School of Computing Science

Honours Individual Project Dissertation

A Digital Multilingual News Surveillance and Classification System

Adam Fairlie

March 24, 2023

Abstract

Current digital news surveillance systems for disease monitoring are lacking in some areas. They lack diversity in their data collection systems, only retrieving data from a selection of popular news sources and under-representing low-resource areas. They also require results to be translated into English before processing, using a potentially biased machine translation process. We redesign an existing system, BioCaster, to create an improved digital news surveillance system. This system collects news from a wider range of sources across 10 different languages. It produces real-time multi-component visualisations from this data, and has a web interface where the data collection system can be managed. We find that the visualisations are easy to understand and represent collected data well. We also find the web interface to be intuitive, clearly presented and mostly aesthetically appealing.

Secondly, we explore various machine learning techniques and strategies in an experiment on multilingual news category classification. In particular, we find that machine translation as a data augmentation technique can yield increased performance, and that classifiers can apply some learning from public datasets to multilingual self-collected data. Finally, we train a multilingual classifier which achieves 89.4% accuracy on real-world collected multilingual data.

Education Use Consent

I hereby grant my permission for this project to be stored, distributed and shown to other University of Glasgow students and staff for educational purposes. **Please note that you are under no obligation to sign this declaration, but doing so would help future students.**

Signature: Adam Fairlie Date: 24 March 2023

Contents

1 Introduction

1.1 Motivation

Diseases can transmit at a very high rate, travelling rapidly between areas before they can be initially detected and contained. It is important to avoid as much harm as possible by having up-to-date information on local disease outbreak risks available to governments and members of the public. Traditional methods of disease surveillance include laboratory networks such as PulseNet from the Center for Disease Control (CDC) which look for patterns in bacteria across infected patients which may indicate an outbreak. These methods can be slow, relying on human experts and conducting experiments which can take several days until results are returned (Swaminathan et al. 2001). These methods are often also lacking in coverage of certain diseases or geographic regions due to factors such as a lack of specialist knowledge (Abdulraheem et al. 2004). Digital forms of surveillance such as online media monitoring have been effective at discovering disease outbreaks at the early stages and mitigating spread, including during the recent COVID-19 pandemic (Kostkova et al. 2021). Several digital surveillance systems have been developed to collect real-time information from online sources, such as news articles, which can be used to quickly find and assess the risk of disease outbreaks across the world. This information can allow governments to respond rapidly to any new outbreaks by providing quick risk information to the public and planning for diagnosis, testing and prevention. It is for this reason that many governments are creating systems which incorporate online sources to monitor disease outbreaks, such as the Global Public Health Intelligence Network (GPHIN)[1] from the Canadian government.

One such digital surveillance system is BioCaster[2], developed by a research team from the University of Cambridge and McGill University, which collects news data and provides an online visualisation of disease alerts on their website. There are flaws with the current BioCaster system, particularly in the news collection system. The current system relies on published news feeds, typically provided by large established sources in high-resource countries, creating a coverage gap in low-resource areas and minority languages. The lack of diversity in the types of sources collected may also impact countries which publish less traditional news, and restricts types of online data such as search trends and social media posts which research has shown to be effective in detecting disease outbreaks early (Seo and Shin 2017).

Another weakness in the current BioCaster system is the reliance on the accuracy of external machine translation systems to translate all documents to English before disease risk can be calculated. These systems can struggle with issues such as domain mismatch (words having different translations depending on the domain) (Koehn and Knowles 2017) and can encode forms of bias such as gender bias (Stanovsky et al. 2019). Multilingual models which eliminate the need for machine translation exist for tasks such as multiclass classification, and have not been tested within the BioCaster system. In this project, we will develop a new digital surveillance system which collects and displays a wider range of news data, and perform experiments to test the capabilities of multilingual news classification.

[1] https://gphin.canada.ca/cepr/aboutgphin-rmispenbref.jsp?language=en_CA
[2] http://biocaster.org/

1.2 Project Aims

The aims of the two contributions of this project (the new digital news surveillance system and the experiment on multilingual news classification techniques) are detailed below:

1.2.1 Digital News Surveillance System

News collection system: We will develop a robust and extensible data collection system which can synthesise data from multiple different sources and source types, allowing for a more extensive selection of online news data to be collected including lower-resource countries and minority languages. We will implement two distinct methods of data collection: It will redevelop an improved version of the current BioCaster data collection system based on RSS[3] feeds so that the current system can integrate into the new system without loss of data sources, and a web scraping system will also be developed to extract important information such as article headlines, main bodies and publish dates from a given webpage.

Data storage and visualisation: The collected news data will be stored in structured form in a database. The information will then be used to produce real-time visualisations of collected data, aggregated into useful statistics, graphs and charts, so that data collected can be observed and analysed. These visualisations should include information on the source countries and languages of collected data, the methods of collection used to retrieve the data and the category distributions of news articles. The system should frequently update and be easy to understand.

Web interface: We will develop a web interface so that the data collection system can be easily managed, including enabling/disabling the collection system, enabling/disabling data sources and adding and removing sources. The system should display the last time sources have collected new data, allowing for the highlighting of stale sources which can be removed to save resources. The interface should be responsive and easy to use and understand.

1.2.2 Multilingual News Article Classification

We will also conduct research on multilingual machine learning models through the task of multilingual news article category classification, training on public news article datasets and collected real-world news data, and evaluating the accuracy of multilingual models and the effectiveness of machine translation techniques on upsampling datasets. In particular, We will research the effectiveness of machine translation as an upsampling technique, the effectiveness of models trained on public data when translated to real-world data and the current effectiveness of multilingual models on real-world news data. The final multilingual classifier will be integrated into the scraping system, assigning each incoming article a category.

[3] Really Simple Syndication

2 Background

2.1 Digital News Surveillance Systems

Several digital news surveillance systems exist for different purposes, including research systems for public health monitoring and commercial systems designed for businesses to track information related to their brand. HealthMap[1] is a public health monitoring system which provides a world map with location-based disease markers for each country, using RSS data and sources gathered by health experts. EpiCore (Haddad et al. 2016) is a similar system, however it instead uses an online network of trained experts to share information digitally. MediSys (Steinberger et al. 2008) is a public health monitoring system which takes information from many web portals and commercial news sites to sort incoming news into topics which can be viewed on the website.

2.1.1 BioCaster

In this project, we will rework and extend some of the functionality currently used in BioCaster. The BioCaster system collects news articles through a variety of RSS feeds across different languages including English, French and Mandarin (Collier et al. 2008). It uses a machine translation server to translate the headlines and article descriptions into English, and various machine learning and rule-based models to filter the English-language data to remove all irrelevant (not disease-related) articles and collect specific details which are used to create a risk alert. These alerts are displayed in a visualisation on the website. A full diagram of the BioCaster system architecture is shown in Figure 2.1.

We aim to rework the first part of this architecture, improving the quality of news sources and removing the machine translation steps. The current BioCaster data collection method is dated: it has not been updated in many years and only collects data through RSS feeds, by running a Perl script once per hour to receive updates (Collier et al. 2008). This requires a news source to publish and maintain a feed for its data to be used in the system. These feeds are provided by a limited number of news sources, which are typically larger and written in high-resource languages and countries, where data is not as underrepresented. This greatly limits the amount of data which can be extracted, particularly from smaller sources in underrepresented countries and minority languages, and as RSS technology is becoming more outdated and falling out of use, the data available for use in surveillance is continuing to diminish in the current system. The data collected from RSS feeds is also limited to a headline and in some cases a short or medium-length description of article content. This provides limited information to models in later stages of the process for performing relevance classification and risk alert creation tasks.

The current BioCaster system also heavily relies on external domain machine translation. These models can suffer when trained on data outside of their domain (Koehn and Knowles 2017), possibly resulting in inaccurate translations as we cannot expect all news collected to be strictly biomedical. The reliance on external resources can also impact the availability of the service, removing some of the control from the BioCaster team.

[1] https://www.healthmap.org/en/

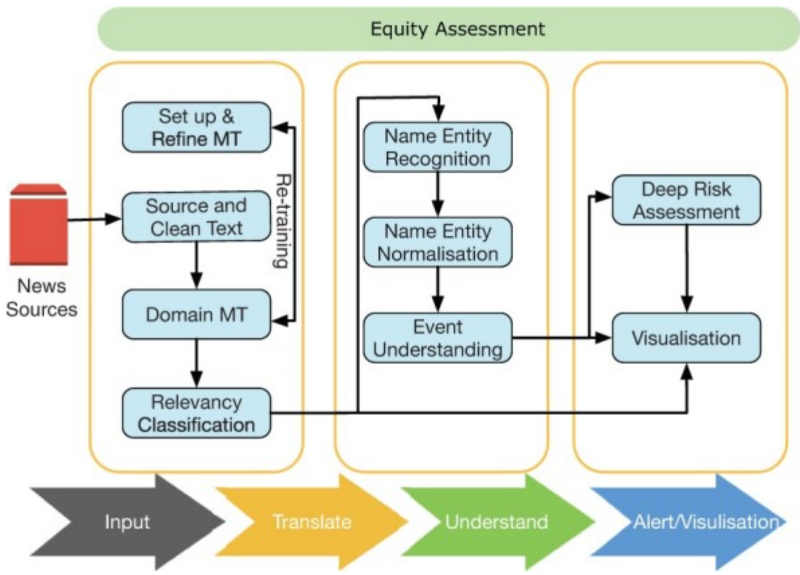

Figure 2.1: *The full architecture of the current BioCaster app. Image obtained from Meng et al. (2022).*

In this project, we will update the BioCaster RSS collection system, as well as add a module for using web scraping to collect news articles from webpage HTML. The system will be designed so that it is easily extensible to other sources of data which may be more prominent in different countries than traditional news, such as social media posts. This also allows for entire article text to be collected instead of just the headline and a short description, which may allow for a richer understanding of the news content and more effective performance in ML tasks. It will also experiment with multilingual news article classification techniques to remove the problems associated with relying on machine translation servers.

2.2 Multilingual News Article Classification

2.2.1 Traditional Models

Historically, many different approaches and models have been used for the task of news article category classification. The most popular model types before the last 5 years are Support Vector Machines (SVM's) and Naive Bayes classifiers. SVM's are models which aim to split the input space into different categories with as large a margin as possible, by finding the best possible hyperplanes in high-dimensional space which split the data into their correct classes. SVM models can achieve non-linearity by transforming inputs through a kernel, a function which transforms inputs into a space which can be linearly separated by the hyperplanes. SVM Models have been repeatedly demonstrated to perform with high accuracy in the news classification literature across different languages, achieving 90+% accuracy on categorisation of news headlines from Sri Lanka (Dilrukshi et al. 2013) and 85% accuracy on news documents from Indonesian news source Kompas (Liliana et al. 2011).

Naive Bayes classifiers use input features, which are (usually incorrectly) assumed to be independent for simplicity. They use these features to update a prior belief about how class data is distributed until the distribution reflects the distribution of real data as accurately as possible. Naive Bayes models are probabilistic models, meaning instead of assigning a single class to an output, they create a probability distribution over all possible classes which can be sampled from for a particular input. A common strategy for Naive Bayes classifiers is to simply take the class with the highest probability for a given input (maximum a posteriori decision rule). The assumption of independence in features can be problematic for classifying news documents, which have heavily codependent features, but methods have been developed to mitigate this issue (Qiang 2010). Naive Bayes models have produced impressive results in the literature, achieving 94% accuracy on an Indonesian news dataset (Septian et al. 2017) and 87% accuracy on English-language Indian news (Kumar et al. 2022). SVM models typically achieve higher accuracy than Naive Bayes models because of the weakness of Naive Bayes inappropriate assumption of independence (Dilrukshi and De Zoysa 2013; Shahi and Pant 2018; Fanny et al. 2018), but Naive Bayes models have the advantage of initialising and classifying data faster than SVM models due to their simplicity, and can sometimes provide comparable or better results (Ting et al. 2011). Other traditional models used in news classification include random forests, which create decision trees in order to discriminate between categories (Liparas et al. 2014) and K-NN classifiers, which takes the K most similar documents and assigns the article the most common class of these documents (Chen et al. 2020).

2.2.2 Transformer-based Models

While these traditional models achieved very high performance on many monolingual datasets, there was often a significant decrease in more complex tasks such as classifying multilingual data (Vogel and Meghana 2020). In 2017, the Google research team released the transformer model architecture, which revolutionised text classification by allowing the context of words within a sentence to be leveraged in the model calculations (Vaswani et al. 2017). A further development was BERT[2], a bidirectional sequence-to-sequence model which removes the decoder section in order to learn word context embeddings (Devlin et al. 2018). BERT very quickly outperformed many established baselines and became the state-of-the-art, and is very widely used in news classification in the last 5 years (Chen et al. 2022; Mujahid et al. 2021; Deping et al. 2021). A detailed explanation of the transformer and BERT architectures can be found in section 5.2.1. These models are widely used because they are pre-trained, and only require fine tuning to a specific task, massively saving computation time and allowing for public access to very complex models. This allows the task of multilingual news classification to be attempted more effectively.

