LONDON'S GLOBAL UNIVERSITY

Using Natural Language Processing (NLP) to develop a pipeline to analyse media representation of people with disabilities in Web-based news articles

Bagus Maulana[1]

MEng Computer Science

Supervisors: Catherine Holloway, Nicholas C. Firth

Submission date: 30th April 2018

Abstract

Project Title: Using Natural Language Processing (NLP) to develop a pipeline to analyse media representation of people with disabilities in Web-based news articles

Author's Name: Bagus Maulana

Supervisor's Name: Catherine Holloway, Nicholas C. Firth

Submission Date: 30th April 2018

Research into the representation of groups (e.g. women, youth) in the news media has been common across different research fields, from social sciences to computer science. A common approach was to manually analyse a small sample of a few hundred news articles and generalise an overall conclusion from that sample, such as proportions of positive/negative depictions. Computational Natural Language Processing (NLP) could be used to process articles much faster and vastly increase the sample size, which could uncover further information such as trends i.e. how dependent variables such as sentiment varies over independent variables such as time.

This project explored the feasibility of developing a computational pipeline that performs data collection from online news sources, filtering, text parsing, sentiment scoring, statistical analysis, and data visualisation. This pipeline was used in an experiment to collect articles from three major British online news publishers and showed trends regarding how the terms used and sentiment in news articles varies over time and between different publishers when reporting news related to disability or people with disabilities.

Results indicated that minor trends in sentiment and word usage over time and between different publishers were apparent using metrics such as moving average. While the results showed the promise of performing quantitative analysis upon large bodies of literature in the media representation domain, it is highly recommended that future approaches to this media analysis problem 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 scoring.

Contents

Chapter 1

Introduction

Natural Language Processing (NLP) encompasses a wide range of computational techniques, often used alongside each other, for machine understanding of human (natural) language. A review article [1] defined 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 NLP umbrella include extracting term frequency distributions, text processing (e.g. tokenisation, stemming), part-of-speech tagging, text classification, information extraction (e.g. entity recognition), sentence structure parsing (parse tree), and sentiment analysis (or opinion mining) [2], [3]. The computational models used in NLP range from simple models, such as term frequency (counting words), to statistical machine learning and neural network models [4].

An advantage of NLP is that machines could process vast bodies of human-created literature (books, articles, posts, e-mails, messages, etc.) much faster than humans can, processing thousands of text documents per second. This allows for high-level quantitative analyses of thousands or millions of text documents from a vast corpus to be feasible, which could uncover information previously inaccessible from manually reading only a small sample of documents and generalising from the sample. This level of quantitative analysis could uncover trends and patterns from a text corpus, to answer questions such as "How does the popularity of the term 'mentally ill' increase or decrease year-on-year in British news media?"

Applying computational NLP to perform meta-analyses over large text corpora has interesting potential applications in improving our understanding of the human world, such as analysing cultural trends quantitatively. One study assembled a vast corpus of articles from regional newspapers in the United Kingdom spanning 150 years to detect long-term patterns of cultural change, such as the increase of female representation in the news, or the popularity of trains and horses for transportation [5]. This was achieved by analysing trends for n-gram frequency (a count of words or phrases in a text document) and named entities (known people, organisations, locations, etc.) in the news articles' text.

More specifically in the domain of media representation of particular groups of people (e.g. women, youth), several researchers have attempted to use features extracted using NLP to computationally analyse text documents, mainly from social media. For example, a tool to classify racist and sexist posts in social media was developed by using NLP to extract n-grams and part-of-speech tags (labels of words corresponding to its definition and context, such as 'noun' or 'verb')

from text posts [6]. However, there is still a research gap in this area, especially in applying NLP for news articles, in the context of specific groups such as people with disabilities.

The representation of groups such as people with disabilities in media has been a popular research theme in social science. For example, a 2002 study analysed a sample of 600 print articles relating to mental illnesses in New Zealand to measure the proportions of positive and negative depictions and predominant themes (e.g. criminality, educational accomplishments) [7]. There were attempts to discover trends, such as a study conducted in 1998 [8] and replicated in 2008 [9] which assessed change in representations of disability and people with disability in Canadian news media. However, the study provided only two data points (1998 and 2008) with relatively small sample sizes of 196 news articles in 1998 and 166 news articles in 2008.

Applying computational NLP to this domain would allow the possibility of analysing a much larger sample of articles and identifying trends by creating and comparing subsets based on independent variables such as time of publication and publisher. In this project, a sample of 305,185 news articles (48,967 after filtering off-topic articles) from British online news sources was used. However, challenges remain in this application of NLP, as contemporary syntax-based NLP approaches tend to be more limited in scope and are prone to inconsistencies (false positives and negatives), where mitigating these inconsistencies is currently an open area of research.

The aim of this project was to show the feasibility of utilising computational NLP approaches to perform a meta-analysis of literature available in the public online news media. More specifically, to collect news articles relating to people with disabilities in British online media and perform analyses using NLP at scale to identify trends such as variations in term popularity and sentiment over variables such as date published and publisher.

This project's goals were to achieve the stated aim by developing a computational pipeline capable of performing analytics on online news media in full, from dataset collection to data analysis and visualisation. Given a list of topics related to disabilities, each topic consisting of key terms and query terms, where terms are either a single word i.e. a keyword or a sequence of words i.e. a key phrase; this pipeline accomplished these following tasks: web crawling and scraping websites to collect a dataset of news articles; filtering relevant articles given key terms; extracting relevant sentences that referred to a key term from these articles; scoring the sentiment of these sentences; and producing relevant visualisations and statistical analyses to show trends. This pipeline is available open source on GitHub (`https://github.com/bmaulana/nlp-media`).

The main NLP techniques that were relevant for this project are: text processing, to prepare raw text for NLP analyses using tokenisation, splitting text into a list of 'tokens' (words), and stemming, reducing words to their word stem e.g. talked $\rightarrow$ talk; term (n-gram) frequency, to quantitatively count the occurrence of words and phrases (sequences of words) in an article; sentiment analysis, to produce a 'sentiment score' of news articles that correspond to its perceived positive/negative view. These techniques were implemented by utilising open-source NLP implementations.

This project was carried out in a modular approach. The pipeline was developed as individual components: a web scraper and crawler for data collection given a list of query terms; a filter to remove irrelevant articles given a list of key terms; a parser to extract key term occurrences and relevant sentences from articles, given a list of key terms; a sentiment scorer for sentences and articles; and a script to perform statistical analysis on the results and produce relevant plots.

The main pipeline script connects these components together by calling them in sequential order, performing analyses for each topic and Web news source (Daily Mail, Daily Express, Guardian). This approach was ideal as changes could be made to a component without affecting or needing to rewrite code in other components. Each component's output is saved to a JSON [10] file and the next component reads the previous component's output file, which ensured that computation can be 'resumed' without recomputing the previous component.

The body of this report is subdivided into four chapters, followed by a conclusion and appendices. Chapter 2 covers related work on the domain of news media analysis (especially in the context of people with disabilities), a background of NLP research, and technologies relevant to 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 experiment's results in diagrams, graphs, and tables. Finally, the conclusion, Chapter 6, evaluates the project, summarises key achievements and takeaways, provides recommendations 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, including graphs and data not in Chapter 5.

The performance of several open-source sentiment analysis implementations was compared to judge their suitability for the sentiment scoring task, given the domain of sentences from news articles related to disabilities or people with disabilities. Results indicated that a relatively simpler rule-based model, VADER [11], outperformed more complex supervised machine learning or neural network models which had been trained on other domains such as tweets and IMDb/Amazon reviews and proved to be less generalisable for this domain.

This project proved the feasibility of utilising NLP-based technologies to derive trends from a large corpus of online news articles relating to disabilities or people with disabilities. The results showed that, for example, the perceived sentiment of Guardian articles are on average significantly higher than Daily Express and Daily Mail articles for the topic 'disabled'. It also showed a decreasing trend over time on the use of 'invalid' and 'handicap(ped)' and an increasing trend for 'accessibl(e)' within Guardian articles.

These results could have a substantial impact on how research should be conducted for similar studies on the media's representation of groups of people, as it showed that NLP could be used to analyse a much larger sample of text documents than traditional approaches and derive meaningful trends. However, challenges remained due to the sentiment scorer's inaccuracy. The accuracy and consistency of results could be further improved in future work by developing a domain-specific supervised model for filtering and sentiment scoring, trained on an adequately-sized labelled dataset of news articles within the domain.

Chapter 2

Context

2.1 Background

Analyses of news media had been a popular research method. News media provides an overview of the prevailing society's conceptions or views regarding a theme or topic, which could be analysed to obtain quantitative information. This approach has been commonly used to study representations of particular groups of people, such as women, youth, or people with disabilities, as the language used in the media reflected and shaped prevailing views, and had 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 disabilities and use appropriate labelling than the Israeli press in 1998 [8]. Furthermore, there is evidence to suggest that news media sources contribute to shape and reinforce beliefs among the society, such as misconceptions and stigma [12].

There had been various studies related to the public awareness of disabilities. A review in 2011 [13] found 75 articles and 68 studies that passed selective inclusion criteria with regards to intellectual disabilities, published in English between 1990 and mid-2011. Their inclusion criteria omitted studies which were found to be irrelevant, duplicate, or not written in English; and only accepted articles which were published in full in peer-reviewed journals, and the study's subject had to be the general public of working age (instead of particular subgroups). This showed interest in the research community to find new ways to understand and quantify the public's perception towards disabilities.

Analyses of news media were primarily carried out by obtaining a small sample of documents (news articles) from a text corpus, such as all news articles published in England for a certain period, and analysing them manually. There had been various such studies on news media within the domain of disability representations:

- In a 2002 study [7], researchers analysed a sample of 600 print articles published in New Zealand relating to mental health or mental illness that were collected by a commercial clipping bureau. The articles were categorised into positive or negative depictions; and further into themes such as danger to others, criminality, vulnerability, etc. The study found that negative themes predominate about 3 to 1, with 27% being positive. Given the study's scope, this conclusion cannot be expanded to learn trends of how the conclusion varies given certain variables such as time or location.

- A similar study in 2005 [14] 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).
- A study conducted in 1998 [8] and replicated in 2008 [9] assessed change in representations of disability and people with disabilities 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 was an attempt to identify trends with regards to media representation of disability, however, it only provided two data points (1998 and 2008) with relatively small sample size.

Data collection and processing of text documents using computational techniques based on NLP are much more feasible in scale, cost, and time, relative to manual collection and reading of text documents. An automated script could be used to collect thousands of news articles published on the Internet per hour, a vast improvement over e.g. contracting a commercial clipping bureau to provide 600 articles as in [7]. By applying NLP-based computational techniques in analysing text, it would be possible to develop a pipeline that could analyse and extract quantitative information from news articles' text at a much faster pace than manual reading, enabling the analyses of a much larger scale of documents.

While a sample of few hundred documents was usually enough to provide statistically significant conclusions, by providing an analysis of the full corpora or a much larger sample, additional information could be uncovered from the dataset. Higher-level trends, such as how word usage varies by year or publisher, could be discovered from a quantitative analysis of the larger dataset, by 'splitting' the result set into smaller subsets based on independent variables such as publication year and publisher, and comparing dependent variables such as term frequency between subsets. Furthermore, computational pipelines for data collection and scoring news articles are more objective and reproducible than manual methods, which may vary due to each researcher's individual biases.

There had been numerous attempts to take advantage of an NLP-based computational approach to conduct a more thorough analysis of large text corpora. A team of researchers assembled a corpus of 35.9 million news articles from 120 publishers in the United Kingdom between 1800 and 1950, which represented 14% of all news articles published in the United Kingdom over that period [5]. Using a computational NLP-based approach, the researchers extracted quantitative time-series information from the vast dataset, represented as n-grams and named entities, which allowed them to discover macroscopic cultural trends. They analysed and compared n-gram trends across various topics and identified trends that reflect cultural shifts, such as 'train' overtaking 'horse' in popularity around 1900, or 'labour party' overtaking 'conservative party' and 'liberal party' in news coverage from the 1920s. Additionally, they used entity recognition to extract named entities and identify trends based on known information about these entities, such as the proportion of female and male entities over time. They also considered the geographical location of the publication, to identify how usage trends of n-grams such as 'british' and 'english' differ based on location.

The British news media study was inspired by prior discussions and studies on the potential

of exploiting large text corpora to detect macroscopic, long-term cultural changes. A seminal study in 2011 [15] started the field of 'culturomics', to perform large-scale quantitative analyses on text corpora. In the seminal study, a corpus of 5 million digitised English language books published over 200 years, or about 4% of all books ever published, were analysed to extract how often n-grams were used over time. This data is available on `http://www.culturomics.org/`. This information was then used to analyse trends in the English language: the size of the English lexicon, regularisation of English verbs from irregular to regular ('-ed') suffixes, or how quickly mentions of years (e.g. '1950') decline in use.

Influenced by the seminal study, numerous other studies adopted similar approaches to analyse large datasets:

- an analysis of 1.7 million Victorian-era books [16]
- an analysis of 17,094 US Billboard Hot 100 songs between 1960 and 2010 [17]
- an analysis of a 3.9 million news articles sampled from the Summary of World Broadcasts (SWB) collection [18]
- an analysis of 2.5 million English-language news articles from 498 online news outlets from 99 countries [19]

This approach had also been criticised for ignoring semantics and context. For example, critics noted that "thirteen hundred words of gibberish and the Declaration of Independence are digitally equivalent" [20], issues with OCR quality and duplicate editions [20], or that the selection of digitised books was biased [21].

There had been some progress in applying similar NLP-based approaches specifically in the domain of how specific groups of people are represented in the media:

- An attempt to classify racist and sexist posts in social media, which used NLP to extract features such as n-grams and part-of-speech tags from social media posts, which were used to label 6,909 tweets [6].
- A study explored potential linguistic markers of schizophrenia in social media; using a dataset of 174 users with self-reported schizophrenia and up to 3,200 tweets per user, and a similarly-sized dataset of 'control' (non-schizophrenic) users. The researchers used NLP to extract features from tweets such as lexicon-based approaches (i.e. a list of mental health related keywords), latent Dirichlet allocation (LDA), Brown clustering, character n-grams, and perplexity, which were fed to a support vector machine (SVM) classifier [22].

Despite these studies, there is still a visible research gap on applying computational NLP approaches with regards to the representations of specific groups, such as people with disabilities (or a specific disability). This project focused on studying the representations of people with disabilities from online news articles.

To date, advances in NLP research has made the 'culturomics' approach much more feasible, efficient, and effective, even under limited time and hardware constraints. A growing number of free and open-source tools for computational NLP and statistical analysis had been developed by the research community; including general NLP tools such as nltk [2], SpaCy [23], and StanfordNLP [24], statistical analysis tools such as numpy [25], scipy [26], and scikit-learn [27], and data visualisation tools such as matplotlib [28]. Section 2.3 contains a further listing and discussion of these tools, alongside the specific libraries and NLP techniques used for this research project.

2.2 Research Methodology and Sources

Background research was 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 NLP, specific NLP techniques, disability, and news media: for example, 'natural language processing review', 'news media' AND 'disability', 'natural language processing' AND ('news media' OR 'cultural trends'), 'natural language processing' AND 'disability', and 'sentiment analysis'. Highly-cited research articles were prioritised as examples, as they were deemed to be the more important papers on its topic. Additionally, the author looked for highly-cited 'key' papers and review articles on each specific topic and consulted its list of citations (older papers) and articles that cited the review article (newer papers) to expand the list of reviewed research papers further.

Technical research, on the other hand, was carried out as necessary. After the requirements and components for the pipeline had been decided, research was carried out to find relevant techniques, formulae, algorithms, tools, repositories, packages, and existing implementations that would be useful to implement each necessary component. This research was carried out in an iterative approach alongside software development. Initial planning and research would provide an initial implementation plan, the plan's implementation would uncover the feasibility of the initial approach and possible alternatives/refinements to be researched, further research may reveal new options/refinements to be implemented, and so on. Research or work done for a component may also reveal possible improvements and/or alternatives for another component, which may require further research and implementation. Again, more popular tools and libraries were prioritised, although several approaches and implementations were considered for most components and subtasks to compare for suitability, runtime, results, etc.

The sources that were used for this literature review are:

- Google Scholar, often used to find research articles related to a research topic, to find other studies similar to a research article, or to retrieve citation information with regards to a research paper, book, or popular Python package.
- GitHub topics, used to find repositories that are relevant to help implement a specific NLP technique or component and find similar/alternative implementations and was also used as a benchmark for the popularity and range of solutions for a topic in each programming language.
- Several GitHub pages also curate a list of repositories for a specific topic [4], [29], [30].
- Python package repositories such as PyPi [31] and Anaconda Cloud [32], which lists all available Python packages and general information regarding them, and provides a search function; useful to find relevant packages for a component/task and gather information about a Python package.
- Official websites and documentation of Python packages, which list and define capabilities of the package, used to explore a package's functionality and capacity to solve a specific task, and to understand the technologies and approaches used by the package's implementation. On more popular packages, citation information regarding the package would often be available from its website.

2.3 Technical Context

Natural language processing (NLP), defined as "a theory-motivated range of computational techniques for the automatic analysis and representation of human language" [1], encompasses a range of computational techniques to extract information from text. In this project, NLP techniques were used to filter irrelevant articles, extract features such as relevant sentences from article text, and score sentiment of articles, as defined in the project's goals.

Text processing was used by all components to 'prepare' text for further analyses, where tokenisation was implemented to transform raw text into a list of tokens (words), and normalisation techniques such as stemming and stop word removal [2] were used to remove the distinction between equivalent tokens. Dataset filtering was implemented by measuring term frequency, the counts of words and phrases (sequences of words). Feature extraction was implemented by using a rule-based matcher to find occurrences of tokens that refer to key terms. Sentiment scoring was implemented using a selection of open-source sentiment analysis tools, which were compared based on their accuracy on a small, manually-labelled sample of the dataset.

Not every component defined in the project goals required NLP techniques. Data collection was performed using established general-purpose tools for sending web requests and scraping text from websites. Similarly, statistical analysis and data visualisation were performed using general-purpose statistical and data visualisation tools. This section covers the technical tools that were researched and alternatives that were considered for all components of the solution, regardless of whether NLP was utilised.

NLP covers three main 'curves' or areas: syntax, semantics, and pragmatics (narratives, understanding) [1]. Syntax specifies the way symbols 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. Contemporary approaches to NLP mainly focus on syntactic analysis, due to the relative ease of extracting syntactic features of text documents, such as term frequency and part-of-speech tags. However, syntactic analysis is much more limited, as it often misses information about the semantic context of a word. For example, the distinction between the word 'one' in "there's no one there" (referring to a person) and in "we have only one car" (referring to a quantity) is lost. This report focuses on syntactic techniques and features, as these are more relevant to the NLP techniques used for text processing, filtering, feature extraction, and sentiment analysis that are feasible with current technology at this scale.

Python 3.6.3 [33] was used as the main programming language for this project. It was chosen for the extensive availability and range of Python tools for NLP, sentiment analysis, statistical analysis, data visualisation, web scraping and parsing, etc. A study in 2016 showed that Python is the most popular language for machine learning and data science [34]. A GitHub search for the topic 'nlp' as of 18 April 2018 reported 1,397 Python and 470 Jupyter (a Python-based interactive 'notebook' technology) repositories with the tag 'nlp', compared to the second most popular programming language being Java with only 251 repositories tagged with 'nlp' [35]. Furthermore, Python is an ideal language for quick experimentation due to the relatively high level of abstraction and low verbosity of Python code, which makes it relatively easier to make changes on the fly. The Anaconda distribution of Python [36] was used due to its suitability to set up and manage Python environments and packages for data science projects.

2.3.1 Data Collection Methods

General-purpose tools for sending HTTP requests (to open web addresses/URLs and store HTML web pages programmatically) and parsing HTML code (to parse article text and metadata from raw HTML web pages) were sufficient for data collection.

The Requests library [37] is a popular Python tool with 400,000+ daily downloads for sending HTTP/1.1 requests. An HTTP GET request retrieves the HTML code, alongside other information, associated with a given URL from a web server, similar to opening the URL on a web browser. Requests stores the web server's response, including the HTML code, as a Python object.

BeautifulSoup [38] is a Python library that provides simple functions to navigate and search HTML code. As the web pages of online news articles from the same source/publisher tend to follow a similar HTML structure, BeautifulSoup could be used to parse article text and relevant metadata such as headline and date of publication, by searching for specific tags and attributes within the HTML code. Similarly, BeautifulSoup could be used to parse outgoing links from a search page.

2.3.2 Filtering Methods

In this project, filtering of off-topic articles was achieved by measuring the term frequency of key terms independently for each document. Term frequency (tf) is a simple and commonly-used metric in NLP, with various existing tools that can compute this metric for thousands of text documents per second. To put simply, the frequency of a term (a token or sequence of tokens) in a document is the number of times that the term occurs in the document. The popular scikit-learn library [27] provides a tool (CountVectoriser) to measure term frequency of all tokens (words) in a corpus of text documents, handling the tokenisation of raw text (converting a raw text document into a list of tokens) and counting token occurrences.

For text normalisation, CountVectoriser provides the option to ignore stop words, or common words in English which do not add topical information, such as 'the' and 'a', which can often bias results due to their relatively high occurrences. Additionally, the nltk library [2] was 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. Furthermore, all text in the article were converted to lower-case to ignore capitalisation, such that e.g. 'disability' and 'Disability' are equivalent.

In literature related to NLP, more complex approaches had 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) [39], [40], a metric that builds on term frequency by considering 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, it is deemed to be less important and assigned a lower score. For example, common words such as 'the' are assigned very low idf scores. The tf-idf score is measured by multiplying tf with idf. However, this approach was not suitable for this project, given the selective nature of the corpus, as only articles containing defined query terms were collected; thus, the idf scores of these query terms were flawed, as at least one query term would exist in every document.

Another often-proposed approach was to use a supervised machine learning model to classify documents into pre-defined categories (the text classification problem). Various supervised ma-

chine learning models were proposed to solve text classification, including Support Vector Matrices (SVM), Naïve Bayes (NB), and k-nearest neighbour (kNN) models [41]. However, this approach was not feasible for this project due to a lack of labelled data, i.e. a dataset of articles with "is this article relevant?" labels.

Topic models, which computes the proportion of abstract 'topics' in a document, had also been proposed. Latent Dirichlet allocation (LDA) [42] 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 (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 key terms. Additionally, LDA is significantly more computationally expensive than tf or tf-idf.

2.3.3 Feature Extraction Methods

Before the sentiment scores of news articles could be measured, the sentences that are relevant to each topic had to be extracted from the article's text, given a list of key terms. This required NLP tools to parse text and extract structural information, such as sentence structure. Other features, such as the frequency of each key term on each article, could also be extracted while parsing text.

SpaCy [23] is a popular Python library for parsing text documents. It handles the tokenisation and normalisation of raw text, and extracts information such as part-of-speech tags and named entities, using pre-trained convolutional neural network models. SpaCy was benchmarked to be the fastest and among the most accurate syntactic parser, being able to parse 13,965 words per second in 2015 [43].

Among the information that SpaCy extracts from text documents are lemmas, or word stems, of all tokens (e.g. 'mentally' $\rightarrow$ 'mental'). SpaCy also features a rule-based matching engine, which retrieves a list of tokens or sequences of tokens within a document that matches a given pattern, such as tokens with a specified lemma, or sequences of tokens with a specified sequence of lemmas. This was used to implement the task of extracting relevant sentences from an article, defined as sentences containing a word (token) or phrase (sequences of tokens) equivalent to a key term.

2.3.4 Sentiment Analysis Methods

Sentiment analysis, or opinion mining, was defined as "the field of study that analyzes people's opinions, sentiments, evaluations, attitudes, and emotions" from natural language [3]. 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' [44]. Furthermore, there exists a community-curated list of sentiment analysis implementations on GitHub [4], which served as a useful starting point to explore open-source sentiment analysis implementations.

