://www.aclweb.org/anthology/P/P14/P14-5010`.

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[21] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Computing In Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007. DOI: `10.1109/MCSE.2007.55`.

[22] M. Xia. (Oct. 20, 2017). A curated list of sentiment analysis methods, implementations and misc., [Online]. Available: `https://github.com/xiamx/awesome-sentiment-analysis` (visited on 04/18/2018).

[23] Keon. (Apr. 16, 2018). A curated list of resources dedicated to natural language processing (NLP), [Online]. Available: `https://github.com/keon/awesome-nlp` (visited on 04/18/2018).

[24] J. Misiti. (Mar. 26, 2017). A curated list of awesome machine learning frameworks, libraries and software, [Online]. Available: `https://github.com/josephmisiti/awesome-machine-learning` (visited on 04/18/2018).

[25] P. S. Foundation. (2018). PyPi – the python package index, [Online]. Available: `https://pypi.org/`.

[26] Anaconda, Inc. (2018). Anaconda cloud, [Online]. Available: `https://anaconda.org/`.

[27] J. F. Puget. (Dec. 19, 2016). The most popular language for machine learning is ..., [Online]. Available: https://www.ibm.com/developerworks/community/blogs/jfp/entry/What_Language_Is_Best_For_Machine_Learning_And_Data_Science?lang=en.

[28] GitHub, Inc. (Apr. 18, 2018). Topic: nlp, [Online]. Available: https://github.com/topics/nlp (visited on 04/18/2018).

[29] Anaconda, Inc. (2018). What is Anaconda? [Online]. Available: https://www.anaconda.com/what-is-anaconda.

[30] K. Reitz. (2018). Requests: HTTP for humans, [Online]. Available: http://docs.python-requests.org/en/master.

[31] L. Richardson. (Aug. 11, 2017). Beautiful Soup, [Online]. Available: https://www.crummy.com/software/BeautifulSoup.

[32] S. Robertson, "Understanding inverse document frequency: On theoretical arguments for IDF," *Journal of documentation*, vol. 60, no. 5, pp. 503–520, 2004.

[33] K. Sparck Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of documentation*, vol. 28, no. 1, pp. 11–21, 1972.

[34] A. Khan, B. Baharudin, L. H. Lee, and K. Khan, "A review of machine learning algorithms for text-documents classification," *Journal of advances in information technology*, vol. 1, no. 1, pp. 4–20, 2010.

[35] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[36] Explosion AI. (2018). SpaCy: Industrial-strength natural language processing in Python, [Online]. Available: https://spacy.io.

[37] J. D. Choi, J. Tetreault, and A. Stent, "It depends: Dependency parser comparison using a web-based evaluation tool," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, vol. 1, 2015, pp. 387–396.

[38] B. Liu, "Sentiment analysis and opinion mining," *Synthesis lectures on human language technologies*, vol. 5, no. 1, pp. 1–167, 2012.

[39] GitHub, Inc. (Apr. 18, 2018). Topic: sentiment-analysis, [Online]. Available: https://github.com/topics/sentiment-analysis (visited on 04/18/2018).

[40] E. Gilbert and C. Hutto, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," in *Eighth International Conference on Weblogs and Social Media (ICWSM-14)*, 2014. [Online]. Available: http://comp.social.gatech.edu/papers/icwsm14.vader.hutto.pdf.

[41] V. Narayanan, I. Arora, and A. Bhatia, "Fast and accurate sentiment classification using an enhanced naive bayes model," in *International Conference on Intelligent Data Engineering and Automated Learning*, Springer, 2013, pp. 194–201.

[42] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, vol. 1, Baltimore, USA, 2014.

[43] S. Banerjee and T. Pedersen, "An adapted lesk algorithm for word sense disambiguation using wordnet," in *International Conference on Intelligent Text Processing and Computational Linguistics*, Springer, 2002, pp. 136–145.

[44] P. Khaturia. (Jan. 20, 2018). Sentiment classification using word sense disambiguation, [Online]. Available: `https://github.com/kevincobain2000/sentiment_classifier` (visited on 04/18/2018).

[45] A. Radford, R. Józefowicz, and I. Sutskever, "Learning to generate reviews and discovering sentiment," *CoRR*, vol. abs/1704.01444, 2017. arXiv: `1704.01444`. [Online]. Available: `http://arxiv.org/abs/1704.01444`.

[46] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, 2013, pp. 1631–1642.

[47] Lynten. (Feb. 14, 2018). Python wrapper for Stanford CoreNLP, [Online]. Available: `https://github.com/Lynten/stanford-corenlp` (visited on 04/18/2018).

[48] S. Loria. (2018). Textblob: Simplified text processing, [Online]. Available: `http://textblob.readthedocs.io/en/dev`.

[49] T. E. Oliphant, *A guide to NumPy*. Trelgol Publishing USA, 2006, vol. 1.

[50] E. Jones, T. Oliphant, and P. Peterson, *SciPy: Open source scientific tools for Python*, http://www.scipy.org/, 2001. [Online]. Available: `http://www.scipy.org`.

[51] J. W. Tukey, *Exploratory data analysis*. 1977, vol. 2.

[52] J. L. Hintze and R. D. Nelson, "Violin plots: A box plot-density trace synergism," *The American Statistician*, vol. 52, no. 2, pp. 181–184, 1998.

[53] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The annals of mathematical statistics*, vol. 18, pp. 50–60, 1 1947.

[54] Statista. (2018). Share of individuals reading or downloading online news, newspapers or magazines in Great Britain from 2007 to 2017, [Online]. Available: `https://www.statista.com/statistics/286210/online-news-newspapers-and-magazine-consumption-in-great-britain/`.

[55] ——, (2018). Newspaper websites ranked by monthly visitors in the United Kingdom (UK) from 2013 to 2016 (in million visitors), [Online]. Available: `https://www.statista.com/statistics/288763/newspaper-websites-ranked-by-monthly-visitors-united-kingdom-uk/`.

[56] Guardian News and Media Limited. (2016). The Guardian – open platform, [Online]. Available: `http://open-platform.theguardian.com/`.

[57] Associated Newspapers Ltd. (). Search tips — Daily Mail Online, [Online]. Available: `http://www.dailymail.co.uk/home/article-10612/Search-Tips.html`.

[58] Express Newspapers. (). Search For " — Page 1 — Express.co.uk, [Online]. Available: `https://www.express.co.uk/search`.

[59] Judicial Council of California. (). Disability terminology chart, [Online]. Available: http://www.courts.ca.gov/partners/documents/7-terminology.pdf.

[60] Department for Work & Pensions: Office for Disability Issues. (Aug. 14, 2014). Inclusive language: Words to use and avoid when writing about disability, [Online]. Available: https://www.gov.uk/government/publications/inclusive-communication/inclusive-language-words-to-use-and-avoid-when-writing-about-disability.

[61] T. Bray, "The JavaScript Object Notation (JSON) Data Interchange Format," RFC Editor, RFC 8259, Dec. 2017, pp. 1–16. [Online]. Available: http://www.rfc-editor.org/rfc/rfc8259.txt.

Appendix A

Other appendices, e.g., code listing

Put your appendix sections here