BERT-based models exist which are trained on a corpus of multilingual data, and can embed and represent multilingual input data for classification tasks. One of the most popular multilingual models is mBERT, or multilingual BERT, a variant of BERT trained on over 100 languages. Research by Pires et al. (2019) suggests mBERT may learn an effective language-agnostic latent space which yields impressive results when the model is evaluated on different languages than it is trained on, but this effect decreases as the languages become less structurally similar. mBERT has been used to high success in multilingual news classification tasks (Kakwani et al. 2020; Hutama and Suhartono 2022).

Another model is XLM-RoBERTa, developed by the Facebook AI team (Conneau et al. 2019). It has been shown to outperform mBERT in many multilingual tasks such as named entity recognition (NER) and relation extraction (Li et al. 2021; Lan et al. 2020), as well as in news classification (Alam et al. 2020). These two models will be considered in our evaluation, as they represent the current state-of-the-art in multilingual news classification.

[2] Bidirectional Encoder Representations from Transformers

3 System Design

3.1 Design Principles

We designed the system architecture with consideration towards a few key software design principles and concepts. The main prnciples considered were:

- **Separation of concerns:** Each individual component of the system is given its own module which encapsulates its main functionality and interaction with other modules. This also means that components could possibly be re-used in future projects.
- **Configurability:** The system is designed to give the user as much choice as possible. The system can read a config file which allows the user to control many aspects of the system, such as whether a local or cloud database is used and the maximum number of active sources which are loaded. In addition, individual components are designed so that they can easily be switched out if new components are developed.
- **Extensibility:** Modules of the system which can have implementations changed or new versions created are designed so that this process is as easy as possible, abstracting the consistent functionality of the module into a superclass that new components will inherit from. As previously discussed, social media data has been shown to be a powerful indicator of disease outbreaks, so this system could be extended by adding a social media crawler, for example.

3.2 System Overview

Figure 3.1 shows a diagram of the overall system architecture.

3.2.1 Modules

The overall system is designed so that modules can be swapped out and replaced or added into the system in order to cover different sources of online data. In this project, the system and its modules will be designed to gather news articles from RSS feeds and news websites. The purpose of each module in the overall system is as follows:

- **Crawler:** Connect to an internet source (in this case, an RSS feed or website) and obtain a list of URL's to be parsed for the system (in this case URLs of news articles).
- **Article parser:** Parse the given list of URLs, mining the page source content for details (in this case, Headline, publish date etc.).
- **Classifier:** Pass each article through a machine learning model in order to assign it a category (in this case, each news article is given a topic).
- **Database connector:** Provide an interface to allow the system to access the external database.
- **Database:** Permanently store the collected and classified data, and the crawlers, in a structured form.

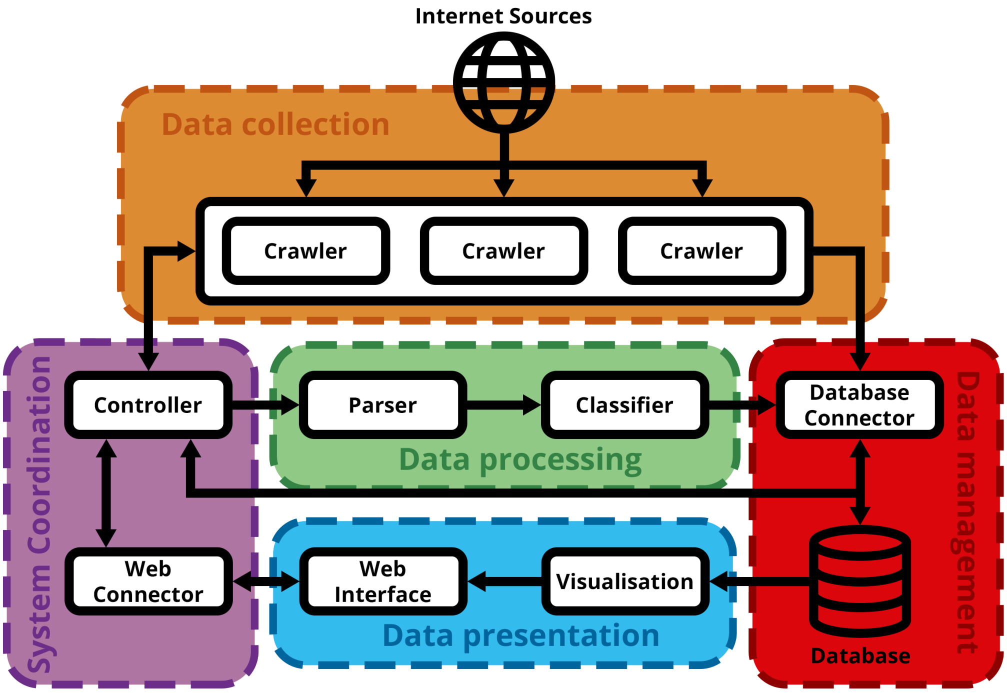

Figure 3.1: The overview of the system architecture for this project, showing the different modules and interactions between them. Each module is contained in a box. Arrows indicate the transfer of data between modules (e.g. A -> B means module B receives data from module A). The larger, coloured boxes indicate different subsystems.

- **Visualisation:** Present the collected and aggregated data in a number of graphs, charts and statistics for monitoring and interpretation.
- **Web interface:** Allow administrators to see the visualisation and to manage the active crawlers.
- **Web connector:** Provide an interface for communication between the management website and the scraping system.
- **Controller:** Initialise crawlers from the sources stored in the database. Send the parsed articles to the classifier and communicate with the web interface (send updates and receive instructions).

3.3 Data Collection

The data collection subsystem crawls an internet source for all URLs which can possibly contain useful information. In this project, we are concerned with finding the URLs of news articles from a news website or RSS feed. To be appropriate in a disease surveillance system, the data should be updated in near real-time so outbreaks can be quickly understood and responded to.

3.3.1 Crawler

The crawler is the first stage of the data collection process. It represents one individual data source collected in the system, either an RSS feed or a website which is crawled for articles. Crawler objects are stored as a row in a data table of sources in the database. The objects are instantiated by the controller from information given by the web interface when a new source is added, or from an existing source in the database. The responsibilities of a crawler are:

- Instantiate itself from information stored in the database or given by the web interface.
- Add itself to the database if it did not already exist.
- Establish a connection with an internet source, in order to receive information.
- Create a list of potential sources of data to be parsed (a list of article URLs).
- Update its state when it has received new data or instructions.
- Update the database with its new state (e.g. delete itself or update its last scrape time).

An abstract crawler class encapsulates most of the functionality of this module, handling all communication between other modules. Any concrete crawler implementations (in this project, there are 2: RSS feed crawler and Web crawler) only have to create a method to crawl specific types of data from the internet and use it to update the last scraped date, as well as override the constructor to add its source type. A full diagram of crawler functionality is shown in Figure 3.2.

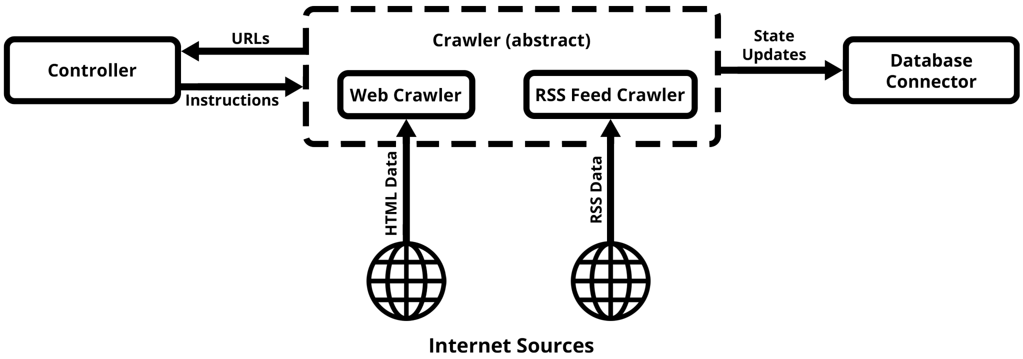

Figure 3.2: *The overview of the inputs and outputs of the crawler system. The diagram shows the concrete Web Crawler and RSS Feed Crawler implementations being encapsulated by the abstract crawler class, which handles interactions with other modules. The concrete classes each collect a different type of data from the internet sources.*

3.4 Data Processing

The data processing subsystem creates structured data from internet sources, converting unclean data such as an RSS feed or the HTML of a webpage into a formatted news article with a headline and article body and associated data. It also attempts to understand the data by assigning a category to the incoming articles. The data processing subsystem should be able to process data at a high volume from sources in 10 different languages: English, French, Spanish, Portuguese, Russian, Indonesian, Mandarin, Korean and Arabic. When data is processed, it will be added to the database so it can be used in visualisation.

3.4.1 Parser

The parser receives as input a combined list of URLs from the crawlers and for each URL, will retrieve the page HTML and use it to extract details about the page (in this case, the pages will be news articles) which can be inserted into the database. The article URLs will be shuffled randomly before parsing to avoid very fast repeated requests to the same host, which smaller sources may not be able to easily handle. The parser assumes that news sources are monolingual to avoid language detection wherever possible, but will attempt to detect the language if parsing is incorrect. After this process is complete, the parser sends the parsed data to the classifier to assign the article a topic. The information required from a news article for this project, which the parser will attempt to scrape, is:

- Article headline (e.g. "UK interest rates: What the rise means for you").
- Article body (e.g. "The Bank of England has raised interest rates...").
- Publish date (e.g. 23rd March 2023 13:00).
- Language (e.g. "EN" for English).

An abstract parser class handles all communication with other modules. Any concrete parser implementations (such as the article parser in this project) only have to implement a specific method for crawling a list of given URLs and retrieving structured data from the URLs and returning a final list. A full diagram of parser functionality is shown in Figure 3.3.

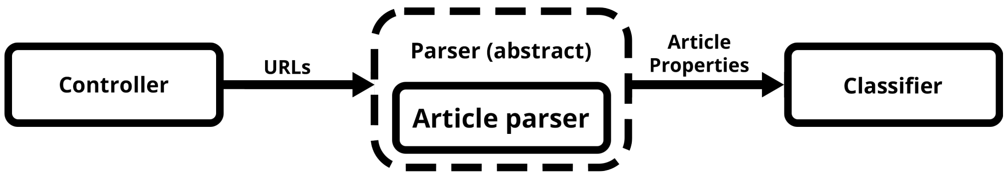

Figure 3.3: *The overview of the inputs and outputs of the parser system. The diagram shows the concrete article parser implementation being encapsulated by the abstract parser class, which handles interactions with the controller and classifier.*

3.4.2 Classifier

The classifier receives a list of parsed articles, with features extracted. It then performs any necessary data preparations and tokenization before feeding the articles through a trained classifier model to assign a category to each news article (e.g. Sports, Entertainment). In this project, the classifier module will use a pre-trained BERT-based classifier model loaded from a file and will tokenize and classify the concatenated article headline and body. The parsed articles with topics will be sent to the database connector and inserted into the database.

An abstract classifier class handles communication with other modules and batching of input data. To create a concrete classifier implementation, we would have to implement:

- A constructor which properly initialises the model and any tokenizers.
- A method which preprocesses a list of input data.
- A method which receives an input batch and outputs a list of corresponding classification labels.

A full diagram of classifier functionality is shown in Figure 3.4.

3.5 Data Management

The data management subsystem stores the data (data sources and collected articles) in a permanent form, where it can be shown in the visualisations. The database system should be able to handle many reads (for article duplicate checking) and writes (for inserting new articles) at a high volume.

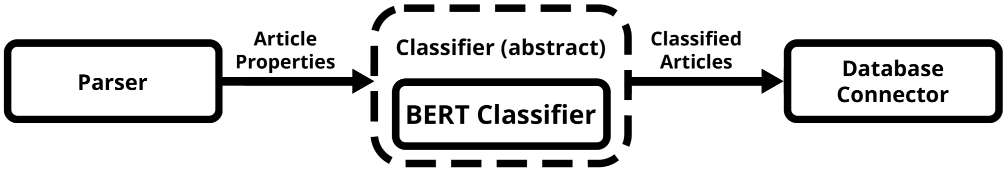

Figure 3.4: The overview of the inputs and outputs of the classifier system. The diagram shows the concrete BERT classifier implementation being encapsulated by the abstract classifier, which handles interactions with the parser and database connector.

3.5.1 Database

For this project, the database will consist of two tables: **Articles** and **Sources**. The attributes for these tables are shown in Table 3.1.