A variety of open-source tools and pre-trained models dedicated to sentiment analysis were researched for comparison. Except for VADER and TextBlob's PatternAnalyser, these models were based on supervised machine learning or neural network approaches, which infers patterns between features (information extracted from text using NLP) and labels (sentiment scores) from a labelled training dataset, and uses these learned patterns to estimate the label of 'unseen'

text. However, patterns learned from other domains (e.g. social media posts) may not be fully relevant in the domain of this experiment (news articles related to disabilities). On the other hand, VADER and TextBlob's PatternAnalyser implement a relatively less complex rule-based model, which computes sentiment scores based on the presence of pre-defined rules or patterns. Unless otherwise stated below, these tools all return a real-valued sentiment score between -1.0 and 1.0 for each sentence.

The sentiment analysis implementations that were explored in this paper are:

- VADER [11] is a relatively simple rule-based model that scores the sentiment of a given sentence based on rules such as the presence of 'sentiment lexicons', a list of terms common to sentiment expression, such as 'good' and 'bad', including slang words, emoticons, and acronyms; negations (e.g. 'not good'); and 'emphasis', or increased sentiment intensity due to punctuation, capitalisation, and degree modifiers (e.g. 'very').
- xiaohan2012's 'twitter-sent-dnn' repository provides a trained convolutional neural network model with dynamic k-max pooling (DCNN). The model considers features based on word and n-gram order of the sentence, and an induced feature graph generated by the DCNN. It was trained on a dataset of 1.6 million tweets with inferred labels based on emoticons. It is an implementation of [45].
- kevincobain2000's 'sentiment_classifier' repository [46] provides a trained supervised machine learning model based on Naïve Bayes and Maximum Entropy Classifier, which transforms a sentence into positive and negative sentiment scores. It uses bigrams (2-grams, sequences of 2 adjacent words) as features, implements Word Sense Disambiguation using wordnet [47] to transform bigrams into 'senses', and considers word occurrence statistics from nltk's movie review corpus. 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 with 4096 units, a relatively simple model optimised for training/convergence time. It was trained on a dataset of over 82 million Amazon product reviews from May 1996 to July 2014, a substantially larger training set than any previous work, 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, used as features. It is an implementation of [48].
- Stanford CoreNLP [24] provides a set of linguistic analysis tools, including sentiment analysis, that runs on a local server. Its sentiment analysis tool uses a recursive neural network model that converts sentences to parse trees to be used as features. It was trained on a dataset of fully-labelled parse trees for 215,154 unique phrases and 11,855 sentences from the Rotten Tomatoes movie review corpus. Unlike other scorers on this list, it classifies sentences into five sentiment classes, from 'very negative' to 'very positive', instead of assigning a real-valued score [49]. 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 [50].
- TextBlob [51] is a general-purpose NLP 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 (with no information on features used or dataset size).

Several other repositories were also explored, however, they were deemed unsuitable for this project either due to requiring labelled training data, which were unavailable for the domain of the experiment (news articles relating to disabilities), instead of providing a pre-trained model, or the implementations were broken or infeasible.

2.3.5 Statistical Analysis and Data Visualisation Methods

For statistical analysis and data visualisation, the conventionally used libraries in Python are numpy [25], scipy [26], scikit-learn [27], and matplotlib [28].

Numpy [25] provides a powerful and efficient n-dimensional array object, often required for other libraries. It provides efficient functions that perform mathematical computations over real values, vectors (1-dimensional arrays), and matrices (2-dimensional arrays) such as scalar/vector/matrix addition and multiplication, extracting vectors from matrices, and boolean filtering. Numpy is highly efficient as its functions are a Python wrapper for functions written in C, which allows for more direct hardware-level manipulation than pure Python [52].

Scipy [26] is a library that extends numpy to provide additional domain-specific functions. It extends numpy's 2-dimensional array to implement a sparse matrix which only stores non-zero values in memory, useful in the case of a very large matrix where most elements are zero; such as a term frequency matrix of 'all tokens' $\times$ 'all documents' where most documents only contain values for a very small subset of tokens. It also provides implementations of statistical equations, such as the Mann-Whitney U Test used to compare subsets of data, defined below.

Scikit-learn [27] provides efficient implementations of algorithms for data analysis, feature extraction, and machine learning, built on top of numpy and scipy. In this project, scikit-learn was used for data processing, such as the CountVectoriser used to compute term frequencies.

Matplotlib [28] is a 2D plotting library that produces visual graphs from lists/arrays. It provides the capability to generate various types of plots, such as scatterplots, line plots, histograms, and box-and-whisker plots. Furthermore, matplotlib's plots are very flexible, as parameters such as colours, labels, and bounds could be modified, and a legend or colour bar could be generated. It also allows to concatenate plots in a grid of subplots and plot multiple graphs on the same axes.

The types of plots and statistical metrics that were deemed relevant in this project are:

- Scatter plot, used to show the distribution of data within two variables (e.g. year published and sentiment score). Colour could be added to show a third variable (e.g. the publisher). Available on matplotlib.
- Histogram (and two-dimensional histogram), used to show the distribution of articles relative to variables such as the publisher, year of publication, and sentiment score ranges. Available on matplotlib.
- Box-and-whiskers [53] and violin plots [54], 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. the 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 numpy.

- The Mann-Whitney U Test [55] is a non-parametric statistical test that measures the null hypothesis 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 or greater than the second value". If the null hypothesis holds, then there is no statistically significant difference between the two distributions. The Mann-Whitney U Test returns the U statistic and a p-value between 0 and 1, which corresponds to how likely the null hypothesis is to be true. Available on scipy.

Chapter 3

Requirements and Analysis

3.1 Problem Statement

The primary aim of this project was to utilise available NLP-based technologies to perform a high-level meta-analysis of online news articles from British media relating to people with disabilities. The main objective was to reveal trends with regards to sentiment and word usage, by varying the dataset for independent variables such as source (the publisher) and date published.

The solution had to collect and scrape news articles from the Internet, filter irrelevant articles, extract information from these articles, perform statistical analyses, and display visualisations of the data to show trends. Of particular interest as a dependent variable was a sentiment (or opinion/polarity) index of articles, to measure how positively does an article view disabilities or people with disabilities, and how the sentiment index varies over independent variables such as source and date published. Thus, the general solution defined by this aim involved the completion of several sub-problems, primarily data collection, filtering, parsing, sentiment scoring, statistical analysis, and data visualisation.

To achieve this aim, the project's goals were to develop a computational pipeline that implements all components required for the general solution. Before the project could be started, the topics relevant to this project had to be defined. Thus, a list of topics relating to the domain of people with disabilities was compiled, where each topic consisted of a list of key terms and query terms related to a specific disability, or disabilities in general. Refer to section 4.2.2 for a list of topics, key terms, and query terms.

Given the list of topics, key terms, and query terms, the pipeline had to implement these following tasks (goals):

- Web crawling and scraping web pages 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 key terms for each topic.
- Extracting relevant sentences that refer to a key term, from the dataset of filtered articles (feature extraction).
- Performing sentiment analysis (using open-source libraries and pre-trained models) on the dataset of relevant sentences.
- Producing statistical analyses and data visualisations to show trends over independent vari-

ables such as source and year published.

As the primary advantage of computational NLP is in its speed, and thus the vastly increased sample size that could be analysed relative to human reading, the solution must be able to collect and analyse the dataset quickly and at scale. The total number of documents in the dataset, given the sources (section 4.2.1) and query terms (section 4.2.2) used in this experiment, was 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 for the full dataset within a feasible timeframe of not more than a handful of days, given available consumer-grade 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 solution followed a component-based design, where each task or goal must be implemented by a component that focuses only the specific task. These components must be linked together via the main pipeline script, that must execute each component in sequential order, iterating through the full list of topics and sources.

This design was ideal as changes could be made to a component without affecting or needing to rewrite code in other components. Additionally, each component's output was saved to a JSON [10] file and the next component reads the previous component's output file, which ensured that computation could be 'resumed' without recomputing the previous component.

The list of required components was as follows: data collection, dataset filtering, feature and relevant sentence extraction, sentiment scoring, and statistical analysis and visualisation.

3.2.1 Data Collection

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

After the list of URLs pointing to online news articles had been compiled, the component must be able to scrape the web pages pointed by these URLs and extract the full article text and relevant metadata from each URL. Aside from the full article text, the relevant metadata that must be extracted was the article's headline, URL, date of publication, and publication source. The information must be returned in a JSON object format, with each JSON object containing the text and metadata of a single news article.

This information must then be saved locally to a file where it could be read by the next component. The file and directory structure of the output file must be consistent given source and topic, such that the next component could find it programmatically. This requirement holds for all other components in this pipeline.

As data collection was expected to consume the longest time compared to other components,

due to the necessity to submit a web request for each article, additional requirements regarding scalability were 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 immediately store collected articles to an external file, and read from the external file upon starting to gather a list of saved URLs and avoid re-requesting saved articles.

3.2.2 Dataset Filtering

This component handled dataset filtering. Initially, this component must be able to load the dataset of collected articles from the output file saved by the data collection component. Each file contains a subset of all collected articles for a specified topic and a specified source.

For each article, it must decide whether the article is relevant to the topic defined by the subset, given a list of key terms related to each topic. This decision should be based on the article's full text and headline and should take all key terms associated with the current topic into account. It should also be able to display a sample of an arbitrary number of articles, and the component's decision (on-topic/off-topic) for each sampled article, which would be used to analyse and improve the filter's accuracy.

This component must save all articles deemed on-topic (relevant to the topic) to a new output file, containing a collection of JSON objects retaining 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.

3.2.3 Feature Extraction and Rule-based Sentence Matching

This component parsed the article text using open-source NLP tools to extract relevant information required by the sentiment scoring and statistical analysis components. Given the output file of the dataset filtering component, this component must be able to load the article text and metadata of all on-topic articles.

The sentiment scoring component required all relevant sentences relating to a key term (containing a keyword, key phrase, or an equivalent term) to be extracted from each article. This component must extract all relevant sentences, as defined above, from all articles.

This 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 each article, and the term frequency of each key term in each article.

After these data had been extracted from each article, it must save the information to a new output file as a collection of JSON objects, where each JSON object contains all the extracted information about a single news article, with a key for each extracted feature e.g. relevant sentences. The metadata of each article (headline, date of publication, and source) must be retained in the new output file, as they were required as independent variables for the statistical analysis component.

3.2.4 Sentiment Scoring

The sentiment scoring component computed a real-valued sentiment score for each relevant sentence extracted by the feature extraction component. Given the output file of the feature extraction component, this component must be able to load the dataset of relevant sentences and other data for all saved articles.

The sentiment scores computed by this component must correspond to the perceived 'sentiment' of the sentence towards a disability, disabilities, a person with a disability or disabilities, or people with disabilit(y/ies), referred by the key term(s) in the sentence, with sufficient accuracy.

Two iterations of the sentiment scorer component must be developed. The first iteration of the sentiment scorer component was used to perform a comparison between several open-source sentiment analysis implementations for this sentiment scoring task. It was not used in the final pipeline. The second iteration of the sentiment scorer component only computed one sentiment score for every sentence, using the best-performing sentiment scorer shown by the first iteration. The second iteration was the one used in the final pipeline.

3.2.4.1 Comparison of Open Source Implementations

The first iteration of the component must select an arbitrarily-sized sample of sentences from the dataset of relevant sentences. Also, it must implement all open-source sentiment analysis implementations that were listed in section 2.3.4.

All sentences in the sample must be given real-valued sentiment scores between -1.0 and 1.0, with a score from each sentiment analysis implementation. 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 show the accuracy of each sentiment scorer, i.e. the proportion of true positives and true negatives, and show a confusion matrix. It must also show the total, per-sentence, and per-article runtime of each sentiment scorer.

This iteration of the component was not used in the final pipeline. Instead, it was only used as a tool to compare the performance of open-source sentiment scorer implementations on the domain of news articles relating to disabilities or people with disabilities.

3.2.4.2 Final Implementation

The second iteration of the sentiment scorer component was the one used in the final pipeline. It must only compute one sentiment score, using the best-performing sentiment scorer that runs in reasonable time, as shown by the results of the first iteration (section 3.2.4.1). It must compute the sentiment scores for all relevant sentences in the dataset.

After the sentiment scores of all relevant sentences within an article had been computed, this component must also calculate the average sentiment score for each article based on the sentiment scores of relevant sentences in the article. Furthermore, the component must run in reasonable time (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 component had scored all relevant sentences and articles in the dataset, it must save information about all articles, with score labels appended to each sentence and article, to a new output file. The JSON objects of each article in the output file must retain

the same format and information as the feature extraction component's output file, except for an additional 'sentiment score' key-value pair that must be added in each sentence's and each article's JSON object.

3.2.5 Statistical Analysis and Visualisation

The last component performed statistical analysis and data visualisation. This component must read the sentiment scoring component's output files to load the dataset of news articles and features extracted by previous components about each article. At this point, information extracted from each article by previous components must include the source, publication date, and sentiment score. To combine information about articles relating to the same topic from different sources, represented by different files, this component must be able to read multiple files in a single call and collate the information into a single dataset.

This component must be able to show trends and visualisations that show how dependent variables (e.g. sentiment score) differ relative to the independent variables (e.g. publication year and source). It should also compute statistical metrics to show whether the differences between subsets, as divided by the independent variables, are statistically significant. The plot types and statistical metrics relevant to this analysis were defined at the end of section 2.3.5.

At a minimum, this component must visualise how the sentiment score varies relative to publication year and source. It should also show metrics related to the distribution of sentiment scores on subsets divided by publication year and source. Additionally, the component should also show visualisations based on information extracted by previous components other than sentiment scores, such as how the term frequency of each key term varies over time.

3.3 Analysis of Requirements

The implementation design and additional requirements of this project were highly influenced by the core requirements, i.e. data collection, filtering, sentence extraction, sentiment scoring, statistical analysis, and data visualisation. The overall design of the pipeline, with isolated components for each sub-problem, stemmed from the largely independent tasks that were required to perform the data analysis experiment in whole, from data collection to analysis and visualisation.

Initial prototypes, or initial 'runs' collecting only a limited number of news articles, showed that the vast majority of the runtime was spent on data collection. For this reason, the requirement where it should be possible to resume progress on data collection, instead of repeating the process for already collected data, was added.