Source		
Attribute	**Type**	**Description**
ID	Primary Key	A unique auto-generated source ID.
URL	Text	The data source URL (e.g. "https://www.bbc.co.uk/news").
Name	Text	A name describing the source (e.g. "BBC News").
Country	Text (2)	The ISO 3166-1 alpha-2[1] code of the country the source originates from (e.g. "GB").
Language	Text (2)	The ISO 639-1[2] code of the sources. main language (e.g. "EN")
Data Source	Text	The type of source this is (e.g. "RSS/Atom feed").
Last retrieved	Date/Time	The time this source last retrieved new data.
Active	Boolean	Whether this source is active (if False, when instantiated the crawler will be disabled).
Article		
Attribute	**Type**	**Description**
ID	Primary Key	A unique auto-generated article ID.
URL	Text	The URL of the article.
Headline	Text	The article headline.
Body	Text	The main body of the article.
Country	Text (2)	The ISO 3166-1 code of the country of the original source.
Language	Text (2)	The ISO 639-1 code of the language of the article.
Published	Date/Time	The time this article was published.
Retrieved	Date/Time	The time this article was retrieved by the scraping system.
Source name	Text	The name of the source this article is collected from.
Source type	Text	The type of source this article came from (e.g. "Web crawler").
Category	Text	The assigned category of this article (e.g. "Entertainment").

Table 3.1: List of attributes in the Source and Article tables, with data type and description. Primary key attributes are highlighted in bold. "Text (2)" denotes that this field is a text field which is exactly 2 characters long.

[1] https://en.wikipedia.org/wiki/ISO_3166-1_alpha-2
[2] https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes

3.5.2 Database Connector

The database connector provides a connection between the system and the external database, allowing the other modules to send and retrieve the data. In this system, the connector must have functionality for:

- Creating the Article and Source tables.
- Adding new sources and articles.
- Retrieving stored sources.
- Deleting sources.
- Updating sources (enabling/disabling, updating last scrape time).
- Searching articles by URL and headline (for duplicate checking).

A full diagram of database functionality is shown in Figure 3.5.

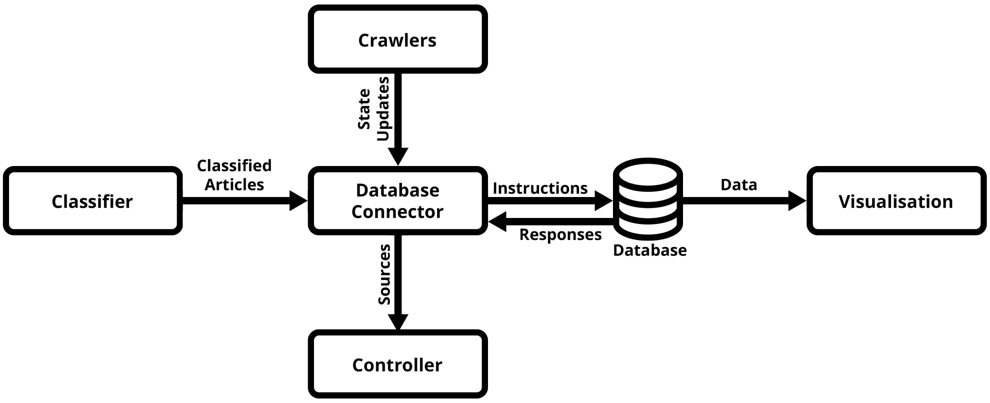

Figure 3.5: *The overview of the inputs and outputs of the database and connector.*

3.6 Data Presentation

The data presentation subsystem is concerned with creating an easy interface for a user to view, analyse and understand the data collected, as well as easily interact with the system to modify its behaviour. The system should have interfaces which are easy to use and understand and provide useful insight and control on the data collection process.

3.6.1 Visualisation

The visualisation is mainly based on the current BioCaster interface, as shown in Figure 3.6. This design has been updated to reflect the changes from my project to the current system, including the addition of news article topics and different source types of information retrieved. We also did not have the specific region or disease information to create the alerts seen on the BioCaster interface. Shown in Appendix A is an initial wireframe of the new visualisation.

After supervisor feedback, more emphasis was placed on filtering by source type. The final visualisation will have the following modules:

- A bar for filtering by category, language and source type.
- A section showing the number of total and active sources.
- A section showing the number of articles scraped today, this week and this month.

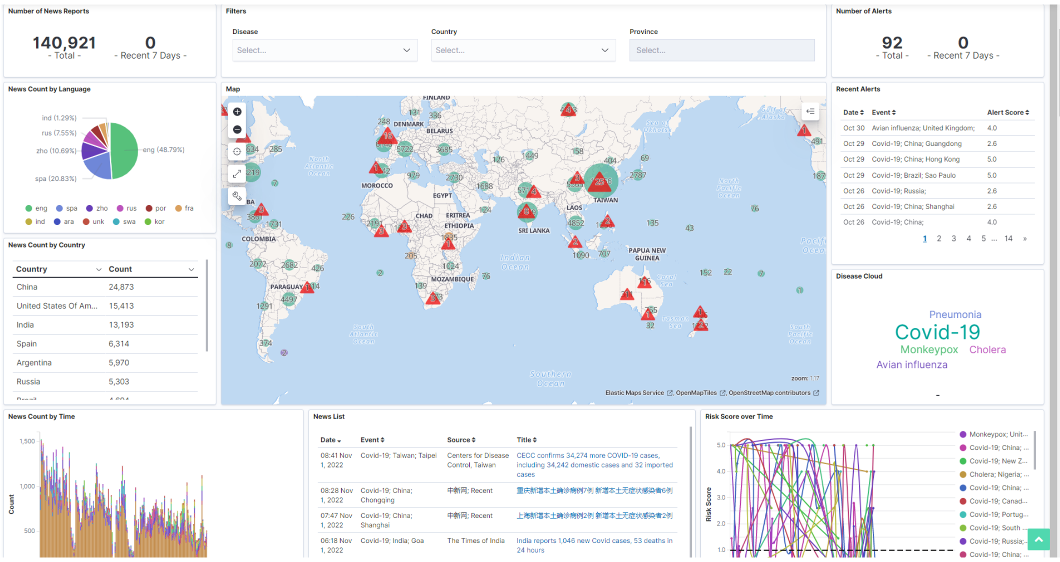

Figure 3.6: The current BioCaster visualisation. Available at http://biocaster.org.

- Pie charts showing data distribution by source type, language and category.
- A table showing the most recent articles collected.
- Time-series data on articles collected over time.
- A world map showing information per country (number of articles, most common category etc.).

3.6.2 Web Interface

The web interface is designed to show the visualisation on a webpage, as well as provide functionality for managing the scraping system. The web interface should provide utility for the following tasks:

- Enabling/disabling the entire scraping system.
- Enabling/disabling individual sources.
- Deleting individual sources.
- Adding new sources.
- Viewing stale sources (this was requested by a member of the BioCaster team in a meeting).

The wireframes for the web interface can be seen in Appendix B.

3.7 System Coordination

The system coordination subsystem is responsible for orchestrating the data collection process and connecting subsystems, as well as allowing the interface to communicate with the scraping system.

3.7.1 Web Connector

The web connector provides the interface between the web server and the scraping system. It ensures that the commands performed on the interface are translated into instructions for the controller, which will perform the requested action. These instructions include enabling, disabling, deleting and adding sources, as well as enabling or disabling the entire scraping system. The web interface can also request updates through the web connector, to show the user the most recent information on the most recent scrape times of each source. Figure 3.7 shows how the web connector integrates with all other modules.

Figure 3.7: The overview of the inputs and outputs of the web connector.

3.7.2 Controller

The controller is responsible for the coordination of the system and is the module called to run the system. It performs the following roles:

- Ensures the tables exist in the database.
- Receives all sources from the database and instantiates crawlers.
- Instantiates parser and classifier object(s).
- Builds the pipelines for data collection and processing.
- Starts the web server for the web interface.
- Controls when the crawlers look for new data.
- Sends new crawler data to the parser(s).
- Communicates with the web interface through the web connector, sending updates and performing instructions.

A full diagram of the controller functionality is shown in Figure 3.8.

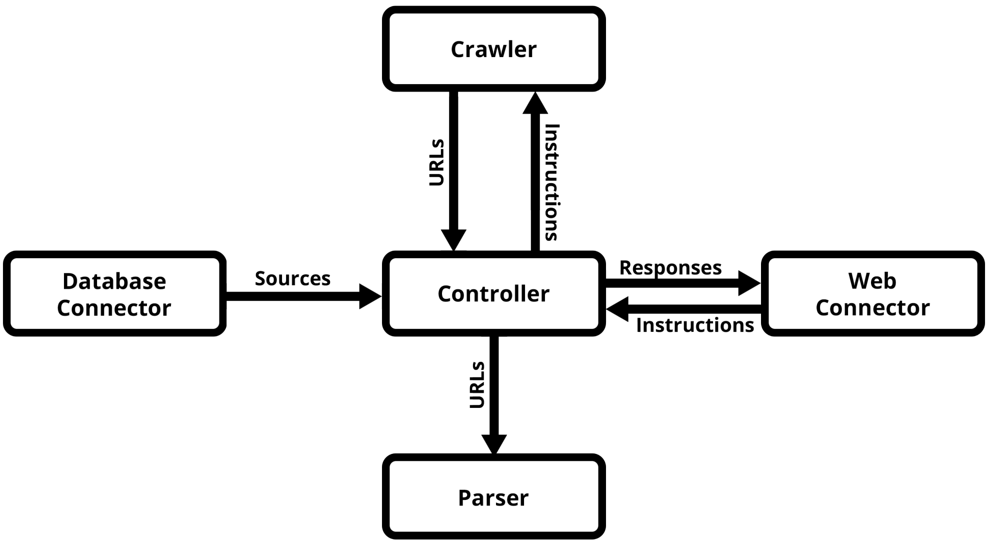

Figure 3.8: The overview of the inputs and outputs of the controller module.

4 Implementation

This section describes the implementation of each of the previously described modules, including details of the algorithms, libraries and technologies used. The system is written in Python as this is a very common language, especially for machine learning programs, and has many libraries available for web crawling, web scraping and database communication.

4.1 Crawlers

Generally, the purpose of the crawler is to return a list of URLs for parsing given a source URL. The crawlers also control the updating of their own state in the database.

4.1.1 Instantiation

Concrete implementations of crawlers will extend the constructor of the abstract crawler, by adding their source type as a parameter and possibly in other ways. The abstract crawler handles the consistent details of instantiation, as listed below:

1. Initialise URL, and "is enabled" variables (crawlers are enabled by default).
2. Initialise language, if given.
3. Initialise source ID if given, otherwise create the source object in the database and initialise the new source ID returned by the operation.
4. Initialise country if given (using a dictionary of country names to ISO 3166 alpha-2 code if necessary), otherwise attempt to mine it from the URL (e.g. ".co.uk" website implies "GB").
5. Initialise the last scrape time if it exists, otherwise set it to a default value.

The abstract crawler also includes all functions which interface with the database connector, using its source ID: Update last scrape time to a given time (if it is more recent than the current time), enable/disable and delete. These functions are called in response to commands from the web interface. If specified in the config, a crawler can automatically disable itself if it is stale. The main "crawl" method which is called by the connector will retrieve a list of new articles using a method overridden by a concrete implementation, and then return all non-duplicate URLs (checked by searching for each of the URLs in the database) packaged with the language, country, source name, source type and a reference to itself so that the parser can update the crawlers last scrape time when it finds the date of an article.

4.1.2 Web Crawler

To retrieve new articles, the web crawler uses the *Newspaper3K*[1] library to create an object which scrapes the whole website of a given URL. It is designed to be fed the homepage URL of a news site, and creates an object which crawls the pages for all available links to news articles, stored in the "articles" property. The web crawler returns all of these articles which do not contain special URL characters ? or # (to prevent duplicate articles).

[1] https://github.com/codelucas/newspaper

4.1.3 RSS/Atom Feed Crawler

RSS Feed entries are collected using the *feedparser*[2] library which given a URL to an RSS or Atom feed will create a dictionary representing all of the RSS feed data. The crawler will iterate through each feed entry, and for any published after the most recent scrape date of the crawler will return the article URL. If no dates are attached, the crawler will return all articles from the feed.

If not supplied a country, the feed crawler will attempt to determine the language from a "language" property in the RSS feed. If this property does not exist, the crawler will use the *langdetect*[3] library to attempt to detect the language of the headline of the first entry. This library uses the Google translate API to return a predicted language for a given text input. If it cannot detect the language, it will set the language to unknown.

4.2 Parser

The parser module will receive a combined list of all of the new URLs obtained from each crawler. It will retrieve the details (title, body and publish date) of each given item using a method defined in a concrete implementation, and update each crawler with the new most recent article retrieval date. It will then pass the parsed results to the classifier module.

4.2.1 Instantiation

The abstract parser is instantiated with two arguments: A classifier, which parsed articles will be passed to, and a lock, which is used to reserve access to the crawler objects while their most recent scrape times are being updated, to avoid inconsistent updates from multiple sources at the same time which may lead to some of the operations not taking effect. The parser also retrieves a minimum article length from the config, to ensure that articles which do not contain enough content are filtered out.