The JSON object formats for each component's output file were largely defined by the requirements for each component, i.e. the information that must be extracted from each component. For example, the JSON object format of an article in the data collection component's output file is as follows:

```
{
        "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 requirements defined for the data collection component.

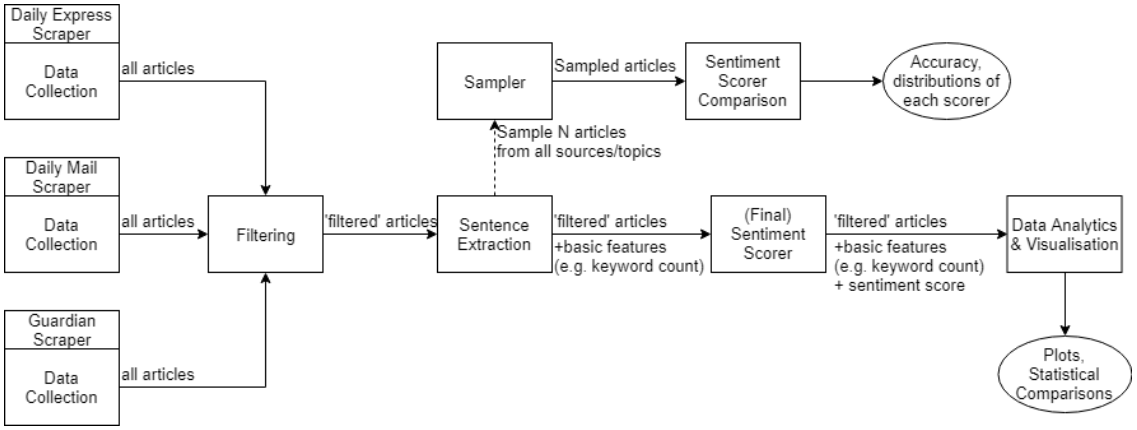
Chapter 4

Design and Implementation

4.1 Overall Design

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

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



In this experiment, the pipeline was run once for each source and topic. Each run up to the 'sentiment scorer' component generated a dataset of all articles for each specified topic from each specified source in a separate file, with all extracted features such as sentiment scores and publication date. The 'plot' component then loaded all files for every source with the same topic, and combined the datasets to produce visualisations that show trends within the topic, and computed statistical metrics for subsets of the combined dataset.

Components passed datasets to each other by saving all information it extracts in a JSON file. 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 these files was a collection of JSON objects, where each line consisted of a single JSON object, and each JSON object represented a single article. The JSON object contained a key-value pair for each extracted feature. Each component retains all information extracted by previous components and appends additional key-value pairs for each

feature it extracts. This design ensures that in the case where computation was interrupted for any reason, the dataset generated by the previous component had already been saved to a file, and computation can restart from the interrupted component, without having to recompute previous components.

This modular design ensured that changes could be made to a component without affecting or needing to rewrite or recompute code in other components, which offered flexibility for experimentation.

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 news articles. As of 2017, 64% of British individuals read and/or download online news, newspapers or magazines; a sharp increase from 20% in 2007 [56].

The Daily Mail (`dailymail.co.uk`) and the Guardian (`theguardian.com`) were the two most visited British news publisher's websites as of 2016, with a monthly online viewership of 11.85 and 10.05 million respectively [57], and are the main subjects of this experiment. The Daily Express (`express.co.uk`), a slightly smaller news publisher with a monthly online viewership of 2.67 million in 2016 [57], was also added as a news source in this experiment, to explore how the experiment performs with smaller sample sizes.

The Guardian provides an API [58] and the Daily Mail and Daily Express provide advanced search tools [59], [60] to query news articles from their websites, which proved 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, the dataset for this experiment contained news articles from these three news publishers or sources; the Daily Mail, the Guardian, and the Daily Express.

4.2.2 Topics, Key Terms, and Query Terms

A list of topics relevant to disabilities and a list of key terms for each topic were compiled for this experiment. The list of key terms was roughly based on guidelines from the Californian [61] and UK [62] governments, ignoring whether the term is considered 'appropriate' or 'inappropriate', as terms considered 'inappropriate' are often still commonly used in the news and still relevant to consider when deciding whether an article is on-topic. Additional key terms were added based on other commonly used terms for the same disability, found on sampled articles from the Daily Express, the Daily Mail, and the Guardian.

The list of query terms was compiled based on the list of key terms, with terms that could have other meanings or phrases which were a combination of multiple common words removed, unless the term is very commonly used to refer to the disability, such as 'blind' and 'mute'. Query terms were used in data collection, where ambiguous key terms were removed to reduce off-topic articles, while key terms were used in subsequent components. Terms, in both definitions, can either be a single word (a keyword), or a sequence of words (a key phrase).

The final list of topics relevant to disabilities and people with disabilities and the key terms and query terms deemed relevant to each topic was:

Topic	Key Terms	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', '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'

4.2.3 Dataset Size

The dataset was composed of all articles found online, given the query terms defined in the section above, published between 2000 to (approximately) end of March 2018. The size of the initially collected dataset, i.e. all articles collected by the data collection component prior to any further processing, measured by the number of articles for each source and topic was:

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 plotted dataset, measured by the number of articles for each source and topic was:

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 was on the 'disabled' topic, as it had the highest amount of news articles within its subset, and was also the most generalisable topic on disabilities as it refers to the general theme of disabilities and people with disabilities instead of a specific disability.

4.2.4 Limitations

As shown in section 4.2.3, the Daily Express had much fewer articles in the dataset for any given topic than the Daily Mail or the Guardian. Some topics, such as 'cerebral palsy', 'developmental delay', 'dyslexia', 'mute', and 'speech impairment', were also severely lacking in sample size for any source in the filtered dataset. Cases where there were less than ~200 articles from a source within a topic are problematic to plot or form statistically-significant conclusions, as the distribution of the data was too varied. Cross-referencing the results in section 5.4 with the size of each subset showed that it was difficult to obtain statistically significant conclusions when comparing subsets with less than ~200 articles. This meant that it was difficult to analyse or compare the subset of Daily Express articles with topics other than 'disabled', 'blind', and 'mental illness'.

Another limitation of this experiment was the length of time that each news source retained articles for in their online archive. Our dataset indicated that by the end of March 2018, when the data was collected for this experiment, the Daily Express only retained articles from after ~2007, the Daily Mail only retained articles from after ~2010, and the Guardian only retained articles from after ~2000. This created an issue when analysing year-on-year trends, as the proportion of articles' sources within a topic were 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, trends over time were only considered to be significant when the trend is consistently repeated for every source's subset for a topic, 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 was composed of two sub-parts. The first part of this component was a scraper object unique to each supported source (see section 4.2.1) that defined two functions:

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

To implement these two functions, the Requests library [37] was used to send HTTP GET requests to retrieve the web page associated with a search page or a news article's URL, and the BeautifulSoup library [38] was used to parse an HTML web page, and retrieve outgoing links from a search page or article text and relevant metadata from a news article's web page.

These two functions required separate implementations for each source, as each web page has a different HTML structure, although the HTML structure of news articles from the same source tend to be consistent. The scraper object creates a wrapper for the implementations of these two functions to be injected into the main data collection script.

This design allowed the use of a single data collection script (the second part of this component) for multiple sources, by having the two functions that required separate implementations for each source injected as a dependency onto the script. For each query term in the category, the script sent HTTP requests to the corresponding search page(s) to gather a list of URLs pertaining to online news articles. It then combined all unique URLs for all query terms in the category into one list. For each URL, an HTTP request was sent to retrieve the article text and metadata.

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 re-sending HTTP requests for articles already in the collection, such that it was possible to update the collected data or recover collected information in case computation is interrupted. To achieve this, the data collection script first loads the existing output file for the source and category if it exists, and loads the list of already-saved URLs from the file. If a URL already existed in the output file, it would be skipped by the data collection script, i.e. no HTTP request would be sent. Each article's text and metadata was stored as a JSON object in an appended line on the output file before sending an HTTP request for the next URL. This design allowed data collection to be resumed in case computation was 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 [10].

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 experiment indicated that data collection took approximately 0.76 seconds per article for the Daily Express, 0.69 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 calculating 'term frequency ranking', which measured how often do key terms appear in each article relative to other terms. Initially, the headline was prepended to the article text, such that both the headline and the article text were considered.

Before term frequency was measured, the article text had to be normalised, ensuring that equivalent terms were treated as such. All text in the article were converted to lower-case to ignore capitalisation, such that e.g. 'disability and 'Disability' are equivalent. Then, the nltk library [2] was used for stemming, such that e.g. 'walk', 'walks', 'walked', and 'walking' are equivalent, regardless of its word form, tense, or other modifiers. Then, stop words, or common words in English which do not add topical information (e.g. 'the', 'a'), were ignored using a parameter in CountVectoriser.

As a topic may contain multiple key terms, there was an additional pre-processing step to ensure all key terms were treated as equivalent: all mentions of any key term in the article were replaced by 'KEYWORD_TOKEN'.

Term frequency ranking was implemented using scikit-learn's [27] CountVectoriser, which tokenises a text document i.e. transforms the document into an array of tokens (words), then transforms it further into an array of term frequency values for each unique token. Then, the array is sorted in descending order, and the position of 'KEYWORD_TOKEN' in the sorted array (plus one) is recorded as the keyword rank of the document. A keyword rank of 5, for example, shows that 'KEYWORD_TOKEN' is the 5^{th} most occurring token in the document.

To use this keyword rank for filtering documents, a constant threshold value of 20 was used to determine whether documents are on-topic or not. A document was considered on-topic if its keyword rank is lower than the threshold value i.e. the key terms are frequently used in the document relative to other terms, or considered off-topic if its keyword rank is higher than the threshold value (i.e. the key terms are rarely used in the document relative to other terms. As the keyword rank was measured relative to other terms in the document, this metric worked for documents of any length.