4.2.2 Article Parser

Choice of library

The main free and open-source technologies for scraping news articles from websites in Python were *Newspaper3K* and *news-please*[4], which is built on top of Newspaper3K and adds some extra features. Another option considered was *Newscatcher*[5], but the free API is limited in how many calls can be made and this made it unsuitable for this project. Finally, we considered *pygooglenews*[6] which provided some promise in using google news to find articles under certain subjects, keywords, languages and regions. For this project, We found it desirable to have better control of the exact sources collected, instead of filtering through keywords and relying on Google's source selection, but a crawler using this library could easily be added to extend the capabilities of the current system. We decided to move forward with the former two libraries and conducted an experiment to compare their capabilities.

We compared the features present in each of the two libraries. Notably, Newspaper3K can perform full website scraping in Python, whereas news-please can only do this using its Command Line Interface (CLI). We attempted to scrape 3 articles from each of 109 previously selected websites, across 10 languages (English, French, Spanish, Portuguese, Russian, Indonesian, Mandarin,

[2] https://pypi.org/project/feedparser/
[3] https://pypi.org/project/langdetect/
[4] https://github.com/fhamborg/news-please
[5] https://newscatcherapi.com/news-api
[6] https://github.com/kotartemiy/pygooglenews

Language	No. Sources	Newspaper3K		news-please	
		No. Correct	Average time (ms)	No. Correct	Average time (ms)
English	28	**28**	**536.3332**	26	704.3305
French	11	11	1548.167	11	**806.2918**
Spanish	10	9	**339.2322**	9	435.0289
Mandarin	9	8	3538.55	**9**	**3137.15**
Russian	9	8	**632.5125**	8	755.09
Portuguese	10	10	**1220.08**	10	1289.466
Indonesian	5	5	**1152.32**	5	1243.918
Swahili	7	6	**874.445**	**7**	1253.669
Korean	6	6	**1629.06**	6	1766.223
Arabic	14	**12**	**513.9133**	11	651.8164
Overall	109	**103**	**1198.4613**	102	1204.2984

Table 4.1: The results measuring correctness and speed of the two news scraping libraries on 109 sources in 10 languages. For each source, 3 articles were selected and each article was scraped three times. An average of all of these scrapes per language (in ms, to 4 decimal places), as well as the number of sources correctly parsed, is shown in the table. An overall result is shown, including the average scrape time per language and total number of sources correct. Bold indicates the best performing library in correctness (if not tied) and speed.

Korean and Arabic), and compared the number of successful scrapes (without error) and the average speed. The results are shown in Table 4.1.

Newspaper3K scraped 103 of the 109 websites (94.5%) without error, whereas news-please scraped 102/109 (93.58%). The average scraping times are similar in both libraries, but Newspaper3K was faster at scraping in 8 of the 10 languages, and average scraping time per article was 14.82% lower. Newspaper3K also has the added advantage of requiring one less library, as it is already used in the web crawler. Based on these factors, We chose to use Newspaper3K for the scraping system.

Parsing news articles

To parse the given list of URLs with extra details, the article parser implements the following algorithm:

1. Maintain a list of (source, publish date) pairs and currently parsed articles.
2. For each article URL:
   (a) Attempt to download and parse the URL with Newspaper3K, using the article source's main language.
   (b) Discard the article if it does not exist or have a title.
   (c) If the article does not have text, attempt to detect the language of the article headline (using the *langdetect* library).
   (d) If the detected language is different to the original language, re-download and parse the article in the new language.
   (e) Discard the article if there is still no text, or if the text is below the minimum article length.
   (f) If the parsed article has a publish date, create a new (source, publish date) pair with the article source and parsed publish date.
   (g) Create a parsed article object which includes URL, title, main text, language, country, publish date, original source and original source type.
   (h) Add the parsed article to the list of parsed articles.
   (i) If an exception occurs (e.g. a connection error or a forbidden URL), skip the article.

3. For every (source, publish date) pair, update the source with the new publish date (as the source only updates with newer publish dates, we do not have to find the newest update for each source).
4. Return the list of parsed articles.

4.3 Classifier

The classifier module will apply a machine learning model to parsed article data, and send the parsed articles and classifications to the database connector to be added to the database. The abstract classifier class provides the basic structure and some common functionality of the classifier module, such as a constructor which initialises the model and tokenizer, input batching and communication with the database. A concrete classifier implementation will have to implement:

- A method for initialising the tokenizer (if necessary).
- A method for initialising the model.
- A method for preparing the input articles for classification (extracting the exact input from the article properties).
- A method for tokenizing prepared inputs.
- A method for classifying a tokenized input batch.

4.3.1 BERT Classifier

The concrete version of the classifier uses the best performing model on collected real-world training data (from Section 6.2.6), loading the pre-trained model from a file with path specified in the constructor.

Training data collection

The sources and keywords for real-world training data were decided as discussed in Section 4.2.2. Any category specific RSS feed was scraped by the system, assigning the corresponding category to each data article before adding to the database. For each news website with category specified in URL, a dictionary of URL keyword to category was created from a manually collected list of category-specific URLs, and any website whose URL contained a category keyword in an appropriate place (e.g. **sports**.sina.com.cn or aljazeera.net/**arts**). Before parsing, the URLs were filtered into categories using this matching system, and any articles which didn't match were discarded. Any of these articles which could correctly parse were added to the database with their corresponding category. This process continued until enough articles (at least 10,000) were collected for training.

Model training

The steps involved in creating the trained model for use in the classifier module are as follows:

1. Load the real-world dataset from file and reformat into a dataset object.
2. Split the dataset into a train and test set.
3. Load the model and tokenizer from Huggingface[7].
4. Perform a grid search in parameter space for hyperparameter optimisation using the training set to find the best model parameters.
5. Using the best parameters, train a new copy of the model on the training data and evaluate on the test data.
6. Save the trained model to a file, which can be imported by the classifier module.

[7]https://huggingface.co

Models were loaded, trained and evaluated using the Huggingface libraries. Huggingface hosts many pre-trained transformer models and datasets uploaded by users on its website, which can easily be downloaded and imported into a Python program using one line of code. The library also allows for saving pre-trained models on the website under a user account. The *transformers*, *datasets* and *evaluate* libraries from Huggingface make each stage of model training and evaluation very easy by providing an intuitive API which can easily connect with standard machine learning libraries such as pytorch[8], which was used in this project. Hyperparameters were optimised using the Optuna[9] backend

Batch inference

To obtain classifications from the model, the following steps are taken:

1. Concatenate the article title and body text as model input.
2. Tokenize the concatenated models using the Huggingface model tokenizer.
3. Feed the model a batch of up to 32 inputs and obtain the logits (log likelihoods) of each item using model(inputs).
4. For each input item, select the most likely category by selecting the largest logit value.
5. Using the models map of category ID to category name, obtain the category name of each input item's most likely category.

4.4 Database and Visualisation

The data from the article collection and processing system is stored on an elasticsearch[10] database. Elasticsearch is the database system used in the original BioCaster system, and allows for easy integration into the Python article collection code and into the visualisation system. It is a noSQL no-schema database which stores entries in unstructured JSON format. The unstructured nature of the data means there are no foreign keys, so some data must be duplicated (source name, country and data type must be stored for each article despite being inherited from the original source). The lack of structure will make the queries required for loading sources and checking for duplicate articles less efficient, but these queries are required at a low enough rate that the system should be able to easily process them. The advantages of elasticsearch are the ease of creating real-time visualisations from the data using the Elastic software stack[11] and the ease of integrating the new system into the current BioCaster system as the infrastructure is already in place for an elasticsearch database.

The BioCaster visualisation is created using Elastic Kibana[12], which provides an interface for creating highly customised interfaces by combining different pre-made modules with different attributes and aggregations from the underlying elasticsearch data. These visualisations can be embedded onto an HTML page and will automatically refresh to represent new data collected. In this project we use a locally hosted database and visualisation system, hosting two separate docker containers for elasticsearch and kibana, but elastic also supports cloud storage and data visualisation.

4.4.1 Database Connector

The database connector is written using the *elasticsearch*[13] Python library, which provides an interface which abstracts some of the JSON request-response system used by elasticsearch, The

[8]https://pytorch.org
[9]https://optuna.org
[10]https://www.elastic.co
[11]https://www.elastic.co/elastic-stack/
[12]https://www.elastic.co/kibana/
[13]https://elasticsearch-py.readthedocs.io/en/v8.6.2/

library establishes a connection with the elasticsearch cluster, authenticating securely through HTTPS using a username and password, and for local connections with an SSL[14] certificate. This connection is saved as an object which is used to perform all requests to the database system. Updates and searches are performed by specifying a JSON body containing the request details, and responses are returned in a JSON format which is processed by the connector system to retrieve specific attributes, An example request and response is shown in Figure **add figure**. The library also provides a bulk request API, which is used to upload large batches of documents. In the event of a failed batch upload, the system will attempt to upload each article individually.

4.5 Visualisation

The BioCaster visualisation is created using a dashboard on Elastic Kibana. A map widget shows the articles collected by country using the ISO 3166-1 alpha-2 country codes stored in the "Country" attribute of each article. The final visualisation is shown in Figure 4.1.

We chose to change the article density to cover the whole country instead of being a dot, to make the source of the data more visible (so that larger data locations do not obstruct the view of smaller countries) and created a monochromatic green colour map, where darker colours represented more articles retrieved as is common practice in world density maps (Our World in Data 2022; Office for National Statistics 2022).

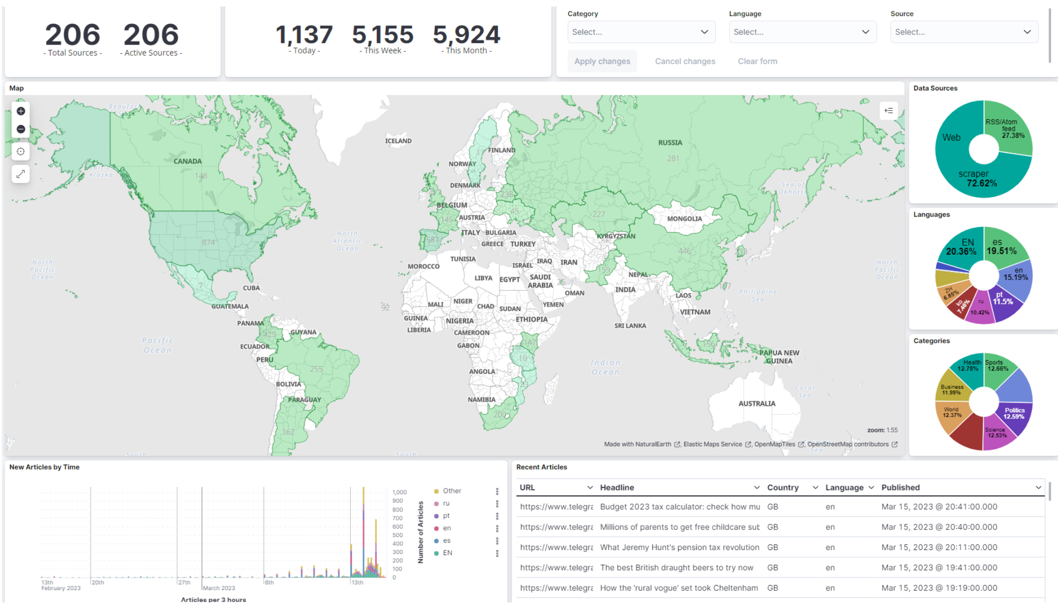

Figure 4.1: The finished visualisation shown on the web interface.

4.6 Web Interface

The web interface was implemented with standard HTML, CSS and JavaScript. It is implemented as a single webpage which can show and hide content to give the appearance of two separate pages: management of the system and management of individual sources. This is done to make the website feel more responsive: the pages will switch instantly instead of having to load another webpage. The overall design is inspired by the original BioCaster website, using the same main

[14]Secure Sockets Layer

colours and element styles to create a cohesive visual design with other BioCaster products. The homepage allows for toggling on and off the scraping system, and viewing or hiding the visualisation shown in Figure 4.1. The final homepage is shown in Figure 4.2. We chose to let the visualisation be toggled on and off as we found it could be distracting when managing the scraping system.

Figure 4.2: *The homepage of the web interface. White space has been removed for clarity.*

The manage sources page shows when sources found their last article, and splits sources which have not found an article in a number of days specified in the config file into a stale section. It also allows for sources to be enabled, disabled, deleted or added. The final manage sources page is shown in Figure 4.3. Stale sources are shown at the top of the page, then enabled sources then disabled sources. We chose this order as we believe it represents the order of importance on a management page: Users will be most interested in which sources are stale, and least interested in inactive sources. Stale sources are also highlighted in yellow to add further emphasis and convey a negative sentiment.

The implementation details of the JavaScript responsive elements are contained in the following section, as the logic is connected with requests to and responses from the Python controller.

4.7 Web Connector

The plain HTML/CSS/JavaScript connects to the code for the scraping system using the Python library *eel*[15]. This library creates an offline web server using a specified directory and by calling a start method with a specified file, will begin running the web server from the Python code.

4.7.1 Communication between Python and JavaScript

By importing "eel.js" in the head section of the index HTML page, an interface is established where the Python and JavaScript modules can pass information aynchronously between each other. Python methods can be exposed to JavaScript using a method decorator, where they can be called in the JavaScript code by calling "eel.<Python method name>(<params>)(<callback function>)" where the return values of the Python function will be given to the callback function which is called when the values are available. No callback function can be specified to return the value into the function giving the original call (but the value must be awaited as communication is asynchronous). Python modules can also call JavaScript functions but this is not used in this project. The communication between JavaScript and Python is illustrated in figure 4.5.

4.7.2 Dynamic Functionality

The dynamic elements of the page are implemented using JavaScript events: functionality is triggered when the page loads and when buttons are clicked. The visual changes to buttons

[15]https://github.com/python-eel/Eel

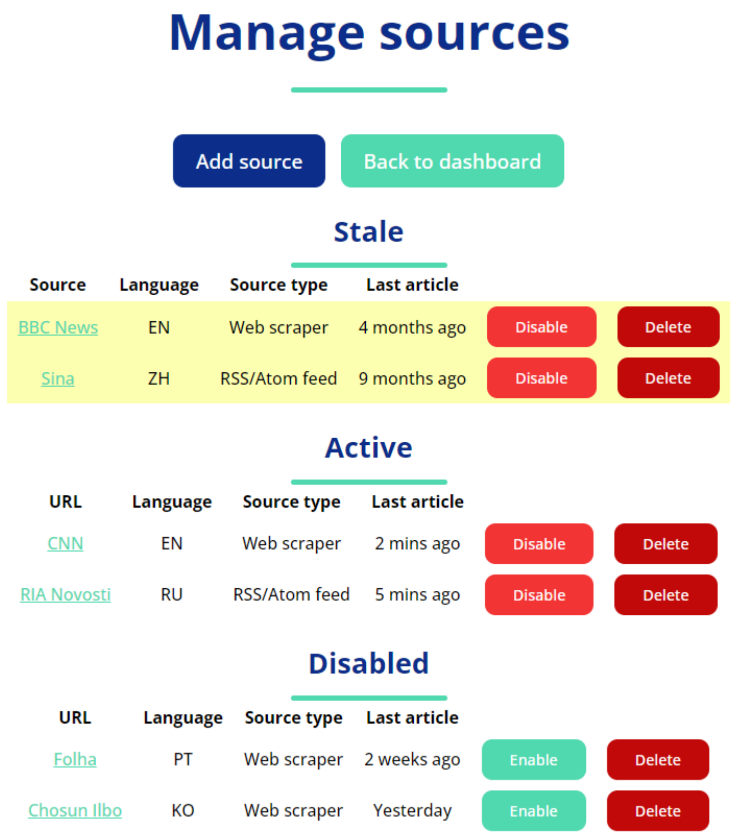

Figure 4.3: *The manage sources page of the web interface. White space has been removed for clarity.*

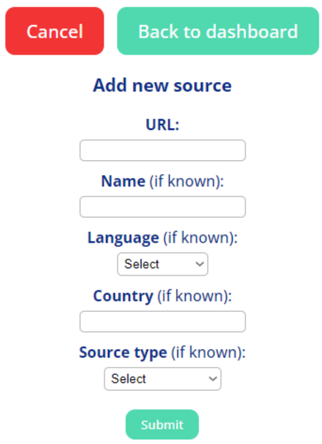

Figure 4.4: *The add source box displayed when the "Add source" button is clicked on the manage sources page of the web interface.*

(e.g. a change in colour of the enable/disable button when toggled) are achieved by adding and removing predefined CSS classes to the clicked elements, and any updates to the state of the system or sources are achieved through communication with the Python controller using eel.

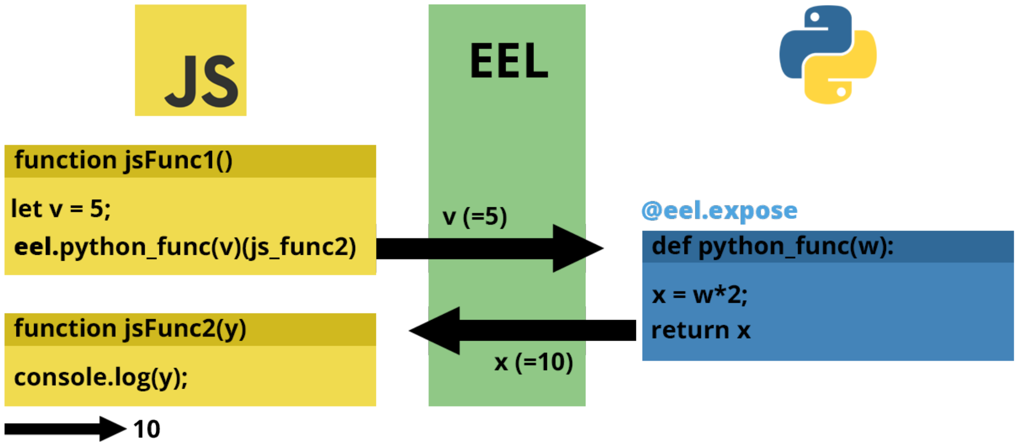

Figure 4.5: A diagram showing the process of calling a Python method from the JavaScript code on the web server, using eel. This uses a simple example where jsFunc1 initialises a value, sends the value to Python function python_func which doubles the value and returns it to jsFunc2, which prints the result. The @eel.expose decorator on python_func makes the function available from the JavaScript file

An overview of the JavaScript functionality and necessary Python methods for communication is given below:

Adding sources to the table

Given a new source, the following steps are performed:

1. If the source is active, increment the number of active sources.
2. Get the difference in time between the current time and the time of last new article.
3. Check if the source is stale (if this difference is larger than the number of days until a source is stale).
4. Generate a string for the last scrape time (e.g. "5 minutes ago") given this difference.
5. Decide which table the row should be in (disabled table if it is not active, stale table if it is active and stale, active table otherwise).
6. Using the date string, stale boolean value and the data from Python, build the HTML for a table row.
7. The ID of the buttons on this new table row will contain the source ID, so that it can be tracked for modifying the source later.

The implementation of this functionality is shown in Appendix E.

JavaScript Events

- **Page loads:** On page load, the number of days until stale and system state are obtained from the Python code, and the state of the Start/stop scraping button is updated accordingly. A method is also started that periodically (once per specified number of seconds) calls the Python **get sources list** method and uses the results to repopulate the tables.
- **Start/stop scraping button:** When clicked, the button calls a method which changes its own text to "Stopping..." or "Starting..." and calls the Python **toggle system** method. When it gets a response, it changes the text and colour of the button (by changing CSS class) to represent the new state and also changes the subtext of the webpage between "Scraping <no. active sources> sources" and "Not currently scraping".

- **Show/hide visualisation button:** When clicked, the button calls a method which changes its own text and sets the display of the visualisation embed to "none" (making it invisible) or "block" (making it appear on the page, below the button).
- **Manage sources button:** When clicked, the button calls a method which toggles the visibility of the div containing the entire manage sources page (changes display from "none" to "block").
- **Back to dashboard button:** When clicked, the button returns the user to the dashboard by setting the display of the div containing the manage sources section to "none".
- **Add source / cancel button:** When clicked, the button changes its own name and colour (from "Add source" in blue to "Cancel" in red) and toggles the visibility of the form for adding sources (shown in Figure 4.4).
- **Submit button:** When the submit button is clicked, all information is extracted from the form inputs. If the input is valid (contains a name, URL and specifies source type), it is sent to the Python **add source** method. When the source details are returned from this method, the source is added to the active table. If the input is invalid error text is shown.
- **Disable/Enable buttons:** When clicked, the button changes its text to "Disabling..." or "Enabling..." and calls the Python **toggle source** method. When the value is returned, it checks whether the operation was successful. If it was successful, it copies the information from the original table row and deleted the table row. It then creates a new table row in the next table (disabled table if being disabled, otherwise active or stale table depending on the last scrape time of the source). If the operation was unsuccessful, the button text is reverted back to its original value.
- **Delete buttons:** When clicked, the button changes its text to "Deleting..." and calls the Python **delete source** method. When the value is returned, it checks whether the operation was successful. If it was successful, it deletes the table row. If the operation was unsuccessful, the button text is reverted back to "Delete".

Python Methods

- **Get days until stale:** Returns the number of days until a source becomes stale.
- **Get system state:** Returns a boolean, indicating whether the system is currently active.
- **Get sources list:** Returns a list of source crawler objects, including every parameter needed by the JavaScript code (ID, URL, name, language, source type, time of last scrape, if the crawler is active).
- **Toggle system:** Turns the system on or off, returning the new state.
- **Toggle source:** Given a source ID, finds the source in the list of sources and enables/disables it. Returns a boolean indicating whether the action was successful.
- **Delete source:** Given a source ID, finds the source in the list of sources and deletes it (calls the crawler delete method and removes it from the list). Returns a boolean indicating whether the action was successful.
- **Add source:** Given the information from the add source form, creates a new source object and adds it to the source list. Returns the required attributes of the new source object for the JavaScript code (ID, URL, name, language, source type).

4.8 Controller

The controller is the module which is run to start the entire system, and when run will perform the following tasks:

- Define the web connector Python methods.

- Load variables from the config (whether to auto-disable stale sources, maximum number of sources to scrape, whether to run the web app, days until crawlers are considered stale, minimum number of seconds between scrapes).
- Ensures the database indices are created through the database connector.
- Load the sources from the database and instantiate the crawlers.
- Instantiate parser(s) and classifier(s).
- Build pipelines from data collection to storage in database.
- If the web app is enabled, start the web app.
- Start the scraping system.

4.8.1 Multi-threading

As the eel web server will block the code when running, only continuing to run when the web server is stopped, it is necessary to start the scraping system on a separate thread to allow both to run simultaneously. Some components are required to ensure components are properly synchronised, such as a lock on the list of crawlers, which is passed to parsers and acquired by any part of the code (parser or Python web interface function) intending to make changes to the internal state of a crawler, to ensure all operations are performed properly. The scraping system thread also uses an event, which is used to communicate between threads that some condition has been met. In this case, when the web server is stopped it will set the event, and the scraping system periodically checks if the event is set, and will stop if it is.

4.8.2 Pipelines

The controller defines data pipelines, which control which sources are passed to which parser and classifier, allowing for different sources of data to be processed differently. The pipelines take the form (list of data source types, parser), where the parser is an initialised parser object (with a given classifier). Any sources in the list which are in the list of data source types will be sent to the corresponding parser. To ensure no data duplication, each type of data source can only be sent to one type of parser.

4.8.3 Scraping System

The scraping system is given the list of crawlers and list of pipelines on execution. The algorithm is as follows:

1. Initialise a list of article URLs for each pipeline.
2. Save the current time as the start time.
3. For each crawler in the list of crawlers.
    (a) Block until the scraper system is enabled, checking for the event to be set.
    (b) Run the crawler to obtain a list of article URLs.
    (c) Find the first pipeline which includes the type of crawler (or discard the data if there is none).
    (d) Add the URLs to the pipelines list.
4. For each pipeline, run the parser on the corresponding list of article URLs.
5. Reset the list of URLs for each pipeline.
6. Get the time elapsed by subtracting the start time from the current time.
7. If the time elapsed is less than the minimum time per scrape, sleep for the rest of the time, checking periodically for the event to be set.
8. Repeat from step 2.

4.9 Summary

The system is primarily written in Python although the web application uses JavaScript. The crawlers use the database connector to update their own state, and concrete implementations use an RSS parser to crawl RSS feeds and a web crawling library to build the profile of a news website and find all article links. The article parser uses the Newspaper3K library to turn the list of URLs into parsed objects, which it sends to the classifier. The BERT classifier is built using Huggingface transformers and uses a previously trained transformer model to categorise each article object from its headline and body, selecting the maximum log likelihood category on inference. The categorised objects are sent by a Python database connector to a noSQL elasticsearch database, which directly feeds into the Elastic Kibana visualisation system. These visualisations are embedded on a locally hosted web app by eel, which provides an interface for communication between the web app code and the Python code. This interface is used to send the web app updates on the state of sources, and for the web app to send instructions to change source states. The controller orchestrates this process, hosting the web connector functions and building pipelines for data collection and processing.

5 News Article Classification

5.1 Task Definition

By researching the topics used by the news sources previously collected, as well as considering the topics used in publicly available news classification datasets, a set of six news topic categories was chosen, C = {Entertainment/Arts(0), Sports(1), Politics(2), Science/Technology(3), Business/Finance(4), Health/Welfare(5)}

Given a dataset of news articles N with and corresponding labels $L \in C$, which denotes the category of the given news item. A news article $n \in N$ is composed of sentences, which are composed of words. An arbitrary article $n_i \in N$ can be represented as $(s_1, ..., s_I)$, where I denotes the number of sentences in n_i. Each sentence $s_j(1 \leq j \leq I)$ can be represented as $(w_1, ..., w_J)$, where J denotes the number of words in sentence s_j. Each news article will be composed of words in one of six languages: English, French, Spanish, Portuguese, Mandarin or Indonesian. The objective of this experiment is to train a multilingual model M to correctly classify an unseen article $n_new \notin N$ into label $L \in C$. A diagram of the task definition is shown in Figure 5.1.