To evaluate the output of the filter, this component could be set to print an arbitrarily-sized sample of articles, alongside each sampled article's predicted label (on-topic/off-topic) and keyword rank. A test run of 10 articles per topic indicated that for most topics, this simple metric worked well in distinguishing between on-topic and off-topic articles. Manual reading of sampled articles indicated that:

- There existed a perceptible correlation between keyword rank and on-topic/off-topic articles, although the sample size was too low to make statistically significant conclusions, as increasing the sample size would require manually reading more articles, which was highly time-consuming.
- In most cases, articles predicted to be 'on-topic' are on-topic, and articles predicted to be 'off-topic' are off-topic.

There were, however, topics where this filtering mechanism did not perform well. As a result, many off-topic articles remained in the filtered dataset. These topics were '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 taken from articles that were mislabelled as 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 similar syntactic approaches such as tf-idf and word vectors.

Several alternatives were considered to term-frequency ranking. The conventional approach is by calculating term frequency — inverse document frequency (tf-idf) [39], [40], 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 was unsuitable for this project, due to the selective nature of the dataset where only articles containing certain query terms were collected. This caused the idf weights of these query terms to be much lower than it should be, as at least one of the query terms would appear on every document in the corpus, highly skewing the document frequency of all query terms within the dataset relative to within all Daily Mail, Guardian, or Daily Express articles. For this reason, limited evaluation showed that tf-idf ranking performed worse than just term-frequency ranking in this filtering task.

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 has been many supervised machine learning approaches that are popular for text classification, including Support Vector Matrices (SVM), Naïve Bayes (NB), and k-nearest neighbour (kNN) models [41]. However, this approach requires an adequately sized labelled dataset in the same domain, i.e. a collection of articles with 'relevant' and 'irrelevant' labels for each topic, which was not feasible.

4.3.3 Feature Extraction and Rule-based Sentence Matching

This component parsed the article text using SpaCy [23] to extract relevant information required by the sentiment scoring and statistical analysis components. Initially, the headline was prepended to the article text to create the text document.

SpaCy tokenises and normalises raw text documents and performs a syntactic analysis of the text. It returns a list of 'enhanced' tokens which contain additional information extracted by SpaCy, such as part-of-speech tags, known entities, lemma (word stem), and the sentence containing the token.

This component iterated over the list of 'enhanced' tokens returned by SpaCy to extract information that would be useful for statistical analysis:

1. Sentences containing a word or phrase relating to a key term, and the number of key terms in the sentence.
2. The frequency of the token or sequence of tokens, for each token that has the same lemma as a keyword or sequence of tokens that has the same sequence of lemmas as a key phrase.
3. Number of relevant sentences in the document (number of occurrences of point 1).
4. The total number of sentences in the document.

5. Number of key term occurrences in the document (number of occurrences of point 2).
6. The 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 required all sentences relating to a key term to be extracted from each article (Point 1). SpaCy provides a rule-based Matcher tool that returns tokens, or sequences of tokens, that fulfils a specific definition or 'rule'. This component used the Matcher tool to find all tokens that have the same lemma as a keyword or sequences of tokens that have the same sequence of lemmas as a key phrase. For each matched token, the component retrieves the text of the full sentence that contains the token. For each article, it builds a list of relevant sentences, along with the number of key terms in each sentence.

Results from running the experiment indicated that this component took approximately 0.37 seconds per article, using a single general-purpose computer (Intel i7-6700HQ CPU @ 2.60GHz, NVIDIA GeForce GTX 1060 GPU).

4.3.4 Sentiment Scoring

The sentiment scoring component implemented open-source libraries to measure the sentiment of a sentence, which ideally corresponds to the perceived view of the sentence towards a disability, disabilities, a person with disabilit(y/ies), or people with disabilit(y/ies), referred by the key terms within the sentence. Sentiment scores are real-valued scores, where a score of 0.0 indicates that a sentence is neutral, while 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 was used to compare between several open-source sentiment analysis implementations for this sentiment scoring task. This iteration computed sentiment scores of all implemented scorers for every article. It was run only on a sample of 180 sentences, as running all sentiment scorers without optimisation took roughly ~1 minute per article, which was infeasible for the full dataset.
- The second iteration of the sentiment scorer component was used in the final pipeline. This iteration only computed one sentiment score per each article and was highly optimised. Two iterations of this component were developed: one implementing OpenAI's [48] model, and another implementing VADER [11]. In both cases, the component's runtime was lower than 0.2 seconds per article.

4.3.4.1 Comparison of Open Source Implementations

This component consisted of two scripts. The first script is the sampler, which loads a small sample of articles for each topic and source in the parsed dataset. Then, it loads all relevant sentences from the sample of articles from each source and topic. From these sentences, it selects an arbitrarily-sized sample of relevant sentences for each source and topic. In the experiment, a sample of 5 sentences per source and topic * 12 topics * 3 sources = 180 sentences were used.

Every sentence in the 180-sentence sample was then scored using the 7 open-source sentiment analysis implementations that were explored in section 2.3.4:

- VADER [11],
- xiaohan2012's 'twitter-sent-dnn' repository [45],
- kevincobain2000's 'sentiment_classifier' repository [46],
- OpenAI's 'generating-reviews-discovering-sentiment' repository [48],
- Stanford CoreNLP [24], using the stanfordcorenlp package [50] to start and query a Stanford CoreNLP local server in Python,
- TextBlob's PatternAnalyzer [51],
- TextBlob's NaiveBayesAnalyzer [51].

The scores of every sentence in the 180-sentence sample were 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 all 180 sampled sentences with all 7 sentiment scores were saved to a text file, where a user can manually change the 'label' fields to either '+' (positive), '-' (negative), 'n' (neutral), or 'o' (irrelevant/off-topic) to be read by the second script. To reduce bias in manual labelling, a second output file which contained only sentences and labels without sentiment scores were used.

The second script is the analyser. Given a dataset of sentences, labels, and sentiment scores, with the format shown above, the analyser plotted sentiment score distributions of positive, negative, and neutral sentences in 7 histograms, one for each sentiment scorer. Additionally, it also calculated 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 negative-labelled sentences with sentiment score > 0.0
- True negative: count of negative-labelled sentences with sentiment score ≤ 0.0
- False negative: count of positive-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 were 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 [11] and OpenAI's model [48] as the two best sentiment scoring tools for this domain of measuring the 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 computed only one sentiment score per each article and was highly optimised. Two versions of this iteration were developed: one implementing OpenAI's model, and another implementing VADER. In the final pipeline used for the experiment, only the version using VADER was used, as VADER's sentiment scores were shown to be better at displaying trends and separating different subsets than the OpenAI model's sentiment scores (refer to section 5.2).

Within the pipeline, this component's task was to score the sentiment of every relevant sentence. Once the sentiment scores of all relevant sentences in an article had been computed, the sentiment score of the article itself was calculated, defined by the weighted average of sentiment scores for all relevant sentences contained within the article:

$$Article's\ sentiment\ score = \frac{\Sigma(sentiment\ score \times number\ of\ key\ terms) for\ each\ sentence}{Total\ number\ of\ key\ term\ occurrences\ in\ the\ article}$$

This iteration was highly optimised for runtime, as it only computes one sentiment score with one sentiment model for each sentence, instead of loading and analysing every sentence with all 7 sentiment models as in section 4.3.4.1. The version that implements OpenAI's model was 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, and calls 'openai_model.transform()' only once on the full corpus. Results from running the experiment indicated that sentiment scoring took approximately 0.16 seconds per article using the OpenAI model, and 0.0031 seconds using VADER, using a single general-purpose computer (Intel i7-6700HQ CPU @ 2.60GHz, NVIDIA GeForce GTX 1060 GPU).

4.3.5 Statistical Analysis and Visualisation

This component was responsible for statistical analysis and data visualisation, plotting graphs and computing statistical metrics for each topic. At this point, each article's JSON representation included information about its source, publication date, sentiment score, and key term frequencies.

Up to this point, the dataset of news articles belonging to each source and each topic were kept in separate files. Instead of loading just one file per call as in previous components, this component had to load all files that were related to a specified topic from every source. Then, it combined all the information from each file to a single dataset. This was necessary to compare news articles from different sources that discussed the same topic.

This component was mainly concerned with plotting the variation of sentiment scores when the publication source and date of publication is varied. For easier data processing, it loaded the publication date, sentiment score, and source information of all articles in a numpy matrix, and sorted the matrix by publication time.

The component then used the matrix to produce the following plots using matplotlib [28]:

- A scatter plot of publication date (X) vs sentiment (Y), coloured based on source (Z).

- A regular histogram that showed the number of news articles in the dataset for each year.
- Two-dimensional histograms that showed:
  - The number of news articles for each year (X) and source (Y) in the dataset.
  - The number of news articles for each year (X) and 'sentiment range' (Y) in the dataset.