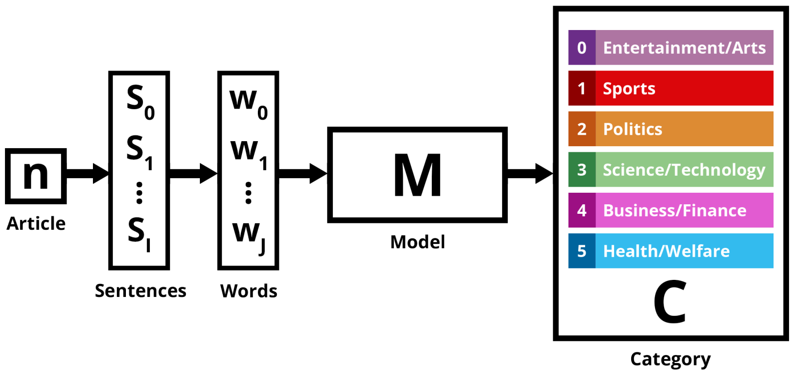

Figure 5.1: A diagram representing the flow of a model classifying a news article n, composed of sentences which are composed of words. The news article is eventually assigned a category from C.

5.2 Multilingual Models

5.2.1 Preliminaries

Transformer models

The first transformer model was introduced in Vaswani et al. (2017). It was built by combining an encoder model, which creates a multidimensional vector embedding of input sequences, with a decoder which converts these embeddings to the desired output (e.g. a classification). The architecture is shown in 5.2.

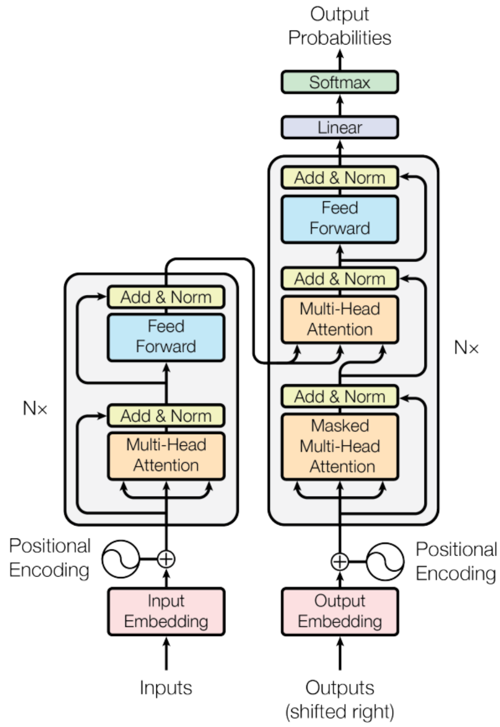

Figure 5.2: *A diagram of the transformer architecture. Obtained from Vaswani et al. (2017).*

Important features of the transformer model which enable its success on text classification tasks include:

- Residual connections which allow previous layers output to directly propagate to later layers in the network.
- Positional encoding using sine functions to create normalised representation of an items position, instead of using a single index.
- Multi-head attention which can represent the importance of other words in the sequence on the current word.
  - Attention is a mechanism which allows a weighted sum of surrounding words to factor into the context of the current word, improving the representation of words with different definitions (e.g. a river **bank** vs a financial **bank**).
  - Multi-head attention repeats this process across multiple attention heads. The results of these heads are concatenated and combined using a fully connected neural network. This can capture the fact that words may relate to each other in multiple ways (Vaswani et al. 2017).

BERT models and fine-tuning

A paper by Devlin et al. (2018) introduced the BERT model. This model is based on the original transformer architecture, but removes the decoder section in order to become sequence-to-sequence (the model transforms an input sequence to an output sequence). BERT is trained using masked language modelling (MLM), where a percentage of input sentences are hidden from the model and it attempts to guess the missing words. It is also trained on next sentence prediction (NSP) where the model guesses if one sentence follows another. These training tasks allow the model to contextualise words and sentences to create a better understanding and representation of the context of words. When trained on a multilingual corpus, the idea is that these representations are language-agnostic, and allow for successful classification of unseen multilingual data. These representations achieved from pre-training can be fine-tuned to complete a down-stream task, such as multilabel classification, by ignoring all but one of the model outputs and training a classifier on top of this output using a new dataset for the exact downstream task (e.g. news articles with categories) to convert the contextual embeddings of words into a classification label. This process is illustrated in Figure 5.3. The fine-tuning process allows the models to leverage their advanced representations of words to adapt to many different tasks with far less training time on each individual task.

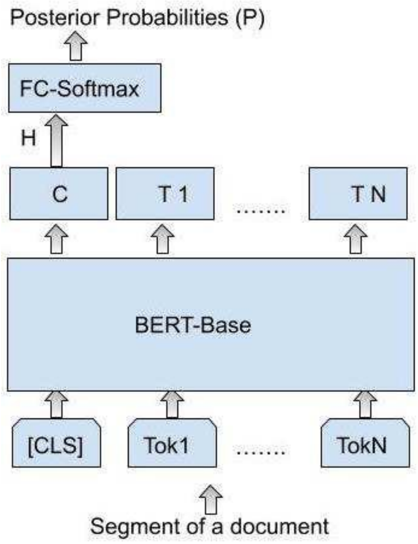

Figure 5.3: *A diagram of the architecture of a fine-tuned BERT model on text classification. Obtained from Pappagari et al. (2019).*

5.2.2 Models Used

The models being used for this experiment were introduced in Section 2.2.2: Multilingual BERT (mBERT) and XLM-RoBERTa. The models were trained on many languages, including the six considered here. The key differences between the models are highlighted below:

- The models are pre-trained on different corpora: mBERT is trained on the Wikipedias of the 104 languages with the most content. XLM-RoBERTa is trained on a larger corpus, using 2.5TB of CommonCrawl data across 100 languages.
- mBERT uses WordPiece tokenization, where each sentence is split into words by spaces or punctuation, and each word is broken into one or more subwords. Each unique subword is associated with a token ID, and input data is converted into a sequence of these token ID's, which are input into the mBERT model for classification. XLM-RoBERTa uses SentencePiece tokenization, a similar subword tokenization scheme which is lossless (it can exactly replicate the input sequence as it retains whitespace and punctuation information) and language independent (Kudo and Richardson 2018).

5.3 Fine-tuning Scheme

This experiment uses three datasets: A public English-language news classification dataset **DS1**, an augmented version of this dataset where all articles have been translated into all 6 languages (including the original English) **DS2**, and a dataset of real-world news articles collected from our developed scraping system **DS3**. Information on each dataset is detailed in Section 6.2.2. Data from each dataset was tokenized using the tokenizer associated with the given multilingual model. For each model, the following fine-tuning experiments were performed:

1. The models were trained and evaluated on the English public dataset **DS1**.
2. The models were trained and evaluated on the multilingual public dataset **DS2**.
3. The previously trained models on **DS1** were evaluated on the real-world collected data (**DS3**).
4. The previously trained models on **DS2** were evaluated on the real-world collected data (**DS3**).
5. The models were trained and evaluated on the real-world collected data (**DS3**).

A diagram of these experiments is shown in Figure 5.4.

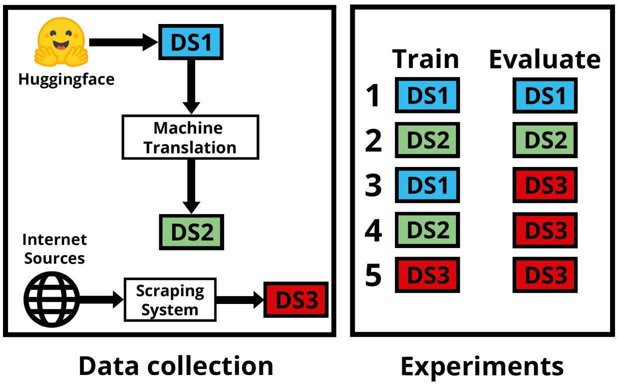

Figure 5.4: A diagram of the fine-tuning scheme for the news article classification experiment, showing how datasets were obtained and the experiments run for each model (training dataset and evaluation dataset.

6 Evaluation

6.1 Digital News Surveillance System

6.1.1 Research Overview

We conducted a usability test on a static version of the web app (not connected to a scraping system) including the final visualisation. We created a questionnaire and sent it to university students and members of the BioCaster research team, receiving 9 responses. This evaluation was conducted following the university ethics guidelines, and a signed ethics checklist is shown in Appendix F.

The purpose of the evaluation was to answer the following research questions:

- **RQ1.1:** Does the visualisation provide a complete and effective overview of the data collected?
- **RQ1.2:** Does the interface allow users to easily and effectively manage the underlying web scraping system?

The respondents were asked to perform the following tasks, and answer questions on their experiences and opinions on the visualisation and web scraper:

- Toggle the scraping system and the visualisation.
- Adding a source.
- Enabling/disabling a source.
- Deleting a source.

Respondents used a scale from 1–5 (1 being the most negative response) to rate the visualisation on how easy to understand it is and how well it represents the data. They were also asked to rate the web interface on how intuitive, clear to present information and aesthetically appealing it is. Open questions were also asked to elicit more detailed feedback. The results of the rating scale questions are shown in Figure 6.1.

The full questionnaire can be found in Appendix G.

6.1.2 RQ1.1: Visualisation

Respondents were asked on a scale from 1-5 (5 being the best response), if they thought the visualisation was easy to understand, and if the visualisation represents the data well. All respondents gave a 4 or higher on both questions, with 4 respondents (44.4%) rating the visualisation a 5 in ease of understanding and 7 respondents (77.8%) rating the representation of data collected a 5. These results suggest the visualisation succeeds in representing the data effectively in an easy to understand way, but could use some minor improvements to increase the clarity of data.

Specific examples of feedback given on the visualisation are:

- It is unclear what the colours in the density map represent, and the colours are too similar.
- Filtering recent articles by country when clicking the map would be a useful addition.
- The visualisation could better fit the whole page without scrolling.

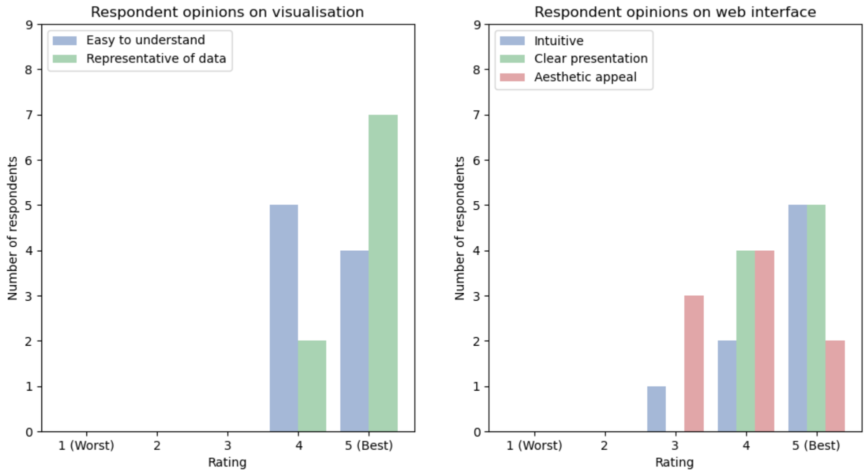

Figure 6.1: *The results of the rating scale questions of the questionnaire, gauging respondent opinions on various factors of the Kibana visualisation and the web interface.*

- Languages should have full names as the two letter representations can be hard to understand.
- The screen can feel full, it would be useful to be able to adjust components.
- A pie chart by country isn't effective for geographic distribution, some way to signal proximity would be useful.
- Top sources by number of articles would be a useful statistic.
- Categories of article over time may be an interesting metric.
- More information on source bias/reliability would be useful in considering the reliability of information.
- The "new articles by time" component can be difficult to interpret.

The feedback is reasonable, and although many of the recommendations are outside the scope of this project (e.g. source bias and geographic proximity), many of these changes are easy to implement and would immediately improve the visualisation, such as better distinction in the colours and more easily readable language names. The requested additional visualisations could be used to replace less effective visualisations in the current system, such as articles by time.

To answer **RQ1.1**, the results suggest we have succeeded in creating a visualisation which provides a complete and effective overview of the data. There were some suggestions for extra features or visualisation components and criticism of existing components, but overall the respondents believed the visualisation was a complete representation of the data, giving a 4.78/5 average rating. Respondents also found the visualisation easy to understand, giving a 4.44/5 average rating. There are some issues to be fixed: The colour of the map could use some clarity, both in displaying its purpose and in more visual distinction between different levels of density. The full names of languages should also be added to make the data more clear to people viewing it.