  - The number of news articles for each source (X) and 'sentiment range' (Y) in the dataset.

  ('sentiment range': news articles were grouped together based on sentiment score in intervals of 0.1; for example, -0.1–0.0, 0.0–0.1, and 0.5–0.6 were sentiment ranges)
- A line graph of the moving average of sentiment score (Y) over time (X), with separate lines for each source. The moving average was defined as the mean of sentiment scores for W previous articles up to the current article, where W is the moving average window size. In this experiment, $W = no.\ of\ articles\ in\ the\ topic\ /\ 10$, with a lower cap of 50 and an upper cap of 500.
- A violin plot [54] and box-and-whiskers plot [53] that showed the sentiment score distributions (means, upper and lower quartiles, ranges, and density plots) for each data subset, separated by source.
- A violin plot that showed the sentiment score distributions for each data subset separated by source and publication year, with intervals of 2 years, separated to two plots: one for 2000–2009 and one for 2010–2018.

These plots were arranged in a 3x3 grid and saved to an output file unique to each topic. Refer to section 2.3.5 for a further description of each plot's usage.

To quantify the distribution of each data subset, the mean, standard deviation, and total count of articles for the full dataset and each possible subset of the data, i.e. a set of all articles for each source, a set of all articles for each year, and a set of all articles for each year and each source, were measured. These metrics were saved to an output text file unique to each topic.

The component also measured whether the sentiment scores of articles published by a source was significantly higher than the sentiment scores of articles published by other sources. The Mann-Whitney U Test [55] is a non-parametric statistical test where if the null hypothesis holds true, then there is no statistically significant difference between two distributions (refer to section 2.3.5 for a more formal definition). It returns the U statistic and a p-value between 0 and 1, which corresponds to how likely the null hypothesis is to be true; the null hypothesis is rejected if the p-value is lower than 0.05. This test was performed to compare the subset of all articles published by each source to the subset of all articles published by each other source; for all articles within a topic, and for all articles published in each year within a topic. These test results were saved to the same output text file.

As well as information relating to sentiment scores, trends regarding the usage of the key terms used to describe disabilities or people with disabilities over time were also plotted. This information was collected from the frequencies of tokens with the same lemma as a key term, as mentioned in point 2 of section 4.3.3. The tokens were first stemmed with a custom 'weak' stemmer: unlike NLTK's stemmer or SpaCy's lemmatiser, this stemmer only removes plurals (-s, -es) and tenses (-ed, -ing), but does not fully reduce words to its base word form, e.g. 'illness' and 'illnesses' are equivalent, but 'mental' and 'mentally' are still distinct.

Then, the dataset of key terms in articles was split into smaller subsets based on publication

year and source. For each subset, the term frequencies for all articles within the subset were averaged, to compute a measure of average term occurrence, or the expected number of occurrences of a term within any given article in a subset:

$$Average\ term\ occurrence = \frac{\Sigma(term\ frequency\ in\ article)\ for\ each\ article}{number\ of\ articles\ in\ subset}$$

For each term, the annual average term occurrences were plotted in a line graph of average term occurrences (Y) over publication year (X). Each unique term has its own plot, with separate lines for each source. These plots were then saved to an output file unique to each topic, separate from the sentiment score plots.

Chapter 5

Results Evaluation

5.1 Focused Topic

Although the pipeline was run on a list of 12 disability-related topics (as defined in section 4.2.2), this chapter mainly focuses on the 'disabled' topic. The 'disabled' topic consist of articles that relates to the key terms: 'disabled', 'disability', 'handicapped', 'cripple', 'invalid', 'accessible', 'ablism', 'ableism'. This topic was chosen as it had the largest sample of news articles within the dataset (see also: section 4.2.3) and is the most generalisable topic as it refers to the general theme of disabilities and people with disabilities rather than a specific topic.

This focus on one topic for the whole of Chapter 5 ensures the consistency of numbers and results throughout Chapter 5. That said, results from other topics are also discussed at a few points in this chapter. Information from other topics will be explicitly mentioned (e.g. "For topics other than 'disabled', ") such that the reader understands where the data does not refer to the 'disabled' topic. Furthermore, the full result plots for all topics are available in the Appendix.

For the 'disabled' topic, the sample size that was obtained for each year between 2000 and 2018, after filtering, 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 data also includes a small amount of articles published before the year 2000.

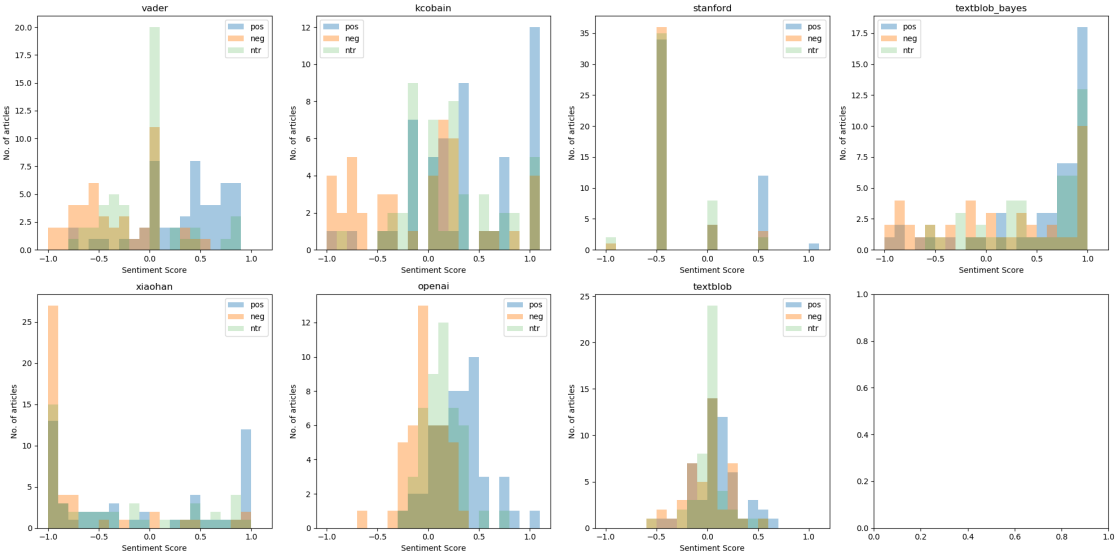
** 2018 data is incomplete and would only include articles up to (approximately) end of March.

5.2 Comparison of Sentiment Scorers

For this comparison, an evenly-distributed sample of relevant sentences, i.e. sentences referring to disabilities or people with disabilities, were taken from the filtered dataset of every topic. The sample contained 5 sentences from each of the 3 sources (Daily Express, Daily Mail, Guardian) and each of the 12 topics (as defined in section 4.2.2), or 180 sentences in total.

Each sentence in the 180-sentence sample was manually labelled to 'positive', 'negative', 'neutral', or 'irrelevant'. These labels were assigned based on the perceived sentiment of the sentence towards a disability, disabilities, a person with disabilit(y/ies), or people with disabilit(y/ies); or 'irrelevant' if it did not refer to any disability-related topic. Then, each sentence was scored using the 7 open-source sentiment scorer implementations that were explored in section 2.3.4.

The sentiment score distributions of positive (blue), negative (red), and neutral (green) sentences for each sentiment scorer was plotted as follows:



From these plots, it was clear that 'vader' [11] and 'openai' [48] were the two most encouraging sentiment scorers on this domain. Both VADER and OpenAI's model showed a clear distinction between the distributions of 'positive' and 'negative' labels, although with some overlap near the centre, while other scorers produce plots where the values were all over the place.

The means of these distributions, the count of 'positive'-labelled sentences with a positive (>0, true positive) and negative (≤0, false negative) sentiment scores, the count of 'negative'-labelled sentences with a positive (>0, false positive) and negative (≤0, true negative) sentiment scores, and accuracy (defined in section 4.3.4.1 as (true positive + true negative) / (all positive + all negative)) were also measured for each sentiment scorer. Refer to section 4.3.4.1 for a formal definition of these metrics.

The values of these metrics for each sentiment scorer are shown below:

Implementation	Mean Positive	Mean Neutral	Mean Negative	Accuracy*
VADER [11]	0.327	-0.048	-0.313	0.789
XiaoHan [45]	-0.071	-0.306	-0.681	0.621
Kevin Cobain's [46]	0.348	0.210	-0.116	0.621
OpenAI's [48]	0.301	0.128	-0.025	0.758
Stanford CoreNLP [24]	-0.196	-0.394	-0.398	0.568
TextBlob [51] (Pattern)	0.086	0.007	-0.030	0.663
TextBlob [51] (Naïve Bayes)	0.572	0.508	0.123	0.653

*Binary classification accuracy (accuracy of scores for 'positive' and 'negative' labels in the sample, disregarding 'neutral' or 'irrelevant' labels)

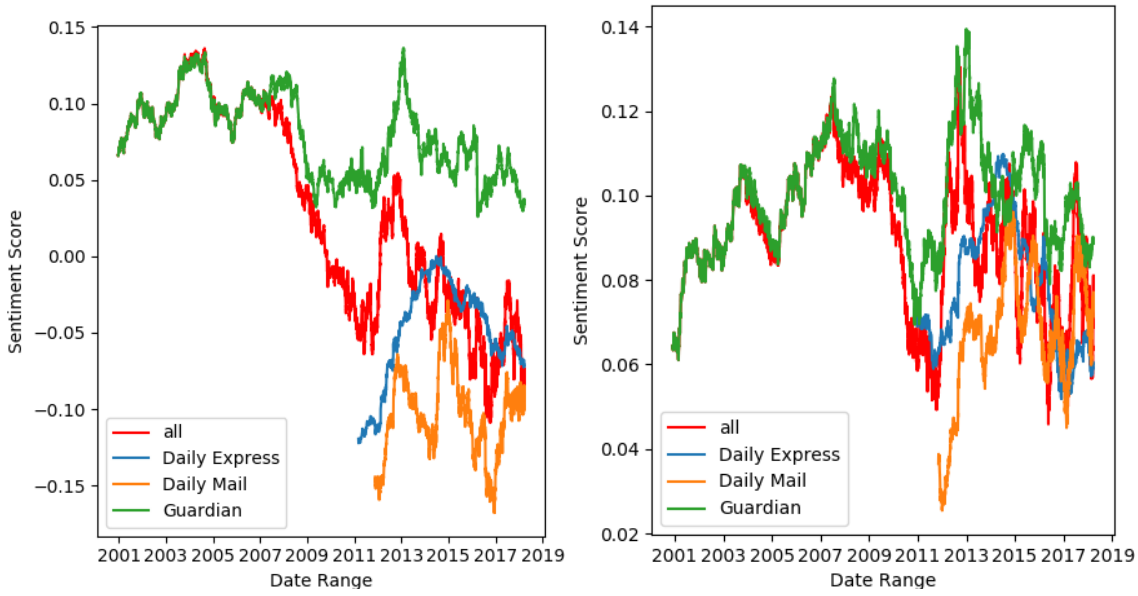
'neutral' or 'irrelevant' labels were disregarded in the accuracy measurement as they do not have an expected value, whereas 'positive' is accurate if value > 0, and 'negative' is accurate if value ≤ 0. For the following measurements, 'neutral' and 'irrelevant' labels were also disregarded:

Implementation	True Positive	False Positive	True Negative	False Negative
VADER [11]	35	4	40	16
XiaoHan [45]	22	7	37	29
Kevin Cobain's [46]	35	20	24	16
OpenAI's [48]	46	18	26	5
Stanford CoreNLP [24]	13	3	41	38
TextBlob [51] (Pattern)	32	13	31	19
TextBlob [51] (Naïve Bayes)	43	25	19	8

These results from the 180-article sample indicated that VADER [11], followed by OpenAI's model [48], were the two best performing open-source sentiment scoring tools for this domain, as shown by the accuracy metric. Although these results are not conclusive due to the small sample size of sentences, it was a sufficient indicator of which sentiment scorers would perform better in predicting labels correctly, and therefore should be chosen for the final pipeline and experiment.

To prove whether these accuracy metrics are relevant for the experiment, both VADER and

OpenAI's sentiment scorer implementations were implemented in the final pipeline. Then, the 500-article moving average sentiment score of both scorers was plotted for the topic 'disabled':



(Left = VADER, Right = OpenAI's model)

These plots showed that the version implementing VADER was more consistent in distinguishing between different sources with a smoother trend line, whereas the line produced by the version implementing OpenAI's model had higher randomness despite using the same moving average window size of 500 articles in both plots.

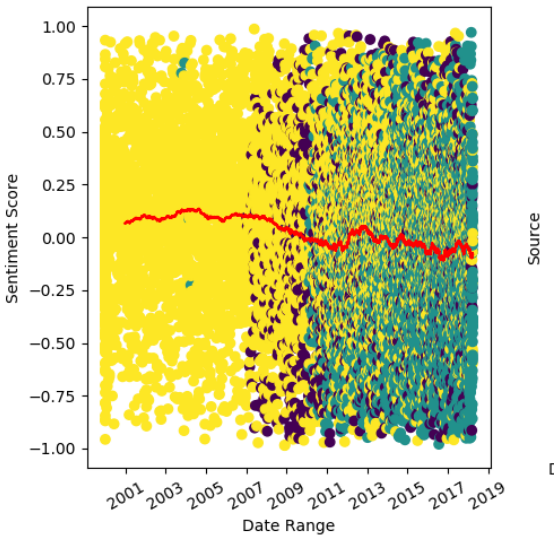
These results came as a surprise, due to the simplicity of VADER's rule-based model relative to other implemented models based on supervised machine learning or neural network approaches (refer to section 2.3.4 or 4.3.4.1 for a list and short description of these sentiment scorer implementations). The suspected reason was that these supervised machine learning and neural network models were trained on data from other domains (tweets, movie reviews, Amazon reviews, etc.), the patterns learned by the model may not necessarily translate well to this domain of sentences from news articles referring to disabilities or people with disabilities. Meanwhile, VADER's simpler rule-based model proved to be more generalisable as it simply checks for the presence of pre-defined rules that determine a general positive/negative sentiment instead of learning patterns from training data.

It is likely that a supervised machine learning or neural network model, trained on an adequately large labelled dataset from this domain, could strongly out-perform VADER. This was encouraged by the fact that the sentiment score accuracy of OpenAI's implementation, based on a neural network model, was very close to VADER's, despite the model being trained on a different domain of 82 million Amazon product reviews. However, obtaining such a large labelled dataset was not feasible for the scope of this experiment.

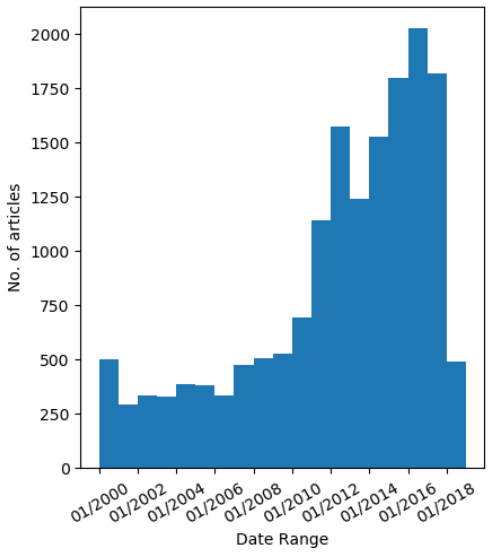
For the following sections, all sentiment score results mentioned are those scored by VADER's implementation [11]. Additionally, news articles with a VADER sentiment score of 0.0 were excluded from the plotted dataset, as they are presumed to be non-opinionated and thus irrelevant in the context of sentiment trends.

5.3 Sentiment Score: Plots and Trends

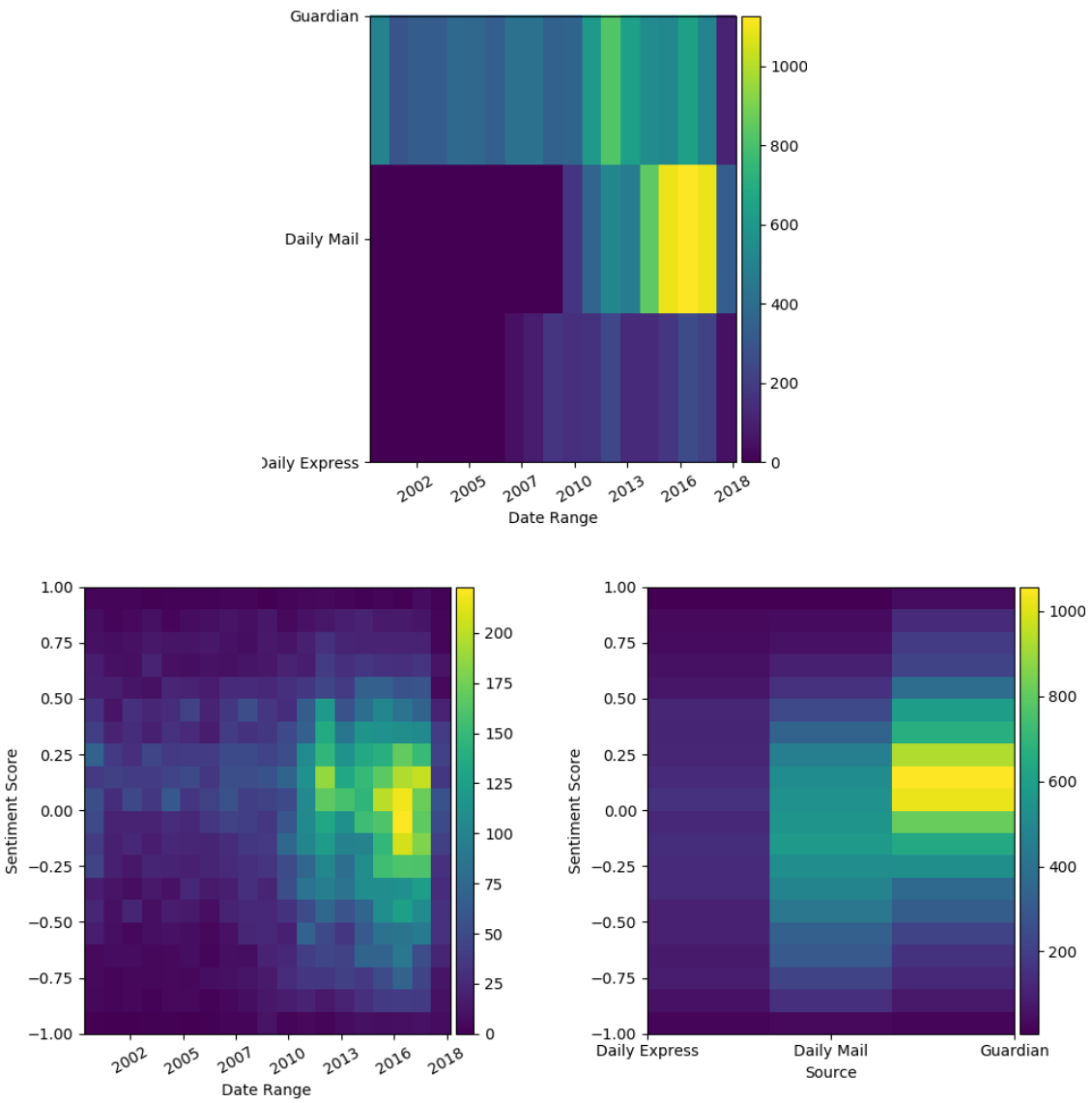
For the 'disabled' topic, the VADER sentiment scores of all 16,411 articles within the dataset were plotted with regards to each article's publication date and source (Daily Express, Daily Mail, Guardian) in several visual representations (as defined in section 4.3.5).



The first plot was a scatter plot of publication date (X) vs sentiment scores (Y), coloured based on publication source (Z). Due to the large dataset size, the scatter plot did not provide much information with regards to distributions and trends, although it showed how the Daily Express (blue-green) only retained articles from after ~2007, the Daily Mail (dark blue) only retained articles from after ~2010, and the Guardian (yellow) only retained articles from after ~2000, as mentioned in section 4.2.4. A moving average line of sentiment scores over time of all articles regardless of source, with a moving average window size of 500, was drawn over the scatter plot.



A histogram of the number of articles in each year was plotted to visualise the distribution of articles over time in the dataset. This did not necessarily reflect on how many articles were published regarding the topic for every year, as news publishers often do not retain all historical articles online, and the number of news articles retained tends to be higher for articles published more recently and vice versa. That said, a spike in the number of articles in the dataset was visible in 2012, which correspond to increased media coverage of disabilities and people with disabilities around the 2012 London Paralympics.



Three two-dimensional histograms were plotted that shows the distribution of articles for:

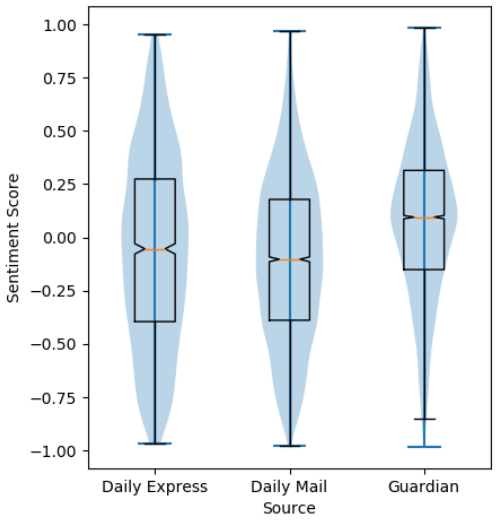
- The number of news articles for each year (X) and source (Y) in the dataset.
- The distribution of sentiment scores (Y) for each year (X).
- The distribution of sentiment scores (Y) for each source (X).

The year-source plot showed that the dataset consisted almost entirely of Guardian articles for the years 2000–2006. In 2007 Daily Express articles and in 2010 Daily Mail articles started

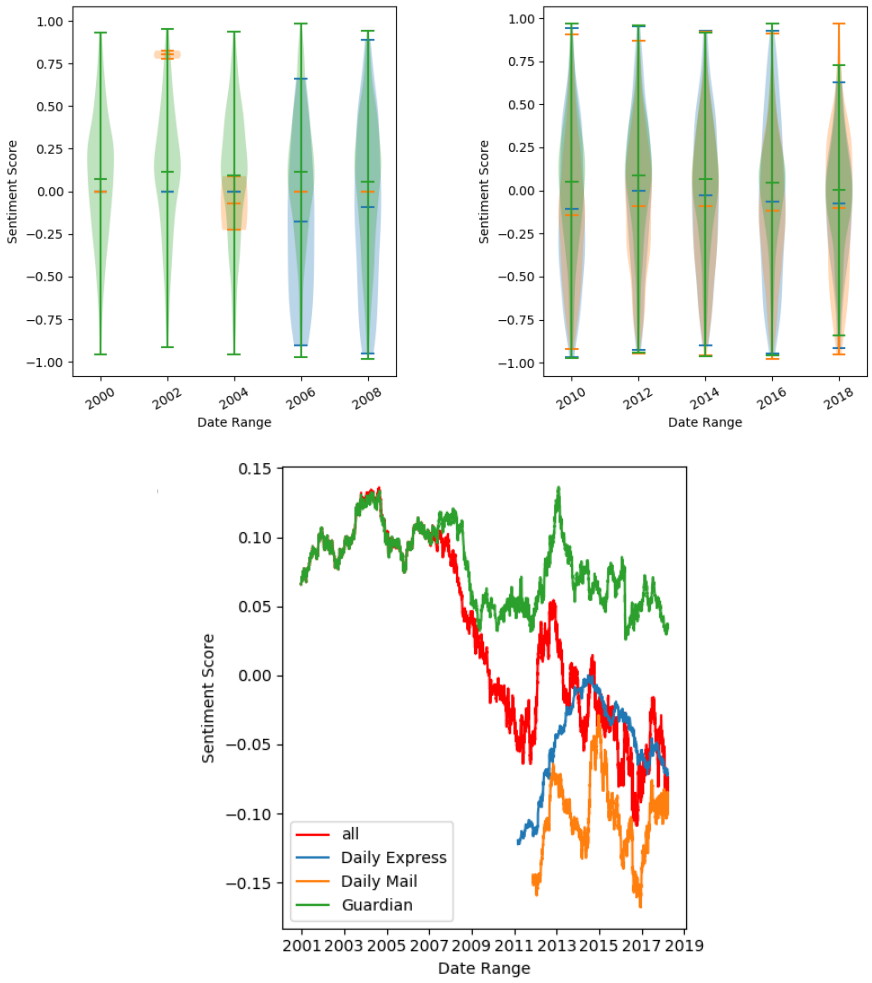
to appear, although Guardian articles still predominate between 2007–2013, until in 2014 where Daily Mail articles started to outnumber Guardian articles by roughly 2:1.

The year-sentiment plot showed that the sentiment distribution of most news articles published in 2000–2009 lie somewhere between 0.0 and 0.3. For news articles published in 2010–2018, the sentiment distribution mostly falls between -0.3 and 0.3, but with noticeably higher variance and more outliers. It also showed a slight 'drop' in the sentiment distribution or an increase of news articles with negative sentiment scores around 2016.

The source-sentiment plot showed roughly that the Guardian has a higher mean sentiment score and less variance than the Daily Mail, while the Daily Express's data was barely visible due to the lower sample size of Daily Express articles in the dataset. The uneven sample size, as there are more Guardian articles than Daily Mail or Daily Express articles over the full dataset, makes this visual representation hard to compare. The violin [54] and box-and-whiskers plots [53] provided a better visual representation of the source-sentiment data:

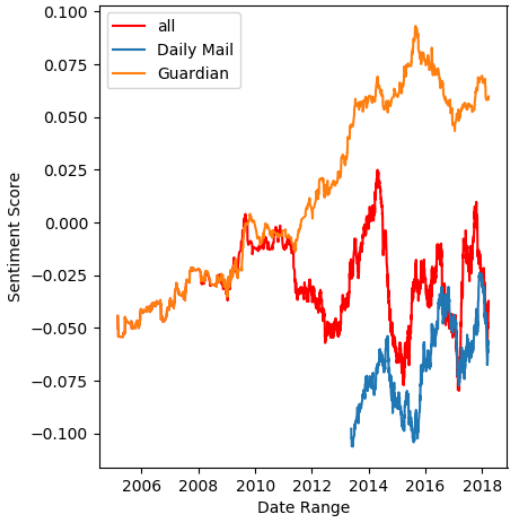


The box-and-whiskers plots showed that the Guardian's mean sentiment score (0.074) is higher than the Daily Mail's (-0.104) or the Daily Express's (-0.060). Furthermore, the violin plots showed that the Guardian's distribution is more 'compact' around the mean (std. dev. = 0.363), the Mail had a slightly higher variance (std. dev. = 0.386), and the Express had the highest variance (std. dev. = 0.447). A version that showed violin plots for each 2-year period were also plotted (green = Guardian, blue = Daily Express, orange = Daily Mail):



From the year-sentiment two-dimensional histogram plot, it looked like that there was a trend of declining sentiment scores over time. However, the moving average line plot, where each data point shows the mean sentiment score of 500 consecutive articles and the last article's date, showed that this was not necessarily the case. When only considering articles from the same source, the moving average sentiment score stayed roughly consistent. The decreasing trend in 'all' was likely better attributed to the gradually decreasing proportion of Guardian articles and the gradually increasing proportion of Daily Mail articles in the dataset.

For topics other than 'disabled', most other topics also showed a similar constant trend for articles within the same source, although this was not always the case. The topic 'autism', for instance, showed a positive trend in sentiment scores year-on-year:



Note also that the Daily Express was not plotted for this topic, due to the sample size being too small (128 articles), lower than the moving average window (316 articles on this topic). This is a common occurrence for most other topics (as mentioned in section 4.2.2).

5.4 Sentiment Score: Statistical Comparison of Sources

To test whether the differences between the distributions of each source were statistically significant, the Mann-Whitney U Test [55] was invoked. The Mann-Whitney U Test is a non-parametric statistical test where if the null hypothesis holds true, then there is no statistically significant difference between two distributions (refer to section 2.3.5 for a more formal definition). The Mann-Whitney U Test function returns a p-value score, which is a measure of the probability that the null hypothesis holds. In this experiment, two distributions were considered significantly different if the p-value returned by the Mann-Whitney U Test is lower than 0.05, where the null hypothesis would be rejected.

Below is a table of p-values from Mann-Whitney U Test results for every source combination in the dataset, for every topic (including topics other than 'disabled'):

Topic	Guardian > Daily Mail	Guardian > Daily Express	Daily Express > Daily Mail
Disabled	**$2.66 * 10^{-167}$**	**$2.29 * 10^{-35}$**	**0.000108**
Autism	**$3.00 * 10^{-12}$**	0.137	0.184
Blind	0.335 **	**$1.62 * 10^{-9}$**	**$6.57 * 10^{-9}$ ****
Cerebral Palsy	0.469	0.156 **	0.169
Deaf	**$6.30 * 10^{-10}$**	0.141	0.0833
Developmental Delay	0.383	**0.00179**	**0.00326 ****
Dyslexia	0.0740	0.362	0.464 **
Epilepsy	**$3.39 * 10^{-5}$**	0.0579	0.322 **
Mental Illness	**$3.54 * 10^{-76}$**	**$1.51 * 10^{-9}$**	0.102 **
Mute	**$6.78 * 10^{-5}$**	**0.0311**	0.346 **
Paralysis	**$2.73 * 10^{-5}$**	**0.00909**	0.0827 **
Speech Impairment	0.235	**0.0200**	0.0546 **

** Indicates where the reverse assumption is true (e.g. Daily Mail > Daily Express instead of Daily Express > Daily Mail)

Cases where a significant distinction between the two distributions existed is highlighted in bold in the above table.

The table showed that a statistically significant distinction could be proven between the sentiment score distributions of two sources for 17 / 36 of cases. In particular, Guardian > Daily Mail was significant for 7 / 12 cases, Guardian > Daily Express was significant for 7 / 12 cases, Daily Express > Daily Mail was significant for 1 / 12 cases, and Daily Mail > Daily Express was significant for 2 / 12 cases. Comparing these values to the size of each dataset (section 4.2.3) showed that lower p-values tend to correlate well with larger sample sizes, i.e. the higher the sample size of both distributions, the higher the chance that a statistically significant difference exists.

For the 'disabled' topic, the Mann-Whitney U Test was also performed to compare the sentiment score distributions of Daily Mail, Daily Express, and Guardian articles for each year's subset between 2007 and 2018. Before 2007, there were not enough non-Guardian articles in the dataset to make a comparison. The p-values of these comparisons are shown below:

Topic	Guardian > Daily Mail	Guardian > Daily Express	Daily Express > Daily Mail
2007	N/A	**$5.24 * 10^{-6}$**	N/A
2008	N/A	0.0739	N/A
2009	N/A	**$2.24 * 10^{-5}$**	N/A
2010	**$6.00 * 10^{-7}$**	**$6.89 * 10^{-6}$**	0.360 **
2011	**$5.73 * 10^{-16}$**	**$9.73 * 10^{-5}$**	**0.0355**
2012	**$4.44 * 10^{-17}$**	**0.00371**	**0.000739**
2013	**$5.03 * 10^{-16}$**	**0.00351**	0.0827
2014	**$1.27 * 10^{-9}$**	**0.00208**	0.321
2015	**$6.87 * 10^{-17}$**	**0.00855**	**0.00718**
2016	**$4.63 * 10^{-26}$**	**$4.74 * 10^{-6}$**	0.0576
2017	**$3.39 * 10^{-13}$**	**0.00244**	0.0867
2018*	0.122	0.120	0.328

* 2018 data is incomplete and would only include articles up to (approximately) end of March.

** Indicates where the reverse assumption is true (e.g. Daily Mail > Daily Express instead of Daily Express > Daily Mail)

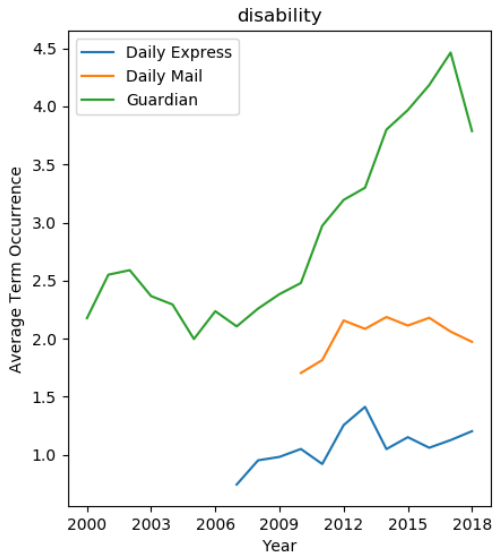
5.5 Key Terms: Plots and Trends

Apart from sentiment score analyses, NLP approaches could also be applied to extract other syntax-based features and analyse trends of these features. Features based on counting words, or term frequency, are a mainstay of NLP research. The pipeline also extracted the term frequency of all terms that match a key term's lemma, from all news articles (refer to section 4.3.3 for details). This information was used to plot trends on key term usage over time.

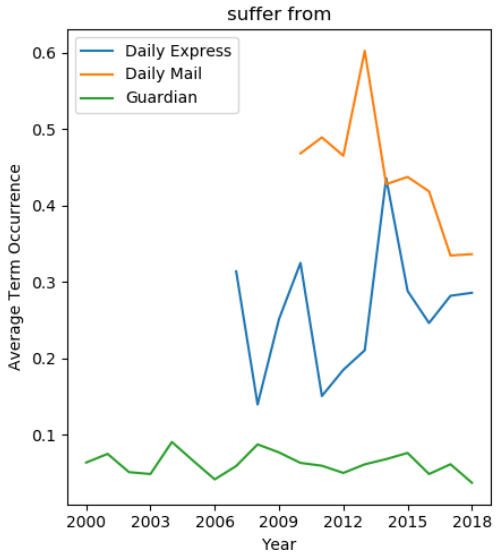
A weak custom stemmer that only stems plural and past/future tense forms and no other affixes was used to group equivalent terms together regardless of context, without losing additional meaning from affixes. For example, 'illness' and 'illnesses' are equivalent, but 'ill' and 'illness' are still considered distinct. Terms were also converted to lower case to ignore capitalisation, thus e.g. 'Disability' and 'disability' are equivalent.

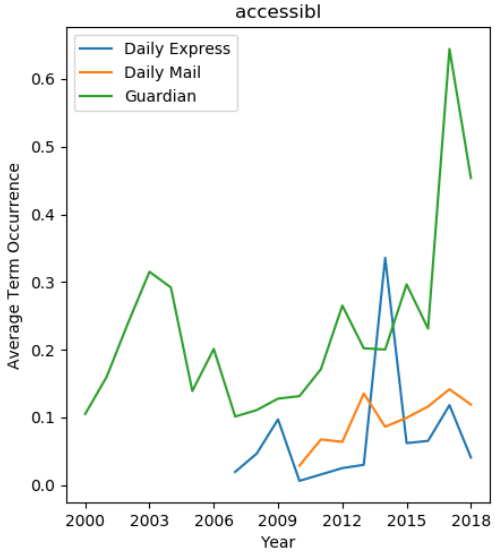
For each term, a measure of average term occurrence, which measures the expected number of occurrences of a term within any given article for a given source and year, was calculated for each year between 2000 and 2018. Refer to section 4.3.5 for a formal definition of this metric.

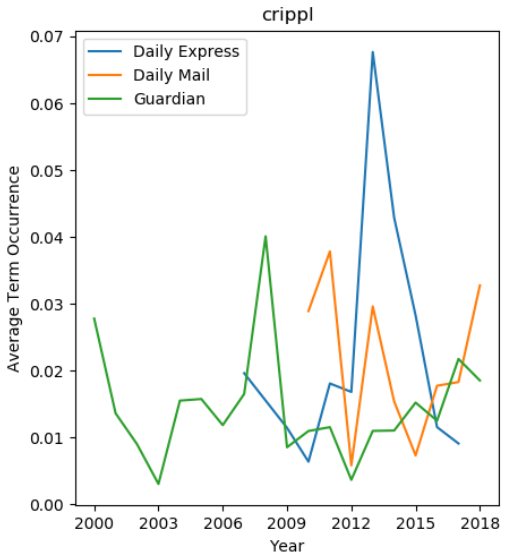
With this approach, the year-on-year average term occurrence trends of each key term in the 'disabled' topic were plotted in line graphs, one for each key term:

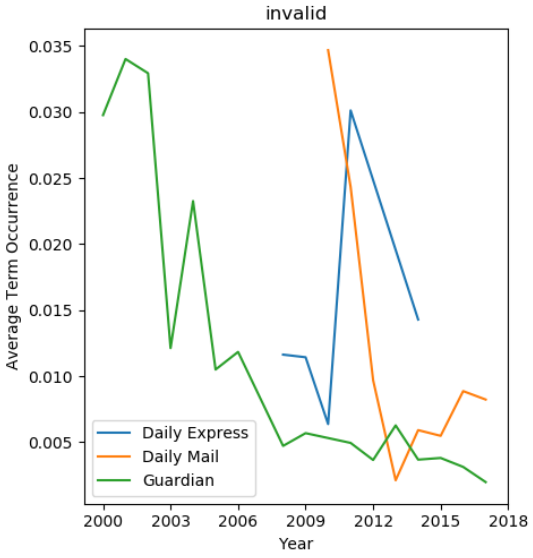

disability
Daily Express
Daily Mail
Guardian
Average Term Occurrence
Year

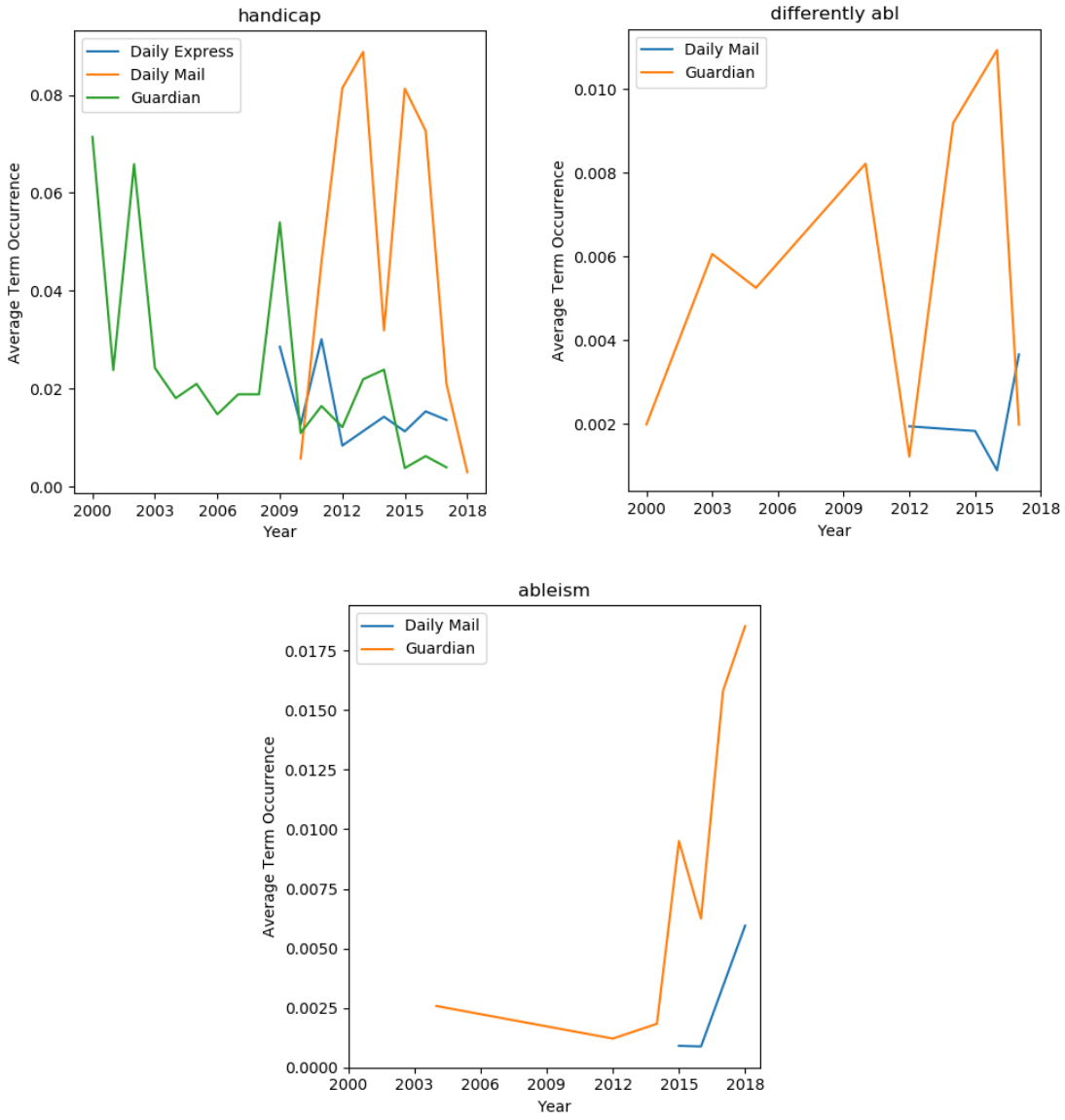

disabl
Daily Express
Daily Mail
Guardian
Average Term Occurrence
Year


suffer from
Daily Express
Daily Mail
Guardian
Average Term Occurrence
Year


accessibl
Daily Express
Daily Mail
Guardian
Average Term Occurrence
Year


crippl
Daily Express
Daily Mail
Guardian
Average Term Occurrence
Year


invalid
Daily Express
Daily Mail
Guardian
Average Term Occurrence
Year



These line plots showed that, for most key terms, the average term occurrence trends mostly stay roughly constant year-on-year for the same publisher. While this exact approach was successfully used to detect cultural changes in British media [5], in this case, the sampled time period was much shorter for significant linguistic changes to have occurred (18 vs 150 years).

In this case, increasing/decreasing year-on-year trends were only visible for a select few terms. The usage of the terms 'invalid' and 'handicap(ped)', for example, showed a rapidly decreasing trend between 2000 and 2018 on the Guardian. 'accessible', on the other hand, showed an increasing trend on the Guardian between 2006 and 2018. While the data was less consistent with the Daily Express and Daily Mail, this was likely caused by a lack of data for these sources before 2007 and 2010 respectively, and a smaller overall sample size especially for the Daily Express.

These plots also showed variations in term usage between different publishers. For example, the Guardian referred to 'disability' or 'disabl(ed)' by name consistently more often than the Daily Mail or Daily Express, and used the term 'suffers(s) from' consistently less often.