6.1.3 RQ1.2: Web Interface

Respondents were asked to rate the web interface on how intuitive it is, how clearly information is presented and how aesthetically appealing it is. Results are shown in Figure. Respondents thought the information was clearly presented, with 55.6% reponding with 5, and the rest with

4, however 12.5% and 33.3% of particpants gave the web interface a 3 on intuition and aesthetic appeal respectively, suggesting there is area for improvement on these metrics.

Specific examples of feedback on the web interface are:

- Some filters on the visualisation are hard to remove (requiring a refresh).
- Some section fonts are too large.
- The web app could benefit from an "about" page to describe the purpose.

Most of the negative feedback come from the problems with the visualisation. The lack of clear purpose shown to some of the respondents may have worsened the experience but this will not be an issue in the final product, as the web app is only intended to be used by internal team members.

To answer **RQ1.2**, despite the limitations of the usability test, the web interface shows promise as an effective method of managing the underlying scraping system. The interface makes controlling the scraping system simple, as participants found it intuitive (giving a 4/5 average raring) and clearly presented (giving a 4.56/5 average rating). Feedback was less positive overall than with the visualisation, particularly in aethetic appeal, which had an average rating of 3.89/5. The comments suggest, however, that this opinion seems to be tied to technical issues with the visualisation and the lack of clarity in the experiment and mock interface itself, rather than as a product of interface design (which had only a few minor complaints and some compliments). A future experiment using a version of this interface linked to the scraping system would provide more evidence to validate **RQ1.2**, as participants couldn't observe the results of their actions on a real system.

6.1.4 Limitations

Due to the nature of the web interface (hosted on a local web server), it was difficult to host an online version of the site which would be connected to a real scraping system. This meant that it was only possible to evaluate on a mock interface, which some respondents mentioned made the web interface less intuitive as they cannot directly see the results of their actions on the system. In addition, due to many of the respondents working in different countries and having busy schedules, it was impossible to conduct a live evaluation where respondents could be monitored using the app. This would have been more useful to understand the exact feedback given and directly view any bugs or technical issues. The low sample size (n=9) of the evaluation also reduces the ability to generalise findings on the effectiveness of the web interface and visualisation, and limits the range of feedback available. Future research could address these limitations and provide stronger evidence to answer the research questions.

6.2 Multilingual News Article Classification

6.2.1 Research Questions

In many languages, particularly minority languages, little to no labelled public datasets are available for topic classification of news, meaning models cannot be trained by traditional means. This creates difficulty when trying to perform real-time classification tasks on news written in minority languages, for purposes such as creating disease alerts. We will investigate the efficacy of some techniques and approaches for multilingual document classification, by investigating the following research questions:

- **RQ2.1:** How effective is machine translation of English articles into other languages, as a method of upsampling, in increasing the effectiveness of multilingual models?
- **RQ2.2:** How effective are models trained on public data (both in English and translated into all 6 languages) when used to classify collected news data directly from sources across the six languages?

- **RQ2.3:** What level of performance can be achieved by multilingual models when trained and evaluated on collected articles from sources in each of the six languages?

6.2.2 Datasets

Public data

The dataset used is a dataset made available on huggingface by Villanova and Abdulmatin (2022). The data originally consists of 3,722 classified English-language news articles scraped from various online news sources with 7 topic labels: World, Politics, Tech, Entertainment, Sport, Business, Health, and Science. To match the categories being considered in this experiment, articles belonging to the "World" category were dropped, and those belonging to the "Tech" and "Science" categories were merged into a new "Science/Technology" category. An augmented dataset was also created by adding copies of each article translated into each of the other five languages (French, Spanish, Portuguese, Mandarin and Indonesian). Translations were obtained using the Google translate API through the Python library *googletrans*[1]. The distribution of topics in the original and augmented datasets is shown in Table 6.1 and Figure 6.2.

	Number of Documents						
	Entertainment	**Sports**	**Politics**	**Science/ Technology**	**Business**	**Health**	**Total**
Original	485	454	442	838	461	467	**3147**
Translated	2910	2724	2652	5028	2766	2802	**18882**

Table 6.1: *Category distribution of the Valurank news categorization dataset. This table includes the original dataset and the augmented version obtained from adding the translations of each article in each of the other 5 languages.*

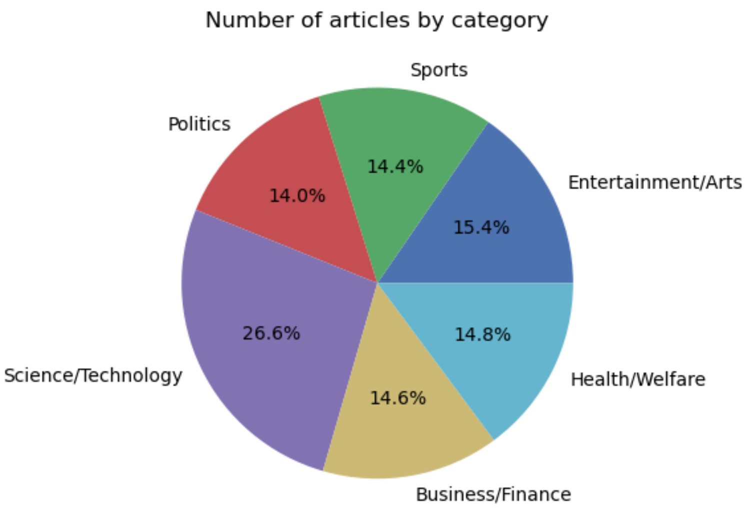

Figure 6.2: *The distribution of categories in the public Valurank dataset*

[1]https://pypi.org/project/googletrans/

Real-world data collection

We collected news articles from some of the most popular online news sources across the six languages. The list of six topics used was created after considering the most common category sections in news sources across different languages, and combining similar categories which were often combined by the original sources (such as business and finance). A set of keywords relevant to each category was also generated, shown in Figure 6.2. Each collected article was self-labelled by the source, and the articles were collected and categorised using the previously developed scraping system, with each article URL obtained and classified either through topic-specific RSS feeds or by matching the URL of article webpages with topics which match category keywords (e.g. abcnews.go.com/Sports). A complete list of data sources used is available in the appendix (link specific appendix).

Category	Keywords
Entertainment/Arts	Entertainment, Art, Arts, Culture, Movies, Cinema, Music, Books, Theater, Television, Dance, Celebrity
Sports	Sport, Sports, Soccer, Football, Basketball, Tennis, Golf, Rugby, Motorsport, Formula 1, Physical Education
Politics	Politics, Election, Parliament
Science/Technology	Science, Technology, Tech, SciTech, Space, Physics
Business/Finance	Business, Finance, Economy, Stock exchange, Stock market, Market, Money, Investment
Health/Welfare	Health, Welfare, Coronavirus, Pharma, Pharmacy

Table 6.2: *List of category keywords used for collecting news data. Sources which had RSS feeds or specific URL patterns with these keywords were selected for the corresponding category.*

Each collected article URL was parsed using the *Newspaper3k*[2] library. Any article with a main body under 100 characters in length was discarded, and the article headline and body were concatenated into one text field, used for classification. Originally data was collected for 10 different languages, but only the six considered returned a sufficient number of results for model training. In the final dataset, "Business/Finance" and "Science/Technology" articles in Mandarin were downsampled in order to balance the categories, by randomly dropping articles of these categories and languages at probabilities of $p = 0.75$ and $p = 0.5$ respectively.

The distribution of articles by category and language is shown below in Table 6.3 and in Figure 6.3.

Language	**Category**						
	Entertainment/ Arts	**Sports**	**Politics**	**Science/ Technology**	**Business/ Finance**	**Health/ Welfare**	**Total**
Mandarin	511	706	329	961	802	217	3,526
English	885	682	435	607	493	390	3,492
Indonesian	478	545	71	295	598	255	2,242
French	342	369	138	88	235	114	1,286
Portuguese	136	197	196	240	232	167	1,168
Spanish	203	400	159	48	238	84	1,132
Total	2,555	2,899	1,328	2,239	2,598	1,227	**12,846**

Table 6.3: *The distribution of collected news articles by language and category.*

[2] https://newspaper.readthedocs.io/en/latest/

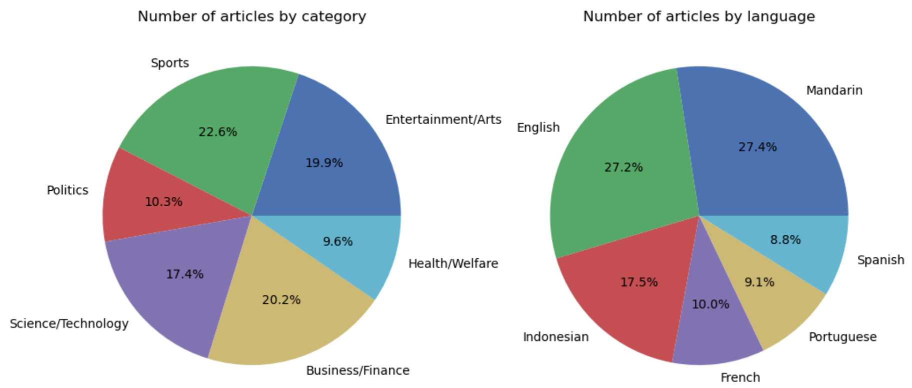

Figure 6.3: The distribution of categories and languages in the public Valurank dataset

6.2.3 Evaluation Setting

Baselines

To test the effectiveness of machine translation upsampling, we will use the models trained on the English only public dataset as a benchmark, to measure how much (if any) accuracy increases using translation to augment the dataset. When comparing how models trained on public data transfer to collected real-world multilingual data, we can use static models which do not learn from the data as a baseline. In this case, we use two static baseline models: a model which uses stratified sampling (it samples a prediction for an input randomly, using the category densities as a probability distribution) and most common sampling (always predicts the category with the most training examples). This allows us to consider how much of what the model learns from the public data can be applied to the real-world data.

Hyperparameters

The hyperparameters investigated in this experiment were batch size (how many examples are used in training per backpropagation cycle), number of epochs (how many times does the model repeat the training data) and learning rate (how quickly does the model update its parameters to improve predictions). The values considered are based on the recommendations and experiments from the original BERT paper, by Devlin et al. (2018), and shown in Table 6.4. In each case, an exhaustive grid search was performed (24 trials) on the hyperparameter values and the highest accuracy model was chosen for the evaluation. The optimal hyperparameters found for each model is shown in Appendix D.

Parameter	Values
Batch size	16, 32
Number of epochs	2, 3, 4
Learning rate	5×10^{-5}, 4×10^{-5}, 3×10^{-5}, 2×10^{-5}

Table 6.4: A list of hyperparameters tuned before training each model for classification. In each case, a grid search was performed with these values.

Data splitting

Because the categories in our datasets were not too imbalanced (the largest category disparity is around 2:1) we decided that data can be split randomly into a train and a test split. For each dataset, a train/test split of 80/20 (80% training data) was used.

Evaluation metrics We will evaluate the performance of each model using the classification accuracy (which percentage of predicted categories are correct), as well as its weighted F1 score. These ratings are widely used in machine learning literature, including classification tasks (Alam et al. 2020; Al-Masni et al. 2020; Ranasinghe and Zampieri 2020). F1 score is a common metric for effectiveness in classification problems, which aims to balance precision and recall. Precision measures the proportion of predicted positive labels (true positives and false positives) are true positives. Recall measures how many of the true positive data items (true positive and false negative predictions) were classified correctly as positive (true positive). F1 score is the harmonic mean of precision and recall, and aims to balance these metrics to favour a model which avoids false positives but also misses as few positive examples.

In a multiclass context, the F1 score for each possible label is calculated separately and combined. There are different methods for combining these scores,but in this experiment we will use weighted F1 score, which takes a weighted average of category F1 scores based on the category weighting. The weighted F1 score for a dataset D with categories $\{c_1, c_2, ..., c_N\}$ can be calculated as:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1\ score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

$$Weighted\ F1 = \sum_{i=1}^{N} w_i \cdot F1\ score_i \qquad w_i = \frac{|D_{d \in c_i}|}{|D|}$$

Where TP, FP and FN are the number of true positive, false positive and false negative classifications. Each experiment will be repeated for three training runs, and the mean and standard deviation of the accuracy and weighted F1 scores will be used as metrics for the effectiveness of each model.

6.2.4 RQ2.1: Effectiveness of Translation Upsampling

Shown in Table 6.5 are the results of training and testing the multilingual models in the original English-only and augmented translated Valurank public dataset. The results show that all models achieve very high (> 95%) accuracy and F1 score for the public dataset. We can also see that both models achieve an increased performance of 2-3% on the larger translated dataset, reaching near-perfect (> 98%) accuracy and F1 score in both cases. In both cases, the mBERT model performs slightly better than the XLM-RoBERTa model, but there is very little difference (< 1%) in accuracy.