Chapter 6

Conclusions

6.1 Achievements

The aim of this project was to explore the feasibility of utilising NLP technologies to discover trends such as how term popularity and sentiment varies over time and between different publications, within the context of news articles relating to disabilities and people with disabilities in British online news media. The results of this experiment showed that this was feasible.

By analysing a dataset of 16,411 news articles related to the key terms 'disabled', 'disability', 'handicapped', 'cripple', 'invalid', 'accessible', 'ablism', and 'ableism'; the results in section 5.3 plotted trends showing the variation of modelled sentiment scores across three different news publishers (Daily Express, Daily Mail, and Guardian) and over time. The results of Mann-Whitney U statistical test in section 5.4 showed that the differences of sentiment score distributions between the three publications were statistically significant for the 'disabled' topic, and showed that 'Guardian > Daily Mail' and 'Guardian > Daily Express' was true for every year between 2010 and 2017. Furthermore, the results of analysing average term occurrences of key terms, as shown in section 5.5, identified increasing or decreasing trends for the terms 'invalid', 'handicap(ped)', and 'accessibl(e)' for the Guardian; and showed variations in term usage/popularity between different publishers.

This experiment was repeated across 11 other topics (as defined in section 4.2.2), with varying degrees of success. The lower sample size of news articles related to other topics in the dataset and the decreased effectiveness of the filter with regards to topics with more ambiguous keywords (e.g. 'blind', 'mute', 'paralysed', and 'stutter') were major factors in the variability of results. The full results of the experiment for all 12 topics are available in the Appendix.

The experiment showed that it was feasible to collect and analyse a large dataset of 305,185 news articles, reduced to 48,967 articles after filtering, within a reasonable time frame using consumer-grade hardware (Intel i7-6700HQ CPU @ 2.60GHz, NVIDIA GeForce GTX 1060 GPU, 200 Mbit/s download speed). Additionally, the vast majority of the time was spent on data collection from online sources. An analysis of an existing news article corpus would take less time, approximately 0.37 seconds per article in extracting text features, with negligible sentiment scoring runtime using VADER. With more powerful hardware available to large institutions, this approach should scale well to analyse millions of documents in reasonable time.

6.2 Evaluation

This experiment showed that it was feasible to apply existing open-source NLP technologies to discover trends on a large dataset of news articles relating to disabilities and people with disabilities, which achieved the primary aim of this project. The solution developed to perform this experiment delivered in performing the data collection and filtering of 305,185 news articles before filtering; and the extraction of relevant sentences and key term frequencies, sentiment scoring, statistical analysis, and data visualisation of 48,967 news articles after filtering. The modular approach to solution design, with five separate components for each goal or task (data collection, filtering, feature extraction, sentiment scoring, and data analysis/visualisation) were ideal in this research, as it enabled experimentation e.g. trying multiple sentiment scorer implementations and data visualisation plots, without affecting or needing to rewrite or recompute code for other components. This solution achieved the project's goals sufficiently as a proof of concept and delivered meaningful results.

The choice of NLP techniques and technologies, extracted features, statistical metrics, and visualisation tools were sufficient and achieved results, but leaves room for further improvement. The main limitation of this research was the lack of labelled data for both filtering off-topic articles and sentiment scoring, which led to inconsistencies in both tasks. The filter had issues with ambiguous terms such as 'blind', 'mute', 'paralysed', and 'stutter'; a supervised filter may be able to overcome this issue by taking the word's context in the sentence into account e.g. by also evaluating other words in the sentence, and provide a better evaluation of filter accuracy. Given this constraint, the filter worked adequately well to prepare the dataset for the experiment, where limited sampling of on-topic/off-topic articles showed that the filter was sufficiently accurate for most topics.

Without labelled data for sentiment scoring, the only possible options were to use generalised supervised models trained on other domains or to use a less complex model based on rule matching. It was found that a less complex model based on rule matching, VADER [11], outperformed more complex supervised models based on statistical classifiers or neural networks trained on other domains, such as Amazon reviews or tweets. However, VADER was still highly inconsistent in this domain, with an accuracy of 0.789, and although it was sufficient to discover trends in this experiment, it is likely that a domain-specific supervised model would outperform it. With a more consistent sentiment scorer, it is likely that the variance of sentiment scores within subsets would be lower, and the trend lines generated in data visualisation would be more consistent.

The usage of different NLP libraries in separate components such as NLTK's stemmer for filtering and SpaCy's lemmatiser for feature extraction also left occasional inconsistencies, such as rare cases of the feature extractor not finding any relevant sentences in articles the filter deemed on-topic.

6.3 Future Work

The results showed that this computational NLP-based approach was effective in analysing news media with regards to the media representation of disabilities and people with disabilities. This could have a substantial impact on how research will be conducted for similar studies. Past studies

(such as [7]–[9], [14]) could be revisited using computational NLP-based approaches to analyse a much larger sample size of articles, which would improve the certainty and representativeness of the conclusion, and possibly identify trends by varying for independent variables such as time and source.

As mentioned in section 6.2, the lack of labelled domain-specific dataset of sentences from news articles relating to disabilities with 'positive' or 'negative' labels limited the sentiment scores' accuracy in this experiment. While exploring general-purpose open-source sentiment scorer implementations, it was found that a simple rule-based model, VADER [11], outperformed more sophisticated supervised machine learning or neural network models for the sentiment scoring task, as they were trained with labelled datasets from other domains. However, one of the neural network based models, OpenAI [48], came close to VADER in accuracy, despite being trained on a separate domain (Amazon reviews), suggesting that a domain-specific supervised model would strongly outperform VADER given sufficient high-quality training data.

If an adequately large dataset of labelled sentences for this domain could be compiled, future work could use this dataset to train a supervised model, which would improve upon the accuracy of sentiment scores used in this experiment. With a more accurate and more consistent sentiment scorer, it is likely that the variance of sentiment scores within subsets would be lower, and it would be possible to derive clearer trends and obtain statistically significant comparisons on subsets with lower sample sizes, e.g. smaller time intervals.

Similarly, a supervised filter, trained on a labelled dataset of on-topic and off-topic news articles, would likely improve filtering accuracy and reduce the proportion of off-topic articles in the filtered dataset, especially for trickier topics with ambiguous terms such as 'blind' or 'mute'.

Another potential application is to develop a public interface that allows users to retrieve the results of analyses performed in this experiment for other domains, given a list of topics, key terms, and query terms, and/or an existing text corpus provided by the user. The solution described in this project could be generalised with an interface, where users can provide their own scrapers that implement a well-documented set of functions for other news websites/sources, define their own list of topics, key terms, and query terms, and possibly even change the independent variables and subset intervals for the analysis. It would also need to provide a clear documentation on its usage and expected input structure/format. Such a solution would provide researchers with the tools to perform similar research with ease, and possibly build on the tool as a part of other projects, using the tool as a component.

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

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

[4] 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).

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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`.

[11] 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`.

[12] 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.

[13] 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.

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

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

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

[22] 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.

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

[24] 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`.

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

[26] 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`.

[27] 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.

[28] 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.

[29] 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).

[30] 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).

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

[32] Anaconda, Inc. (2018). Anaconda Cloud, [Online]. Available: https://anaconda.org/.

[33] Python Software Foundation. (2018). Python, [Online]. Available: https://www.python.org/.

[34] 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.

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

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

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

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

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

[40] 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.

[41] 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.

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

[43] 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.

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

[45] 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.

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

[47] 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.

[48] 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`.

[49] 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.

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

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

[52] J. F. del Río. (Sep. 22, 2016). Getting started with the C side of numpy, [Online]. Available: `https://github.com/numpy/numpy/wiksi/Getting-Started-With-The-C-side-of-Numpy`.

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

[54] 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.

[55] 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.

[56] 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/`.

[57] ——, (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/.

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

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

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

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

[62] 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.

Appendix A

Evaluation Results

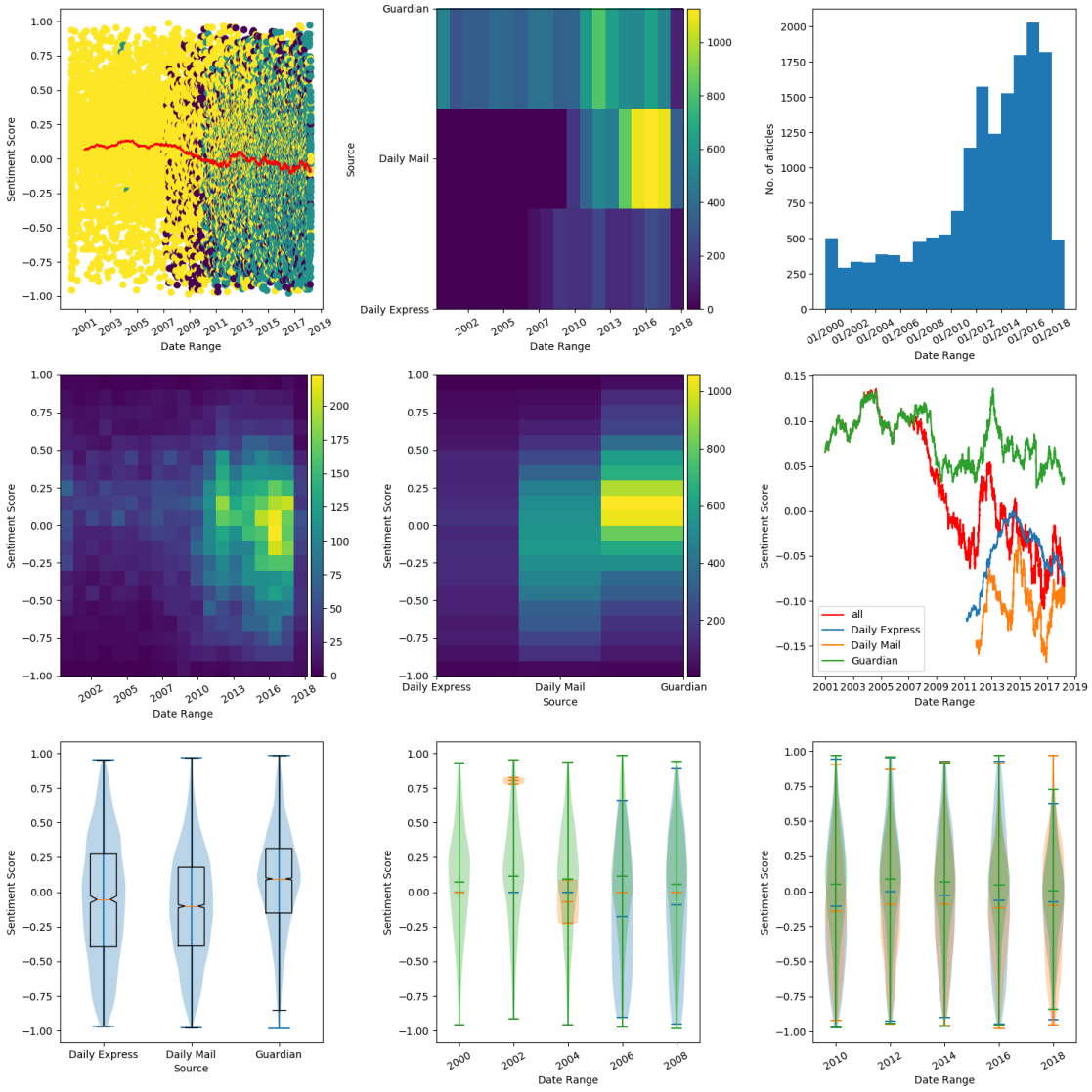
A.1 Topic: ‘disabled’

Key Terms: ‘disabled’, ‘disability’, ‘handicapped’, ‘cripple’, ‘invalid’, ‘accessible’, ‘ablism’, ‘ableism’, ‘differently abled’

Query Terms: ‘disabled’, ‘disability’, ‘ablism’, ‘ableism’, ‘differently abled’

Sample size, n = 16,411

A.1.1 Sentiment Score Plots



A.1.2 Mann-Whitney U Test Results (p-values)

Topic	Guardian > Daily Mail	Guardian > Daily Express	Daily Express > Daily Mail
All	$2.66 * 10^{-167}$	$2.29 * 10^{-35}$	0.000108
2007	N/A	$5.24 * 10^{-6}$	N/A
2008	N/A	0.0739	N/A
2009	N/A	$2.24 * 10^{-5}$	N/A
2010	$6.00 * 10^{-7}$	$6.89 * 10^{-6}$	0.360 **
2011	$5.73 * 10^{-16}$	$9.73 * 10^{-5}$	0.0355
2012	$4.44 * 10^{-17}$	0.00371	0.000739
2013	$5.03 * 10^{-16}$	0.00351	0.0827
2014	$1.27 * 10^{-9}$	0.00208	0.321
2015	$6.87 * 10^{-17}$	0.00855	0.00718
2016	$4.63 * 10^{-26}$	$4.74 * 10^{-6}$	0.0576
2017	$3.39 * 10^{-13}$	0.00244	0.0867
2018*	0.122	0.120	0.328

* 2018 data is incomplete and would only include articles up to (approximately) end of March.

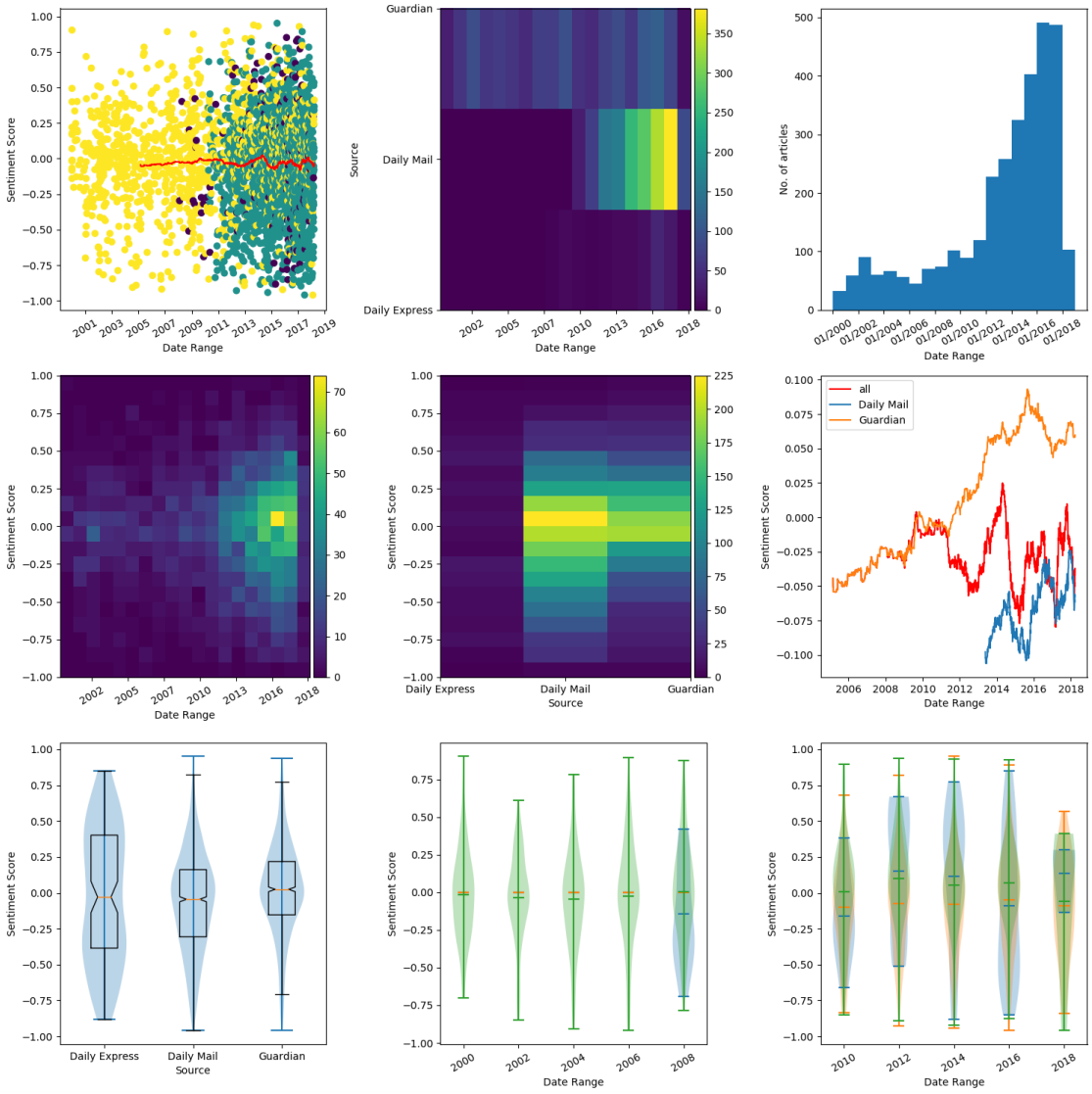
** Indicates where the reverse assumption is true (e.g. Daily Mail > Daily Express instead of Daily Express > Daily Mail)

A.1.3 Key Term Trend Plots

A.2 Topic: ‘autism’

Key Terms: ‘autism’, ‘autistic’, ‘asperger\’s’, ‘ASD’
Query Terms: ‘autism’, ‘autistic’, ‘asperger\’s’, ‘ASD’
Sample size, n = 3,161

A.2.1 Sentiment Score Plots

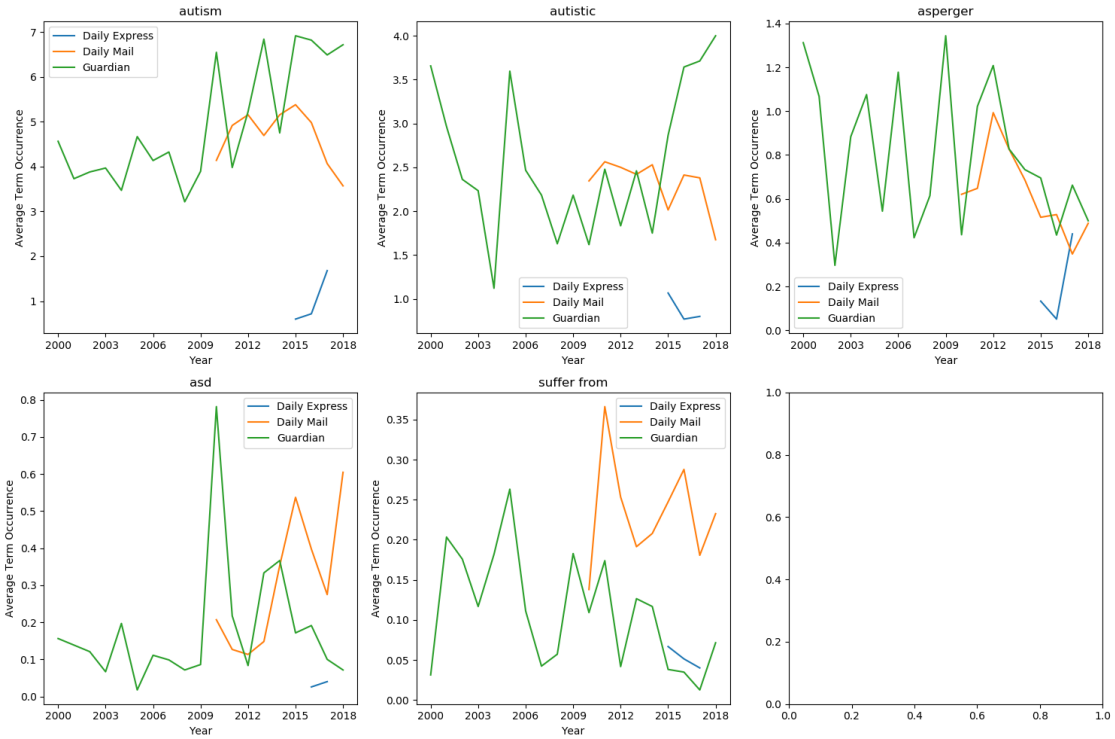


A.2.2 Mann-Whitney U Test Results (p-values)

Topic	Guardian > Daily Mail	Guardian > Daily Express	Daily Express > Daily Mail
All	$3.00 * 10^{-12}$	0.137	0.184
2010	0.100	N/A	N/A
2011	0.0649	N/A	N/A
2012	0.000150	N/A	N/A
2013	$1.52 * 10^{-5}$	N/A	N/A
2014	0.00162	N/A	N/A
2015	0.000508	N/A	N/A
2016	0.00760	0.0121	0.0432 **
2017	0.000114	0.0667	0.352

** Indicates where the reverse assumption is true (e.g. Daily Mail > Daily Express instead of Daily Express > Daily Mail)

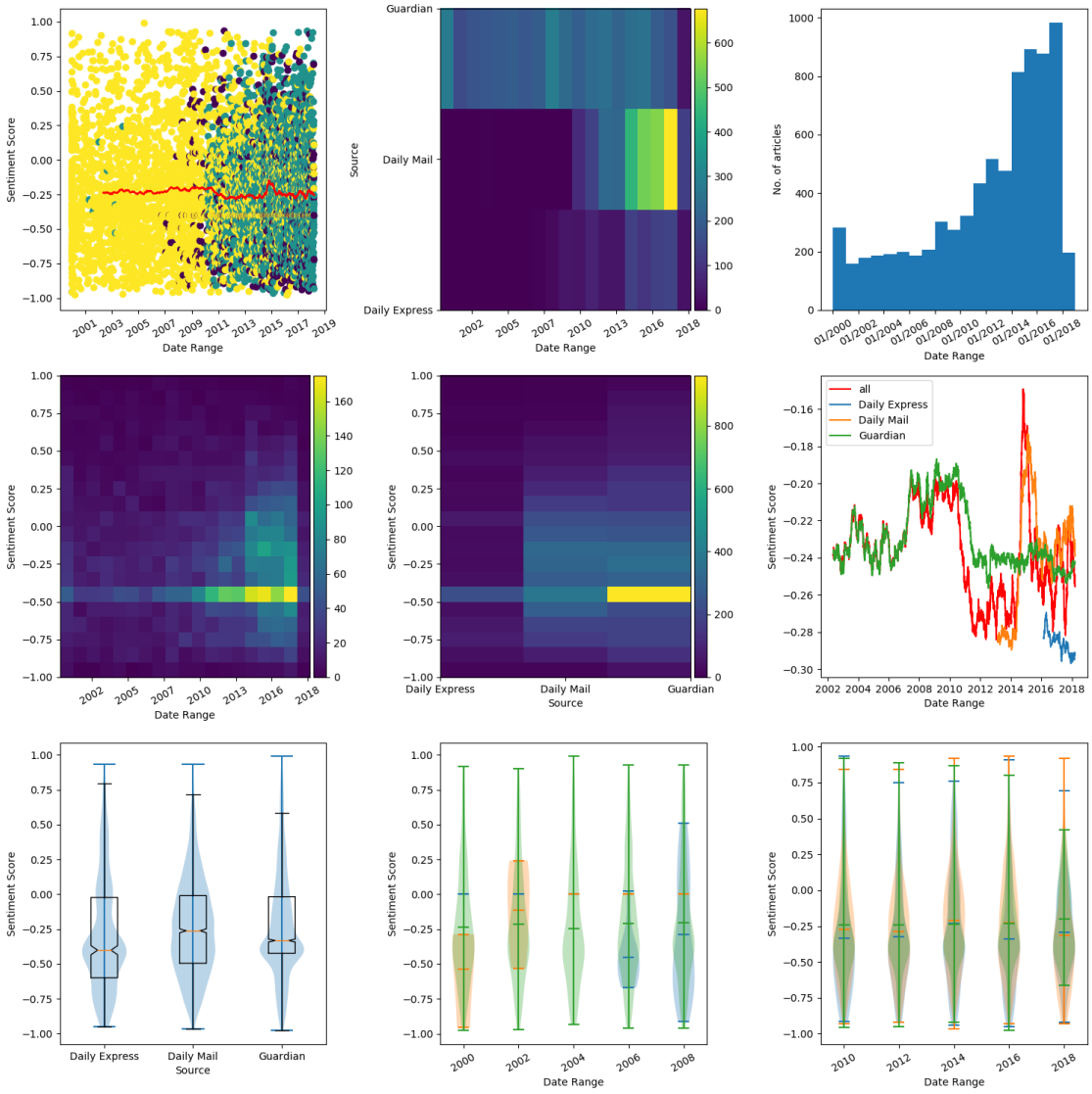
A.2.3 Key Term Trend Plots



A.3 Topic: ‘blind’

Key Terms: ‘blind’, ‘blindness’, ‘blindism’, ‘visual impairment’, ‘partially sighted’, ‘vision loss’
Query Terms: ‘blind’, ‘blindness’, ‘visual impairment’, ‘partially sighted’, ‘visually impaired’
Sample size, n = 7,677

A.3.1 Sentiment Score Plots



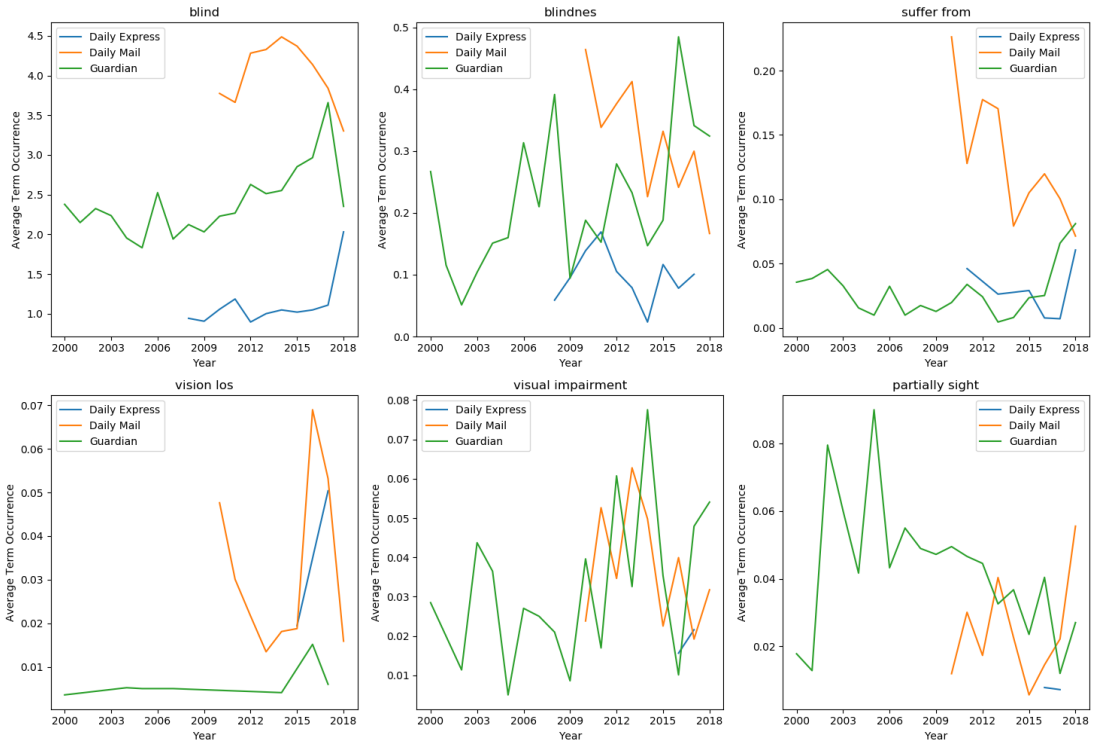
A.3.2 Mann-Whitney U Test Results (p-values)

Topic	Guardian > Daily Mail	Guardian > Daily Express	Daily Express > Daily Mail
All	0.335 **	$1.62 * 10^{-9}$	$6.57 * 10^{-9}$ **
2009	N/A	0.339	N/A
2010	0.388	0.0331	0.0720 **
2011	0.493	0.0101	0.0149 **
2012	0.00585	0.00586	0.220 **
2013	0.330 **	0.163	0.158 **
2014	0.0670 **	0.263	0.0857 **
2015	0.181 **	0.214	0.125 **
2016	0.455	0.00119	0.000966 **
2017	0.421	$3.16 * 10^{-5}$	$1.46 * 10^{-5}$ **
2018*	0.0118	0.0919	0.246

* 2018 data is incomplete and would only include articles up to (approximately) end of March.

** Indicates where the reverse assumption is true (e.g. Daily Mail > Daily Express instead of Daily Express > Daily Mail)

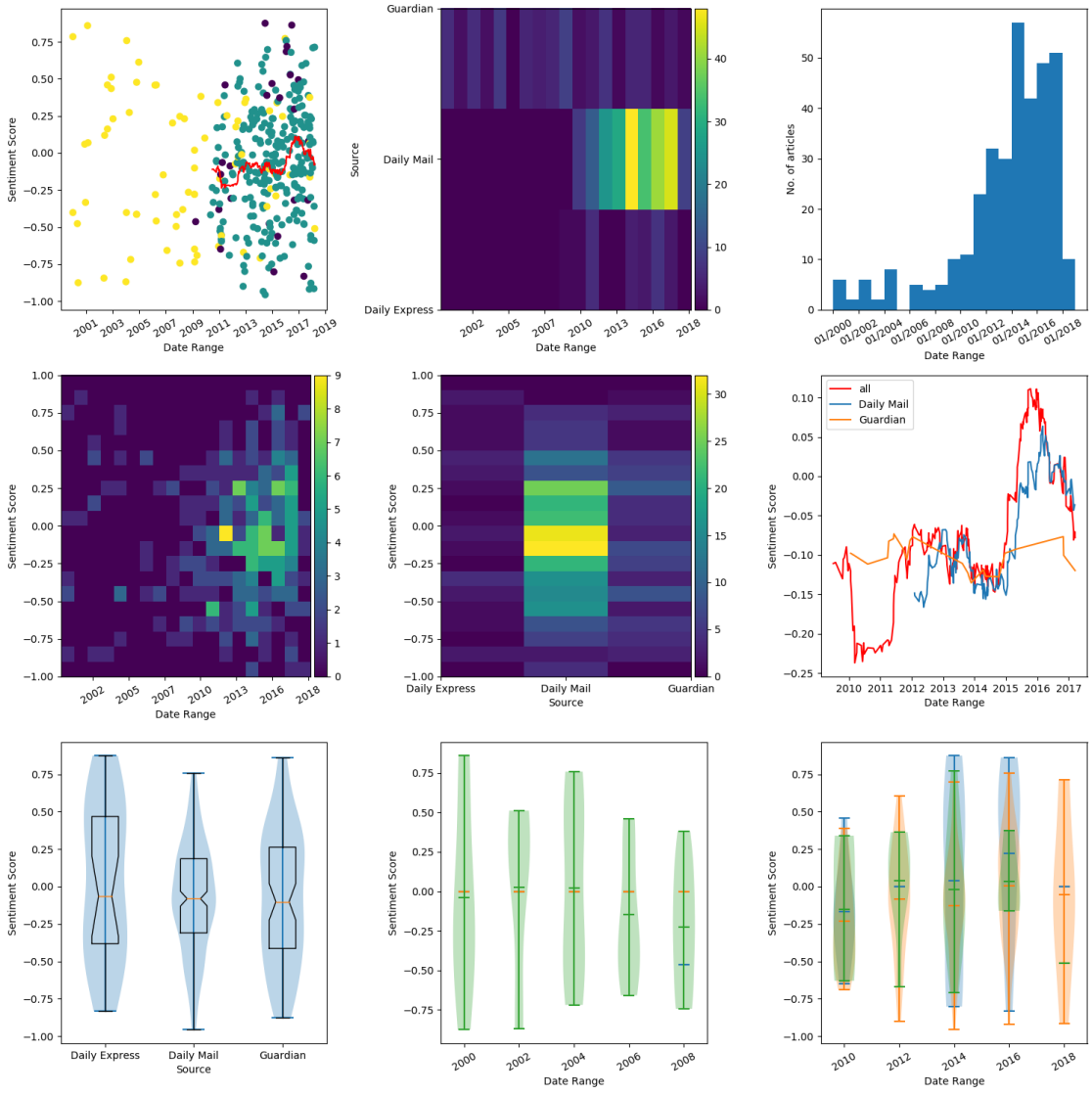
A.3.3 Key Term Trend Plots



A.4 Topic: 'cerebral palsy'

Key Terms: 'cerebral palsy', 'spastic'
Query Terms: 'cerebral palsy', 'spastic'
Sample size, n = 353

A.4.1 Sentiment Score Plots



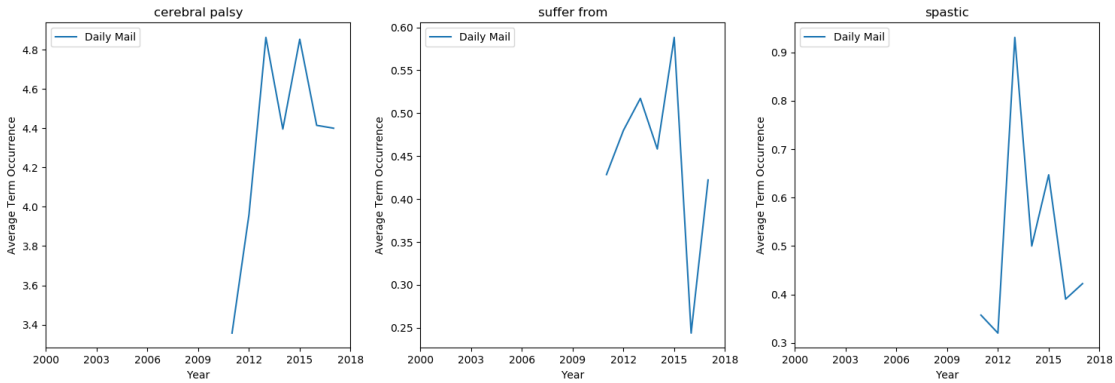
A.4.2 Mann-Whitney U Test Results (p-values)

Topic	Guardian > Daily Mail	Guardian > Daily Express	Daily Express > Daily Mail
All	0.469	0.156 **	0.169

Insufficient sample size for year-by-year comparisons. (n=353)

** Indicates where the reverse assumption is true (e.g. Daily Mail > Daily Express instead of Daily Express > Daily Mail)

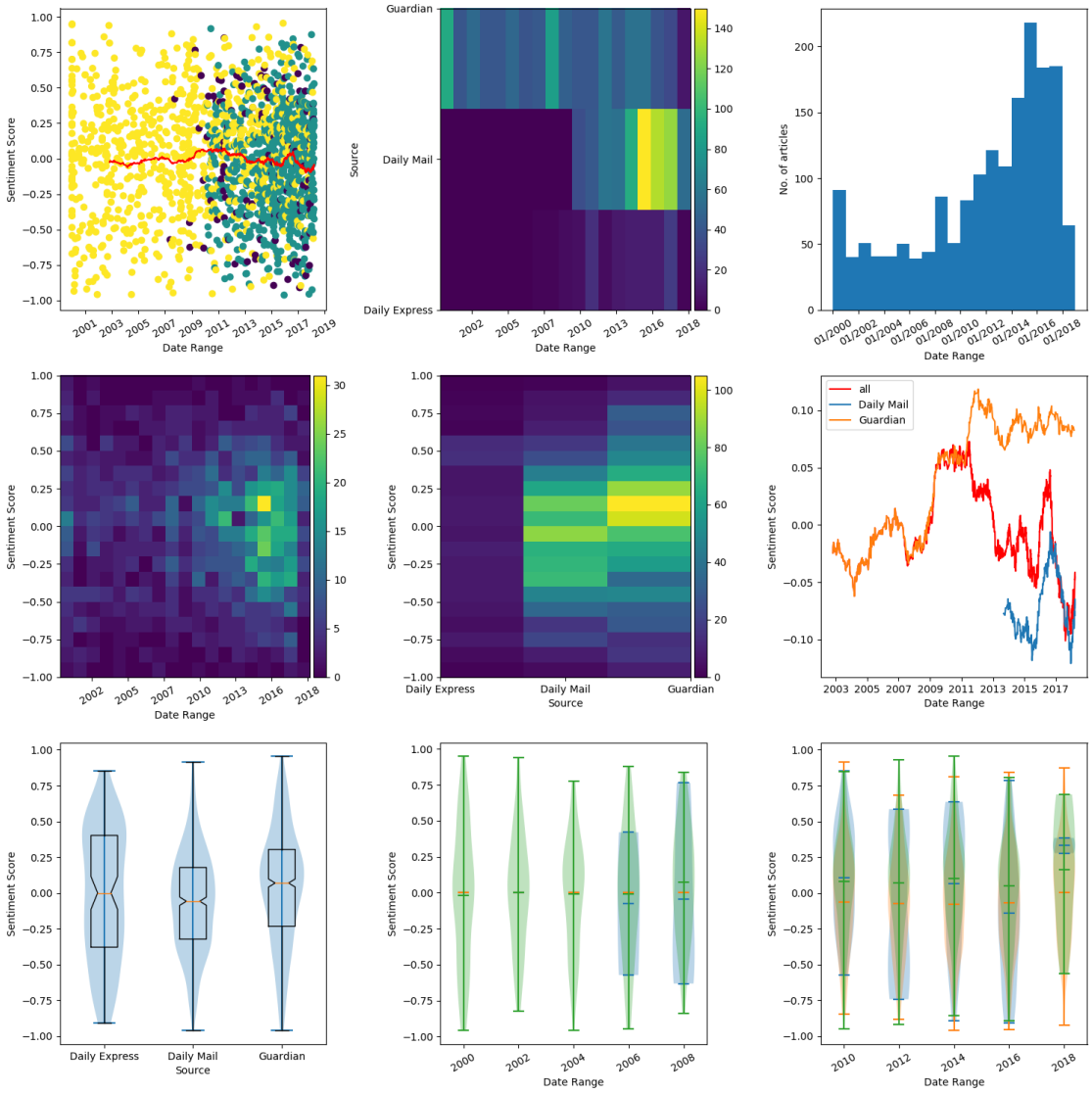
A.4.3 Key Term Trend Plots



A.5 Topic: ‘deaf’

Key Terms: ‘deaf’, ‘deafness’, ‘hearing impaired’, ‘hard of hearing’, ‘hearing loss’
Query Terms: ‘deaf’, ‘deafness’, ‘hearing impairment’, ‘hard of hearing’, ‘hearing impaired’
Sample size, n = 1,762

A.5.1 Sentiment Score Plots

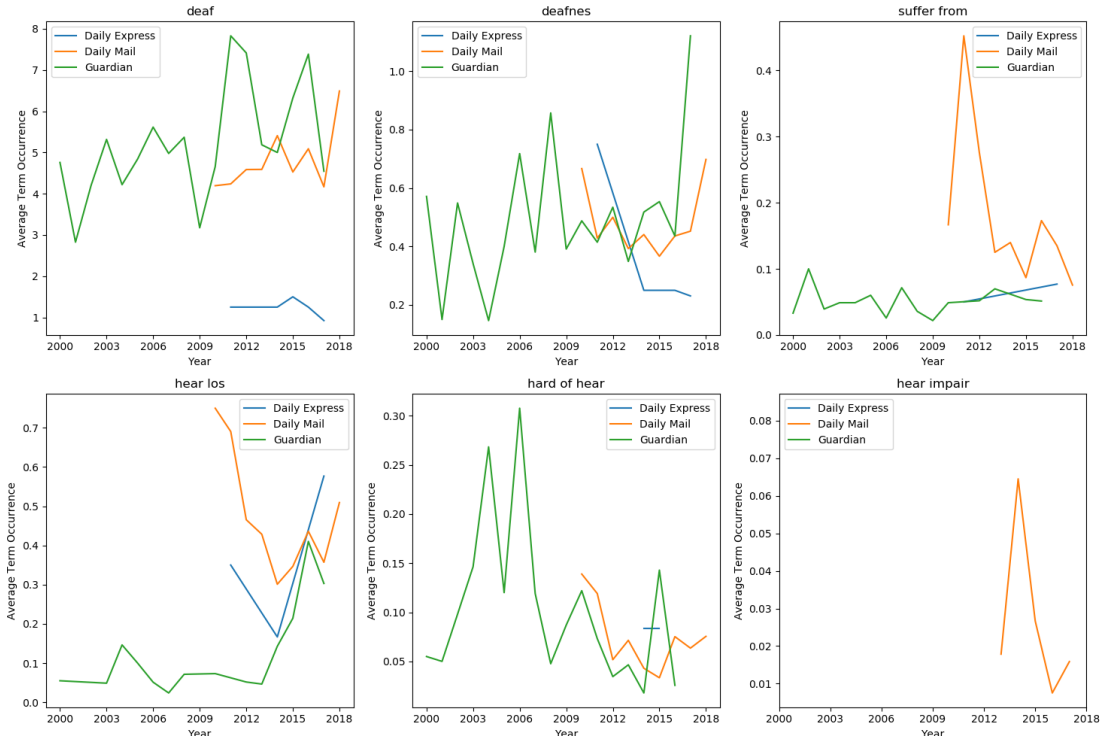


A.5.2 Mann-Whitney U Test Results (p-values)

Topic	Guardian > Daily Mail	Guardian > Daily Express	Daily Express > Daily Mail
All	$6.30 * 10^{-10}$	0.141	0.0833
2010	0.263	N/A	N/A
2011	0.00262	N/A	N/A
2012	0.00349	N/A	N/A
2013	0.0403	N/A	N/A
2014	0.000131	N/A	N/A
2015	0.00260	N/A	N/A
2016	0.0437	N/A	N/A
2017	0.0307	0.0480	0.233 **

** Indicates where the reverse assumption is true (e.g. Daily Mail > Daily Express instead of Daily Express > Daily Mail)

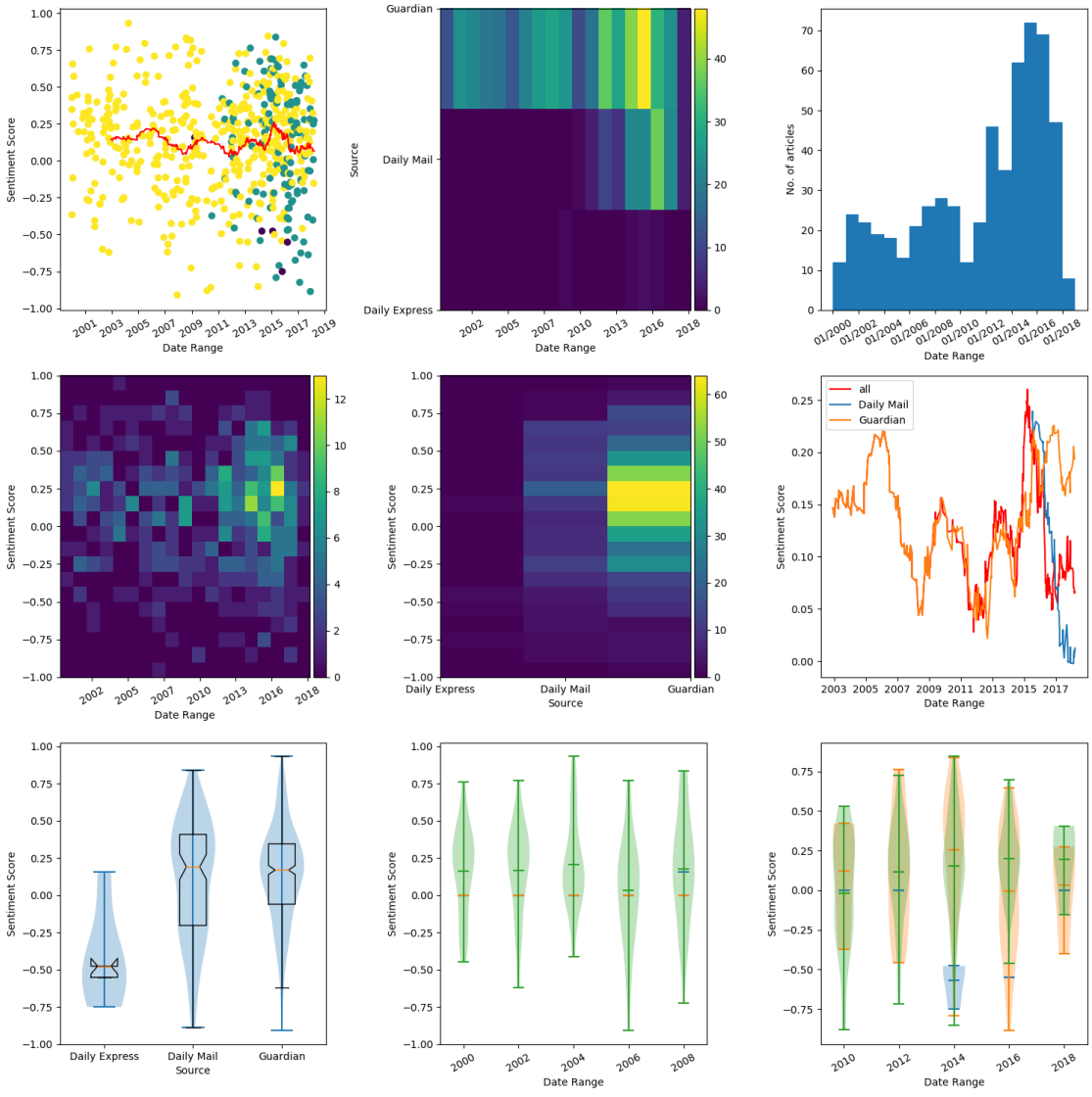
A.5.3 Key Term Trend Plots



A.6 Topic: ‘developmental delay’

Key Terms: ‘developmental delay’, ‘developmental disability’, ‘developmental disorder’, ‘learning disability’, ‘slow learner’, ‘intellectual disability’
Query Terms: ‘developmental delay’, ‘developmental disability’, ‘developmental disorder’, ‘learning disability’
Sample size, n = 582

A.6.1 Sentiment Score Plots

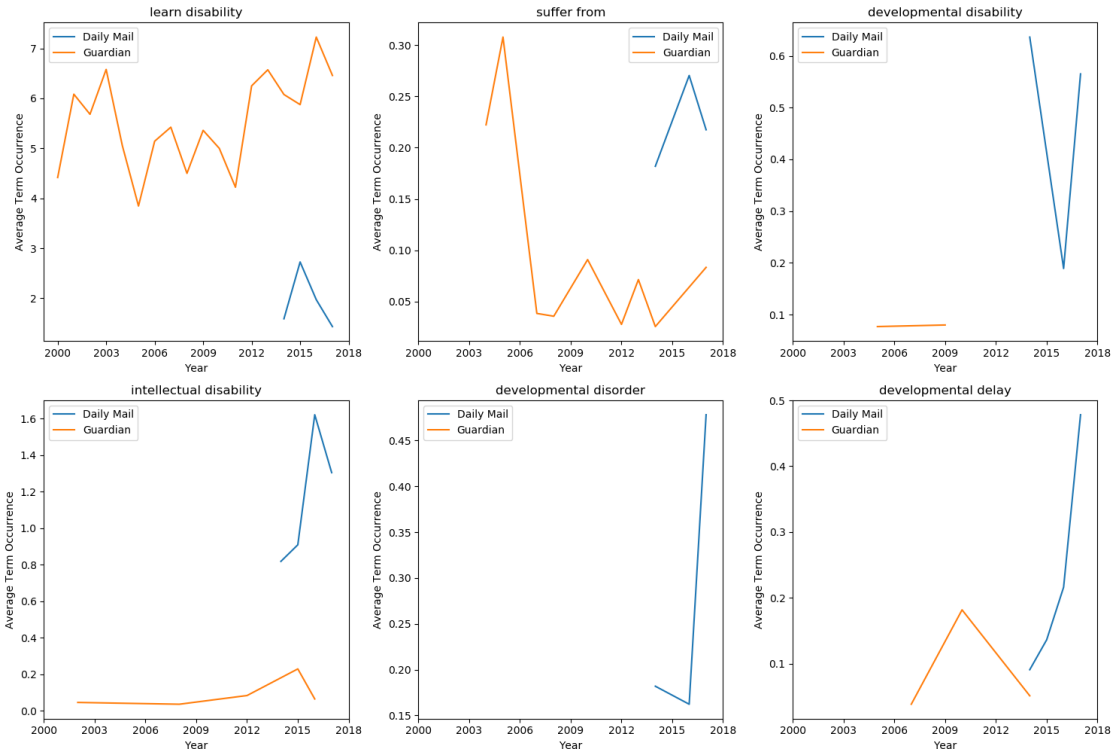


A.6.2 Mann-Whitney U Test Results (p-values)

Topic	Guardian > Daily Mail	Guardian > Daily Express	Daily Express > Daily Mail
All	0.383	0.00179	0.00326 **
2014	0.0695 **	N/A	N/A
2015	0.106 **	N/A	N/A
2016	0.000845	N/A	N/A
2017	0.251	N/A	N/A

** Indicates where the reverse assumption is true (e.g. Daily Mail > Daily Express instead of Daily Express > Daily Mail)

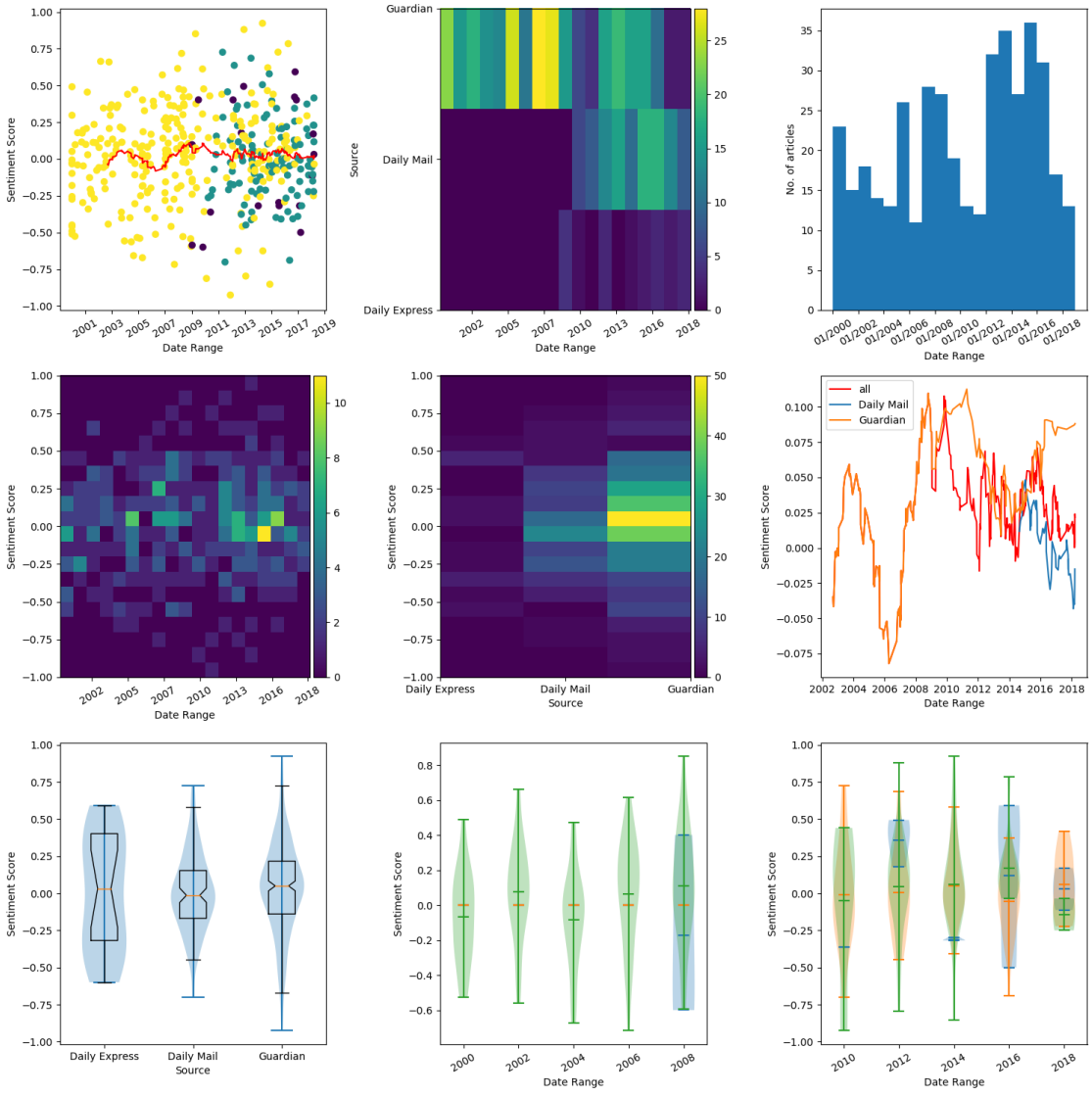
A.6.3 Key Term Trend Plots



A.7 Topic: 'dyslexia'

Key Terms: 'dyslexia', 'dyslexic'
Query Terms: 'dyslexia', 'dyslexic'
Sample size, n = 410

A.7.1 Sentiment Score Plots



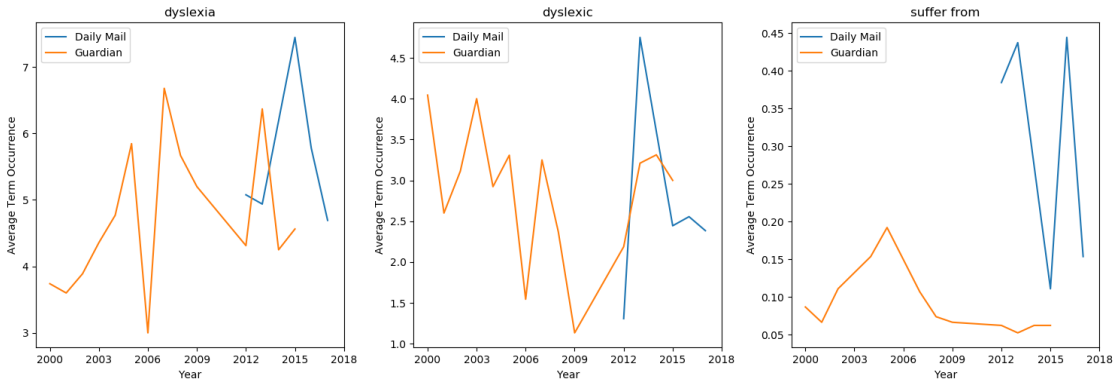
A.7.2 Mann-Whitney U Test Results (p-values)

Topic	Guardian > Daily Mail	Guardian > Daily Express	Daily Express > Daily Mail
All	0.0740	0.362	0.464 **

Insufficient sample size for year-by-year comparisons. (n=410).

** Indicates where the reverse assumption is true (e.g. Daily Mail > Daily Express instead of Daily Express > Daily Mail)

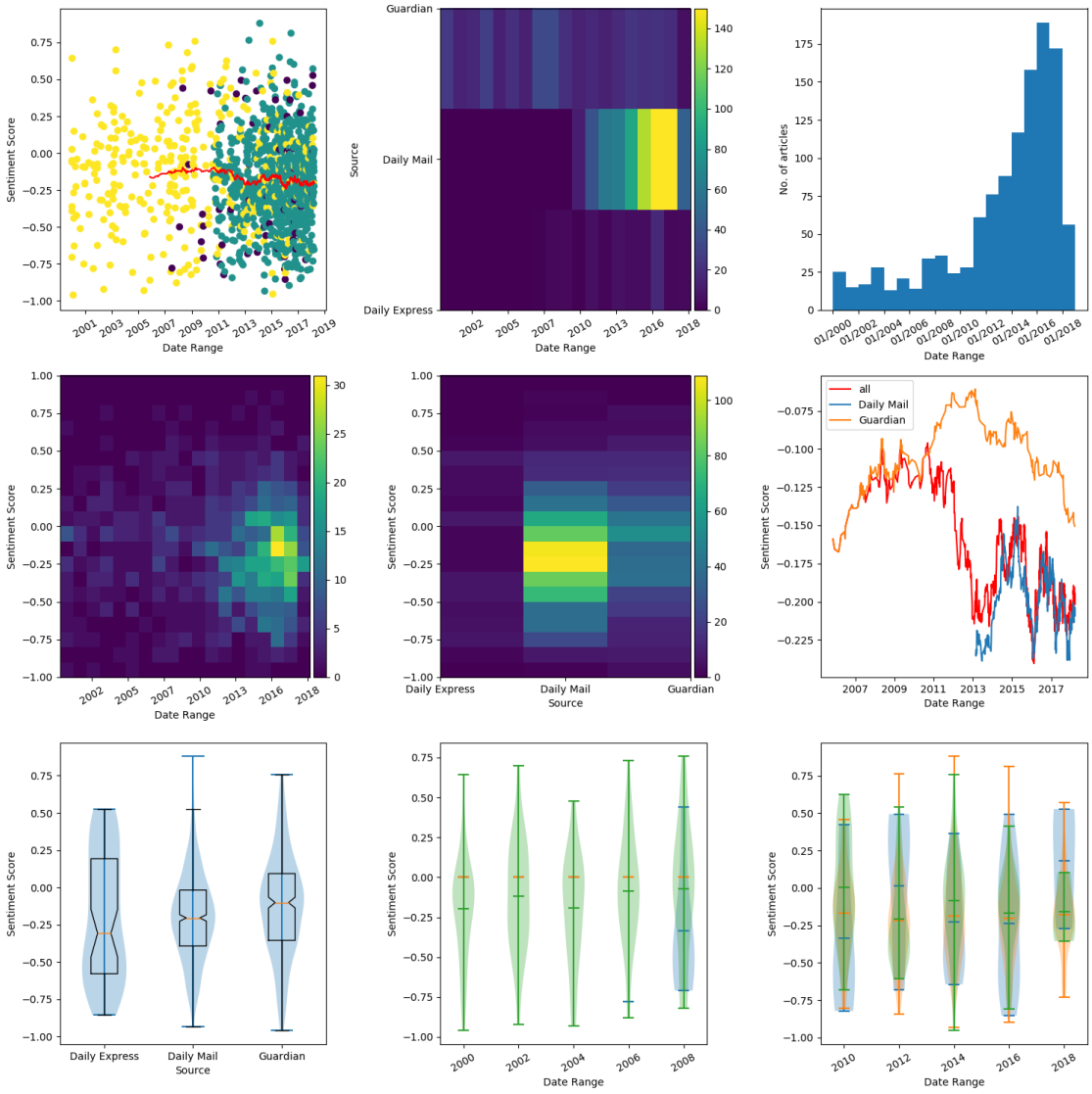
A.7.3 Key Term Trend Plots



A.8 Topic: ‘epilepsy’

Key Terms: ‘epilepsy’, ‘epileptic’, ‘seizure’
Query Terms: ‘epilepsy’, ‘epileptic’
Sample size, n = 1,172

A.8.1 Sentiment Score Plots

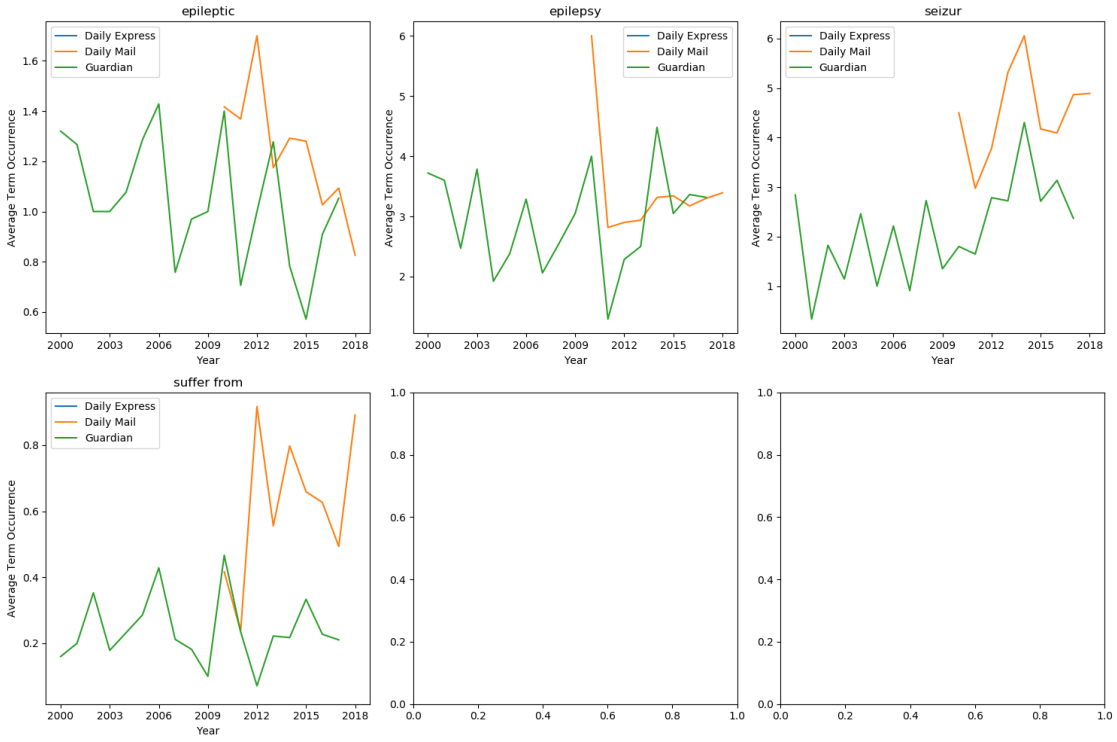


A.8.2 Mann-Whitney U Test Results (p-values)

Topic	Guardian > Daily Mail	Guardian > Daily Express	Daily Express > Daily Mail
All	$3.39 * 10^{-5}$	0.0579	0.322 **
2014	0.0130	N/A	N/A
2015	0.193	N/A	N/A
2016	0.398 **	N/A	N/A

** Indicates where the reverse assumption is true (e.g. Daily Mail > Daily Express instead of Daily Express > Daily Mail)

A.8.3 Key Term Trend Plots



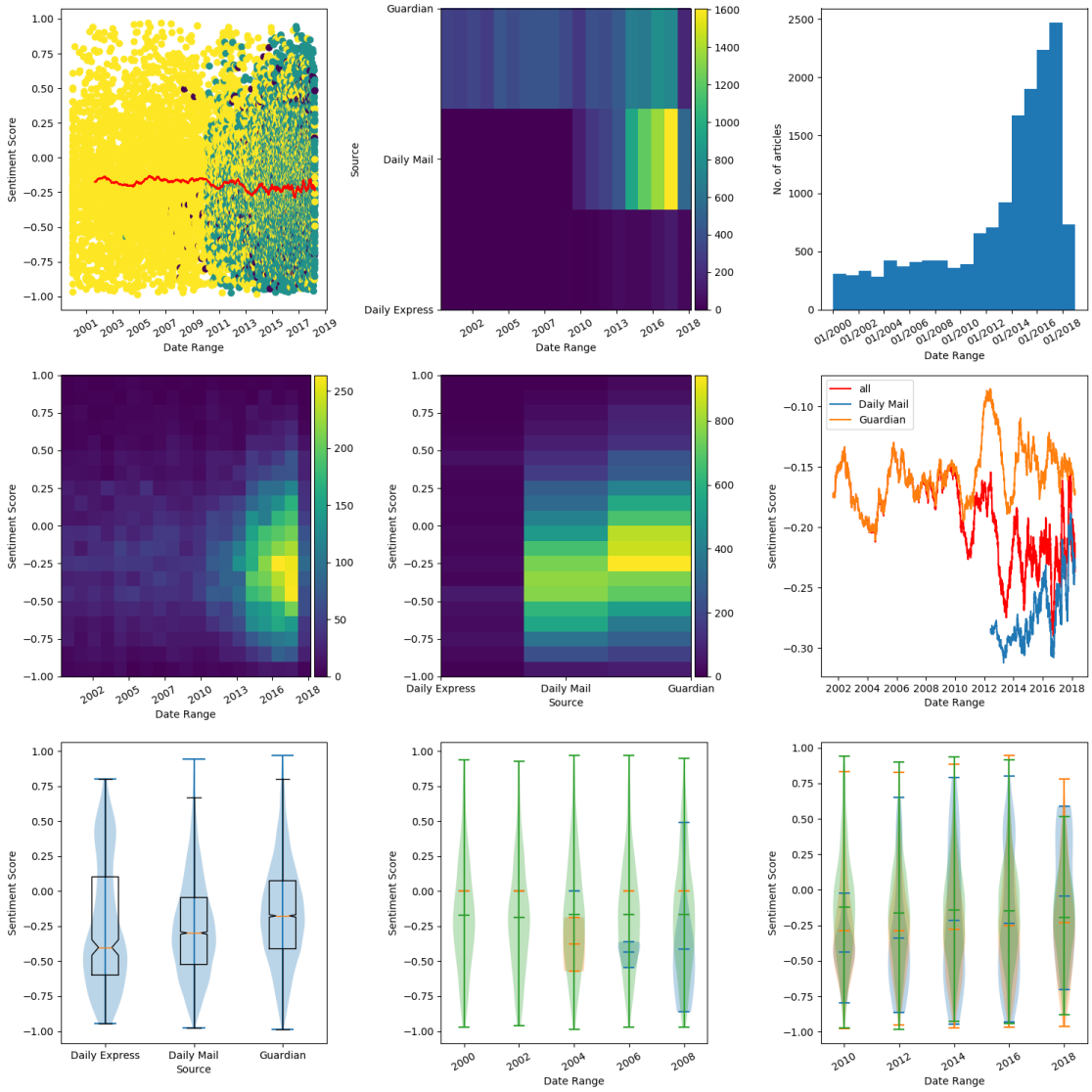
A.9 Topic: ‘mental illness’

Key Terms: ‘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’

Query Terms: ‘mental illness’, ‘mental health’, ‘mental disorder’, ‘mental disability’, ‘mentally ill’, ‘mentally disabled’, ‘mentally handicapped’

Sample size, n = 15,329

A.9.1 Sentiment Score Plots

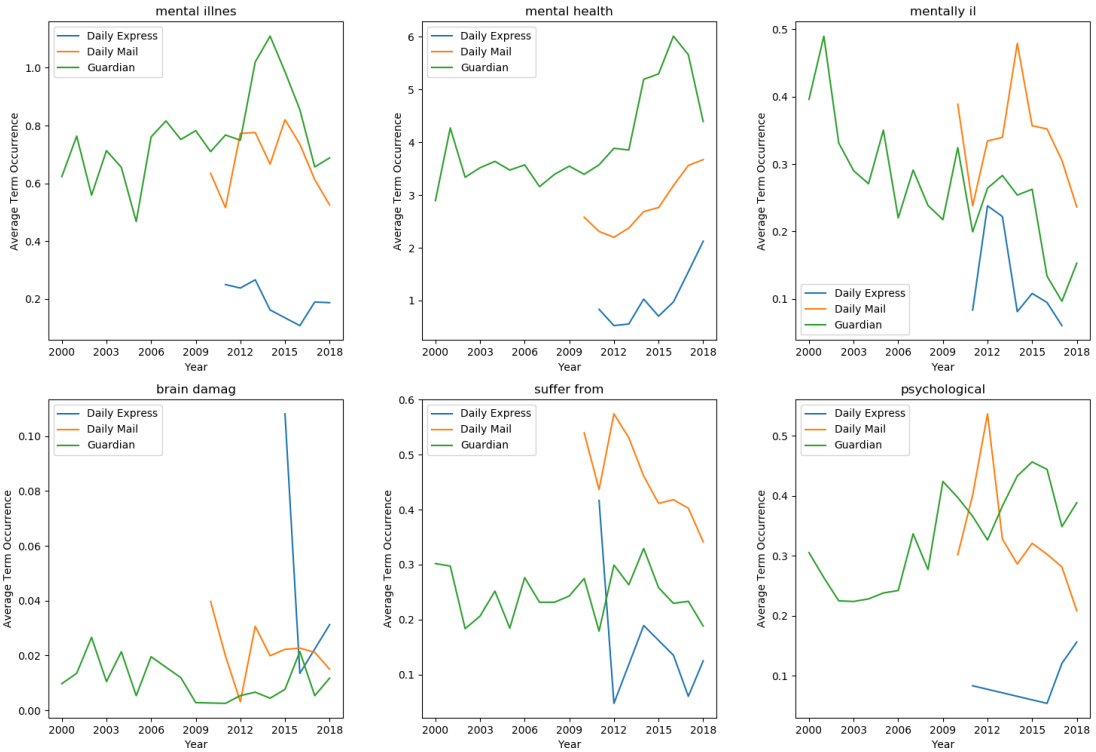


A.9.2 Mann-Whitney U Test Results (p-values)

Topic	Guardian > Daily Mail	Guardian > Daily Express	Daily Express > Daily Mail
All	$3.54 * 10^{-76}$	$1.51 * 10^{-9}$	0.102 **
2010	0.000255	N/A	N/A
2011	$8.79 * 10^{-11}$	N/A	N/A
2012	$5.70 * 10^{-8}$	0.377	0.112
2013	$1.13 * 10^{-7}$	$1.43 * 10^{-7}$	0.000132 **
2014	$1.19 * 10^{-22}$	0.108	0.178
2015	$2.77 * 10^{-10}$	0.0588	0.401 **
2016	$1.83 * 10^{-19}$	$2.56 * 10^{-11}$	$2.03 * 10^{-5}$ **
2017	$5.55 * 10^{-10}$	0.389 **	0.00826
2018*	0.0531	0.0413 **	0.00878

* 2018 data is incomplete and would only include articles up to (approximately) end of March.
** Indicates where the reverse assumption is true (e.g. Daily Mail > Daily Express instead of Daily Express > Daily Mail)

A.9.3 Key Term Trend Plots

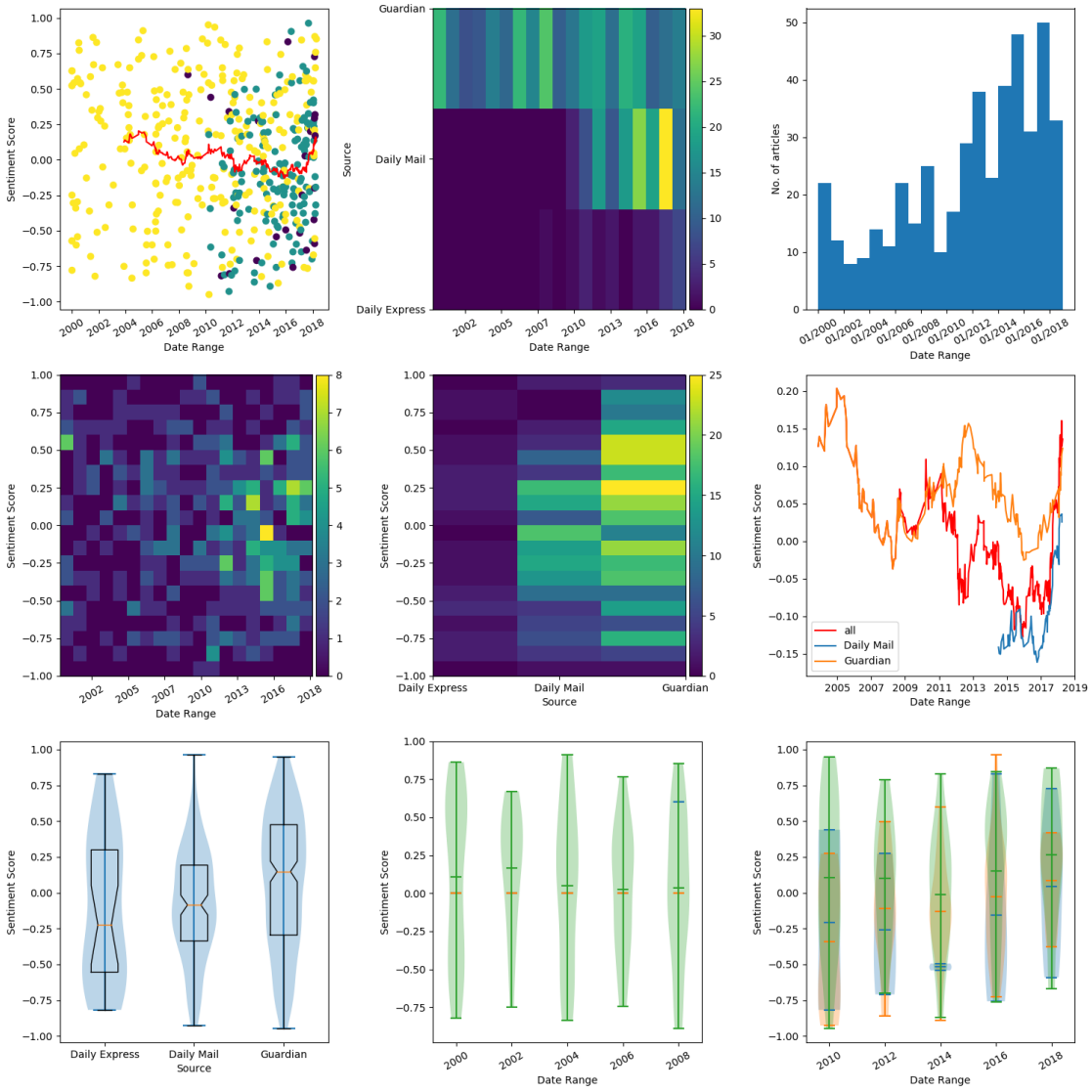



psychiatric
mental disability
mentally disabl
mental disorder
brain injury
retardation
mentally handicap
mental issu
intellectual disability
behavioural disorder
emotional disorder
brain injur
Daily Express
Daily Mail
Guardian
Average Term Occurrence
Year

A.10 Topic: ‘mute’

Key Terms: ‘mute’, ‘muteness’, ‘mutism’, ‘cannot speak’, ‘difficulty speaking’, ‘synthetic speech’, ‘non-vocal’, ‘non-verbal’
Query Terms: ‘mute’, ‘muteness’, ‘mutism’, ‘non-verbal’
Sample size, n = 456

A.10.1 Sentiment Score Plots



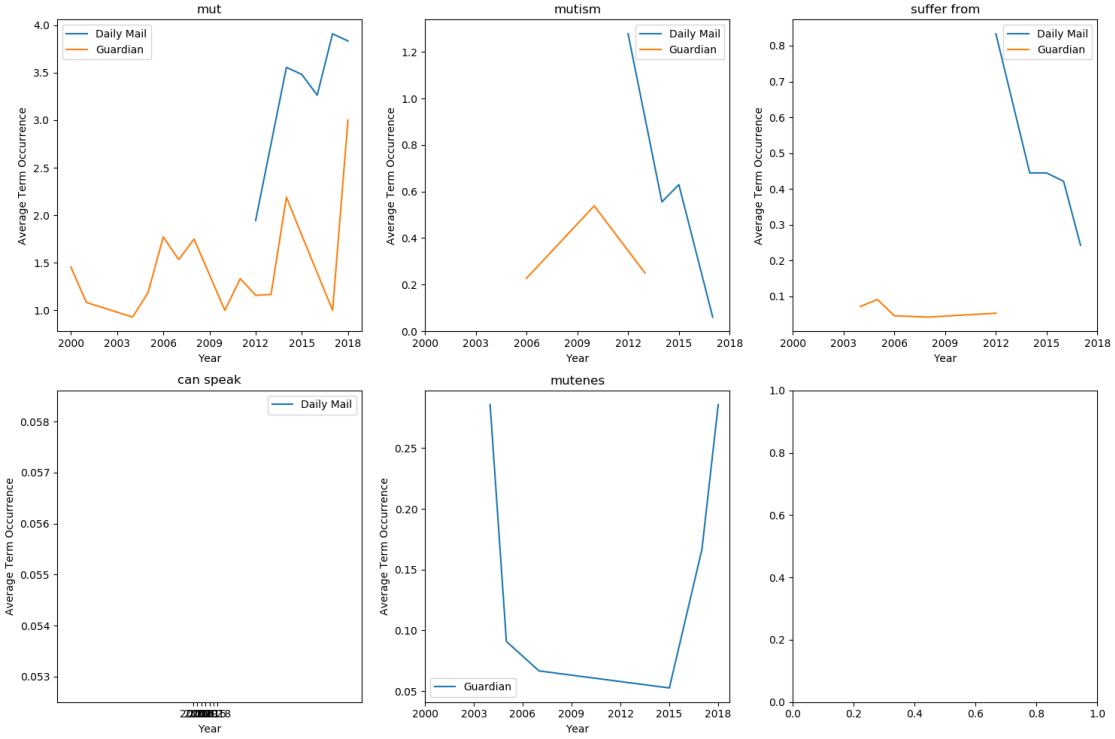
A.10.2 Mann-Whitney U Test Results (p-values)

Topic	Guardian > Daily Mail	Guardian > Daily Express	Daily Express > Daily Mail
All	$6.78 * 10^{-5}$	0.0311	0.346 **

Insufficient sample size for year-by-year comparisons. (n=456).

** Indicates where the reverse assumption is true (e.g. Daily Mail > Daily Express instead of Daily Express > Daily Mail)

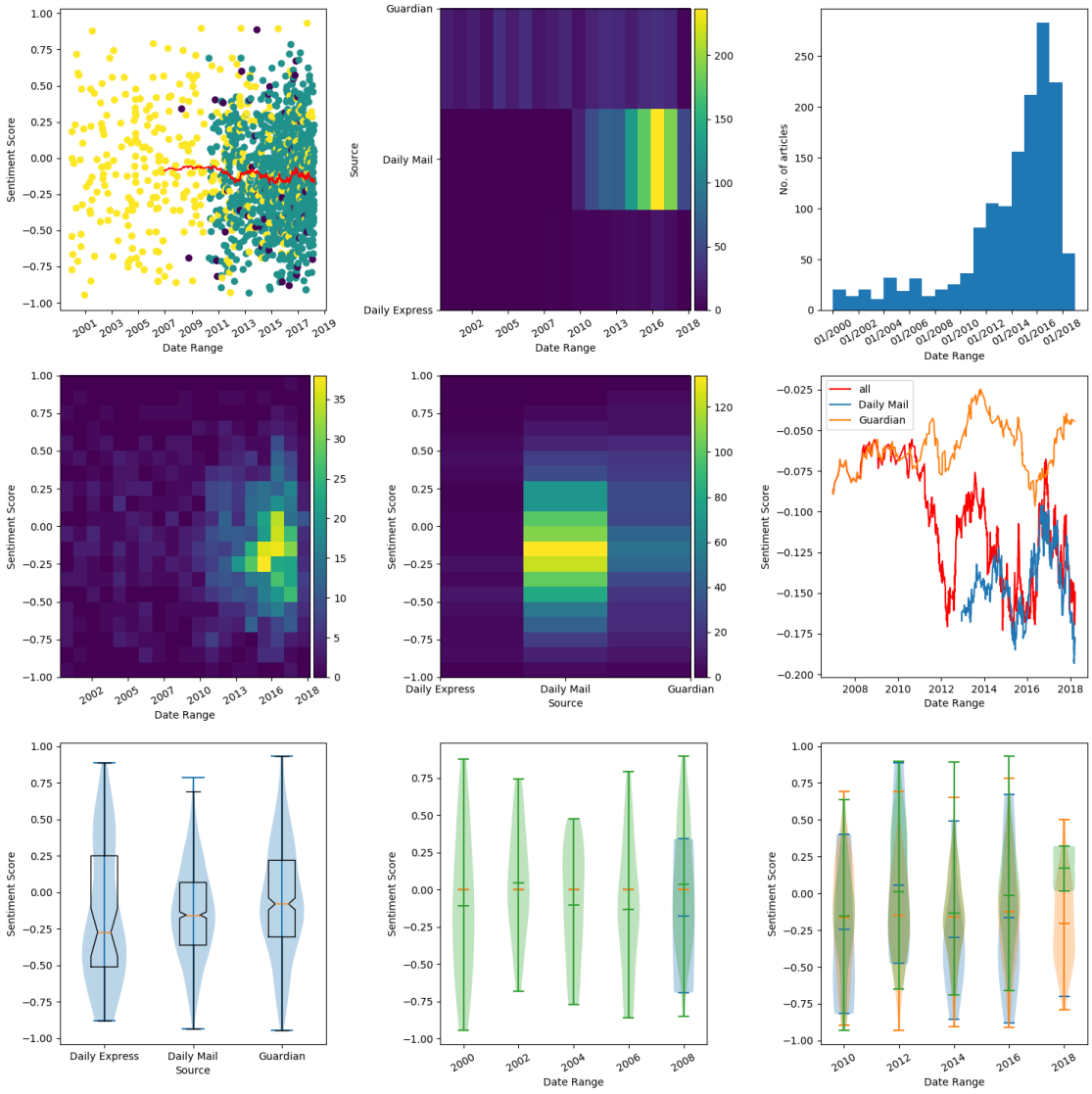
A.10.3 Key Term Trend Plots



A.11 Topic: ‘paralysis’

Key Terms: ‘paraplegic’, ‘quadriplegic’, ‘spinal cord’, ‘paraplegia’, ‘quadriplegia’, ‘paralysed’, ‘paralyzed’, ‘paralysis’, ‘crippled’, ‘leg braces’, ‘wheelchair’
Query Terms: ‘paraplegic’, ‘quadriplegic’, ‘paraplegia’, ‘quadriplegia’, ‘paralysis’
Sample size, n = 1,461

A.11.1 Sentiment Score Plots



A.11.2 Mann-Whitney U Test Results (p-values)

Topic	Guardian > Daily Mail	Guardian > Daily Express	Daily Express > Daily Mail
All	$2.73 * 10^{-5}$	0.00909	0.0827 **
2011	0.352 **	N/A	N/A
2012	0.00184	N/A	N/A
2013	0.374	N/A	N/A
2014	N/A	N/A	N/A
2015	0.331	N/A	N/A
2016	0.149	N/A	N/A
2017	0.00431	N/A	N/A

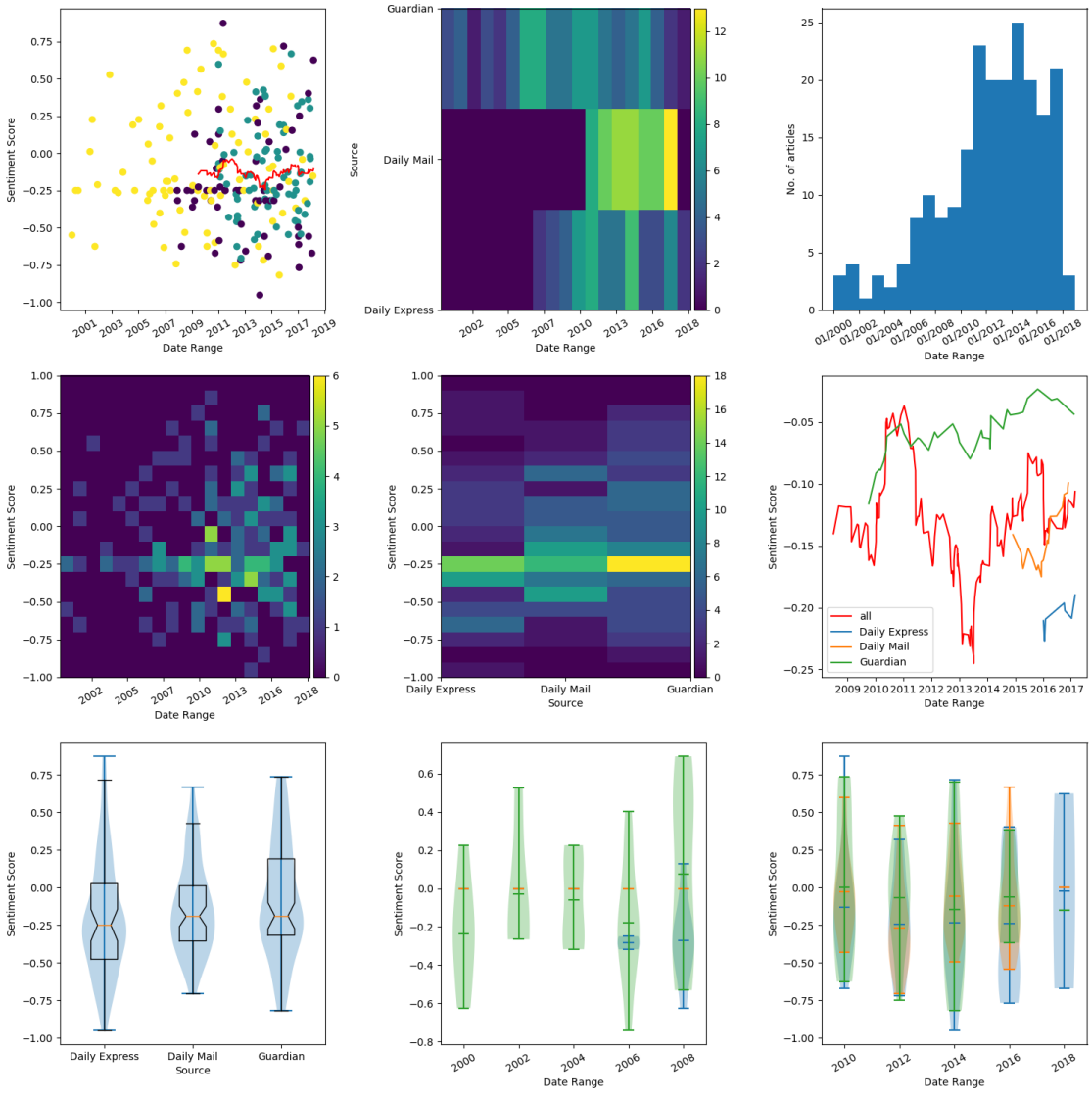
** Indicates where the reverse assumption is true (e.g. Daily Mail > Daily Express instead of Daily Express > Daily Mail)

A.11.3 Key Term Trend Plots

A.12 Topic: 'speech impairment'

Key Terms: 'speech impairment', 'stutter', 'speech disability', 'speech disorder', 'communication disability', 'difficulty speaking', 'language impairment', 'language disorder', 'language disability', 'speech impediment', 'stammer'
Query Terms: 'speech impairment', 'stutter', 'speech disorder', 'speech impediment'
Sample size, n = 215

A.12.1 Sentiment Score Plots



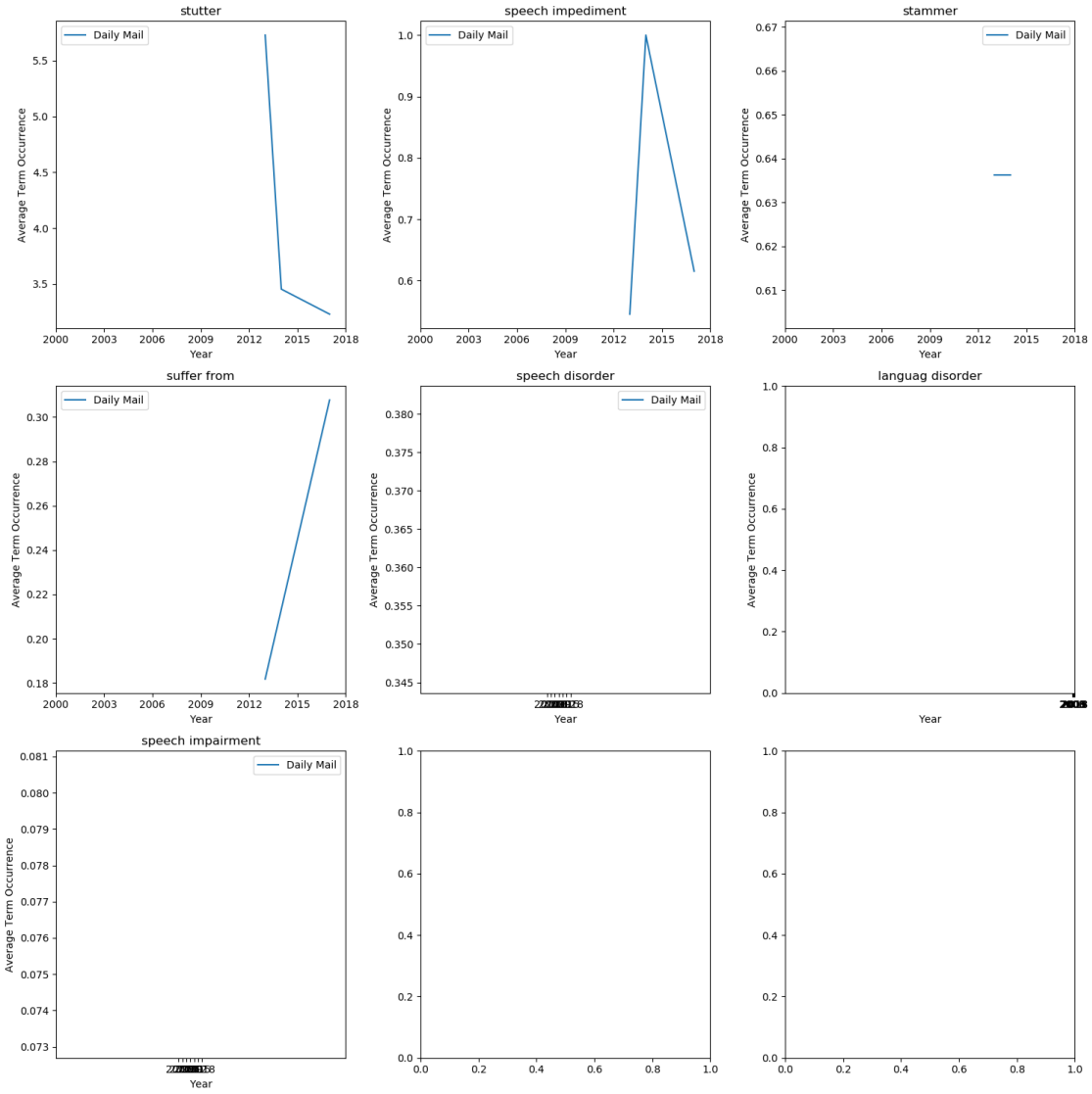
A.12.2 Mann-Whitney U Test Results (p-values)

Topic	Guardian > Daily Mail	Guardian > Daily Express	Daily Express > Daily Mail
All	0.235	0.0200	0.0546 **

Insufficient sample size for year-by-year comparisons. (n=215).

** Indicates where the reverse assumption is true (e.g. Daily Mail > Daily Express instead of Daily Express > Daily Mail)

A.12.3 Key Term Trend Plots



Appendix B

System Manual

The source code that was developed to conduct the experiment and produce the results described in this report are available at `https://github.com/bmaulana/nlp-media`.

To run the pipeline used for this experiment, follow these instructions:

B.1 Set up conda environment

First, download and install Anaconda (`https://www.anaconda.com/download/`)

Then, run the following commands in your command line:

- 'conda update conda'
- 'conda create -n <env-name> anaconda'
- 'activate <env-name>'
- 'conda install -n <env-name> spacy'
- 'python -m spacy download en_core_web_lg'
- 'pip install -U vaderSentiment'
- 'pip install -U tqdm'

B.2 Activate conda environment

Run 'activate <env-name>' in your command line, e.g 'activate nlp-media'.

B.3 Run pipeline script

In your command line, navigate to the root directory of the 'nlp-media' repository, then run 'python pipeline.py' while within the conda environment.

Appendix C

Project Plan and Interim Report