Model	English Data		Multilingual Data	
	Accuracy	Weighted F1	Accuracy	Weighted F1
mBERT-base	**96.50±0.22**	**96.52±0.22**	**98.77±0.10**	**98.77±0.10**
XLM-RoBERTa-base	95.66±0.37	95.66±0.34	98.70±0.00	98.70±0.00

*Table 6.5: Accuracy and weighted F1 score (mean ± standard deviation) of models trained and tested on the Valurank public dataset, both in English only and translated into 5 additional languages, over 3 training runs. The most effective model for each dataset is denoted in **bold**.*

To answer **RQ2.1**, machine translation of news articles does show some promise as an effective means of upsampling annotated public data, yielding an improvement in classification accuracy and F1 score over training on the monolingual dataset. For smaller datasets, this could be an effective way of synthesising much more training data to allow multilingual models to learn more effectively and achieve increased performance. Future research is required to determine if machine translation is also effective in upsampling multilingual data.

6.2.5 RQ2.2: Transfer of Models to Real-world Data

Table 6.6 shows the results of transferring the models trained on public data to real-world collected data. The trained multilingual models perform significantly better than the baseline models, but achieve significantly lower accuracies and F1 scores than when they are evaluated on the public datasets. This suggests there are some similarities between public news datasets and real-world news, but not enough that public datasets can be an effective substitute. The XLM-RoBERTa model seems to generalise more effectively on both training sets, achieving 7-9% better accuracy and F1 score than the corresponding mBERT model. In both models, there is no significant difference in how the models perform on real-world collected data when trained on English or multilingual data.

Model	**English Data**		**Multilingual Data**	
	Accuracy	Weighted F1	Accuracy	Weighted F1
Stratified Sampling	17.72%±0.80	17.74±0.80	17.72%±0.80	17.74±0.80
Most Common	23.54%±0.00	8.97±0.00	23.54%±0.00	8.97±0.00
mBERT-base	65.78%±0.21	65.96±0.14	66.03%±0.51	66.00±0.48
XLM-RoBERTa-base	**74.78%±1.27**	**74.92±1.24**	**73.38%±0.00**	**73.34±0.00**

Table 6.6: *Accuracy and weighted F1 score (mean ± standard deviation) of stratified sampling and most common static baseline models, as well as multilingual models trained on the Valurank public dataset (English only and translated into 5 additional languages). All models were evaluated on the collected multilingual dataset, over 3 training runs. The most effective model for each dataset is denoted in* ***bold.***

To answer **RQ2.2**, the large increase in performance from the baseline models when the multilingual models are trained on public data and evaluated on collected data suggests that the multilingual models have learned some patterns which still apply to real-world data. However there is enough difference to create a significant decrease in performance from the public dataset. The decrease in accuracy and F1 score when generalising to real-world data is likely due to the differences in how news articles are categorised and written across languages. Even when translated into multiple other languages, the public dataset only captures the writing and categorisation conventions of English-language news and these may not generalise effectively between language and source type. As an example, some Chinese sources categorise lottery results as "Sports", whereas English-language sources almost never do this. This suggests that for effective classification of news in minority languages, effort should be taken to obtain training data which is written in the language instead of relying on machine translation.

6.2.6 RQ2.3: Classifying Real-world Data

Table 6.7 shows the results of training and testing the multilingual models on the real-world collected dataset. There is little significant difference between the two multilingual models, with XLM-RoBERTa achieving slightly higher accuracy and F1 scores of 89.5%. Both models achieve a lower accuracy on the collected dataset than on either of the public datasets, but perform significantly better when trained on the same dataset than on the public dataset. This suggests that real-world news may be more difficult to accurately classify.

Table 6.8 shows the confusion matrix for the best performing model (XLM-RoBERTA-base) on the real-world test data. We can see that the most common misclassifications are predicting Business/Finance as Science/Technology (55 occurrences), predicting Politics as Business/Finance (48 occurrences) and predicting Science/Technology as Business/Finance (39 occurrences). This suggests that there may be large similarities or overlap between these three categories.

Model	Accuracy	Weighted F1
mBERT-base	87.81%±0.24	87.75±0.26
XLM-RoBERTa-base	**89.40%±0.15**	**89.34±0.17**

Table 6.7: *Accuracy and weighted F1 score (mean ± standard deviation) of models trained and tested on the collected multilingual dataset, over 3 training runs. The most effective model for each dataset is denoted in* ***bold.***

Actual label	Predicted label					
	Entertainment/ Arts	Sports	Politics	Science/ Technology	Business/ Finance	Health/ Welfare
Entertainment/Arts	**445**	6	1	19	4	6
Sports	6	**558**	0	3	2	0
Politics	5	1	**196**	11	48	15
Science/Technology	4	3	0	**385**	39	8
Business/Finance	6	3	8	55	**486**	8
Health/Welfare	3	2	0	14	14	**211**

Table 6.8: *Confusion matrix for the XLM-RoBERTa-base on real-world multilingual test data. The matrix shows where the prediction errors lay, and is colour coded so that deeper reds indicate more common incorrect classifications (the green diagonal shows correct classifications).*

To answer **RQ2.3**, While performance of the multilingual models on real-world datasets (shown in Table 6.7) remains high (89% accuracy) there is a significant decrease in performance in the models from the public dataset evaluation (shown in table 6.5. This is likely because real-world data is less simple to sort into one category than most publicly annotated data: category boundaries are often not clear and even human annotators may disagree on classifications, many articles could be considered to fit into multiple categories and some are even given multiple categories by the original news source. The model performs very well on Sports articles, where there is less overlap with other categories and hence less ambiguity, but often confuses Business/Finance, Politics and Science/Technology articles. In real-world articles, these subject areas often overlap: many political decisions affect the economy, and technology is often a topic of political debate. Future research could consider a more appropriate task for news article classification, such as multiclass classification with more labels, using human-annotated data.

6.2.7 Limitations

Due to the lack of public labelled data for news article topic classification in many languages, in order to consider this task in a multilingual setting it was necessary to collect my own public data using RSS feeds and web scraping. The data collected is not annotated or reviewed by a human, and instead solely relies on the tagging of the news sources being used. This means that training data may be mislabelled or contain errors, and results achieved through experiments using these data may not be as valid as results obtained from manually annotated data. A human-validated multilingual dataset would provide stronger evidence for the research questions, and future research could consider collecting this data.

7 Conclusion

7.1 Project Summary

In this project, we improved an existing digital disease surveillance system, BioCaster, by analysing the areas the current system is lacking and developing improvements in these areas. In particular, we found two main issues. The first main issue is that the data collection methods in the current system are very limited, relying on published RSS feeds. These feeds are typically only published by popular news sources in high-resource countries, which means BioCaster is lacking in coverage of disease outbreaks in low-resource areas. We designed an extensible digital news surveillance system based on the BioCaster system, which improves the range of data which can be collected, adding functionality for the scraping of news website HTML, and providing infrastructure to easily extend to other online sources of data, such as social media posts. The system has the following components:

- **Crawlers** produce a list of possible data URL's from a source URL.
- The **parser** scrapes a data URL for structured information, such as headline and publish date.
- The **classifier** assigns a category to structured articles. The classifier used was trained in the multilingual news classification experiment.
- The **database** provides a permanent store for parsed and classified articles. The **database connector** facilitates communication between the database and the news collection and prcocessing system.
- The **visualisation** component provides real-time multi-component visualisations using the data from the database.
- The **Web interface** provides functionality for a user to control the news collection system, by sending instructions and receivign updates through the **web connector**.
- The **controller** starts the program, initialising crawlers from sources stored in the database, creating and running data processing pipelines and communicating with the web interface through the web connector.

The system was designed in Python, using the *Newspape3K* and *feedparser* libraries for collecting news data, and using *huggingface transformers* to classify the news data using transformer models. The database system uses noSQL database elasticsearch, connected to the Python code through thr *elasticsearch* library, and real-time visualisations are produced from this data using Elastic Kibana. The web interface uses standard HTML, CSS and JavaScript, but is connected to the python code through the *Eel* library, which creates a local web server where information and instructions can be passed between the Python and JavaScript code.

We also conducted an experiment with two state-of-the-art multilingual transformer models using the task of classifying a news article in one of 6 languages to one of 6 selected categories. This evaluation used three datasets: A public, English-language dataset available online, an augmented version of this dataset with all items translated into all 6 languages, and a dataset of real-world data collected by the previously developed scraping system. This collected dataset is composed of articles self-categorised by the news sources. We investigated the effectiveness of machine translation as a data augmentation technique to upsample datasets, how effectively models trained on public datasets perform on real-world data. Finally, we trained and evaluated the models on real-world data to evaluate the current accuracy on multilingual news article classification tasks.

7.2 Evaluation Summary

We conducted evaluations on the visualisation, web interface and the multilingual classification experiment, and came to the following conclusions:

- **RQ1.1**: The visualisation is easy to understand and represents the data collected well.
- **RQ1.2**: The web interface provides is intuitive and clear, but the effectiveness of managing the scraping system was difficult to measure due to experiment limitations.
- **RQ2.1**: Machine translation upsampling yields some improvement in classification accuracy.
- **RQ2.2**: Models trained on public data far outperform static baselines on real-world data, but still suffer a significant performance decrease.
- **RQ2.3**: Our best model on real-world collected data was XLM-RoBERTa, which achieves 89.4% accuracy.

7.3 Future Work

7.3.1 Web Scraping System

As discussed in Section 6.1.4, the evaluation on the interface was limited, as respondents were only able to use a mock version of the interface not connected to the news collection system. Future work could find a method for this interface to be hosted online, so that an evaluation can be conducted on a live system.

While being designed to easily accommodate the adding of new sources, there are many potentially useful sources of information which this project doesn't implement, which could be implemented by future projects. Potential sources include data aggregated from google news, social media data and search trends. The system also has some room to improve in efficiency. For example, a future system may use multi-threading to parse multiple articles at once, removing the bottleneck of communication with news URLs in the current single-threaded system.

7.3.2 Multilingual News Article Classification

As discussed in Section 6.2.7, future work may consider multiclass classification as it may be more appropriate for this task. Due to limited online resources and time, the real-world dataset is unlabelled and relies on categorisations from the source themselves. To improve the validity of results and strengthen conclusions, future research could collect human-labelled multilingual news category datasets for use in training the multilingual models, instead of assuming the validity of articles scraped from the internet and labelled by the data source.

Research by Wu and Dredze (2020) suggests that multilingual BERT is less effective in transferring its results to low-resource languages, and pre-training on a multilingual task instead of masked language modelling (MLM) would produce a better model for language transfer. Due to limited resources and time, this pre-training was not possible in this evaluation, but a future study could consider how these methods are applied to the task of multilingual news classification and what results can be obtained from models trained using these methods.

A | Wireframe of proposed new visualisation

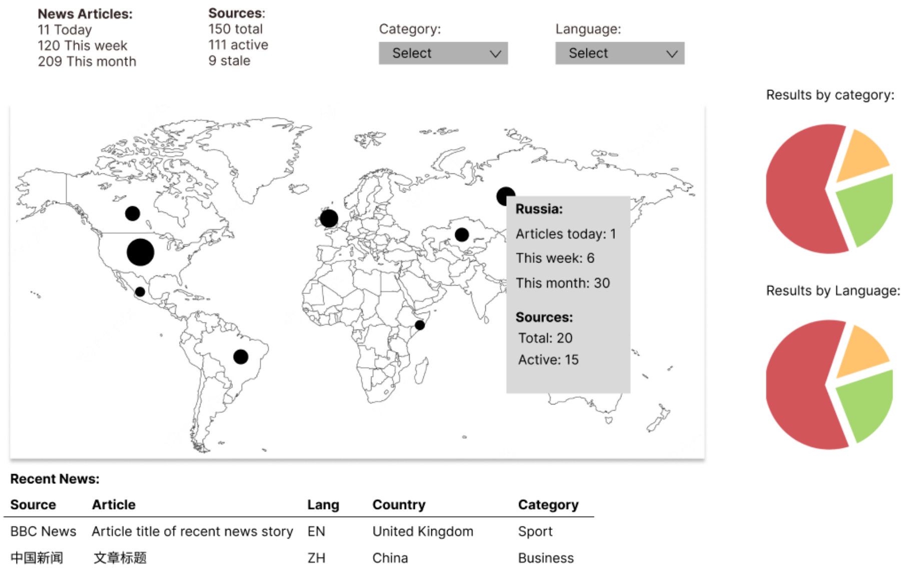

Figure A.1: The wireframe for the proposed new visualisation.

B Wireframe of proposed web interface

Figure B.1: *The wireframe for the homepage of the web interface when the system is enabled. White space has been removed for clarity.*

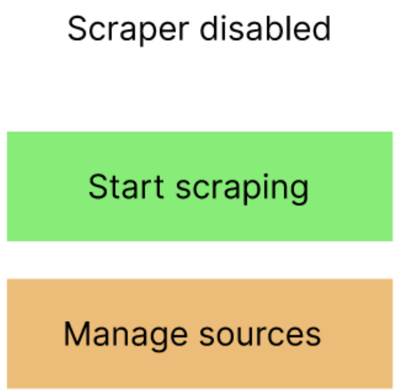

Figure B.2: *The wireframe for the homepage of the web interface when the system is disabled. White space has been removed for clarity.*

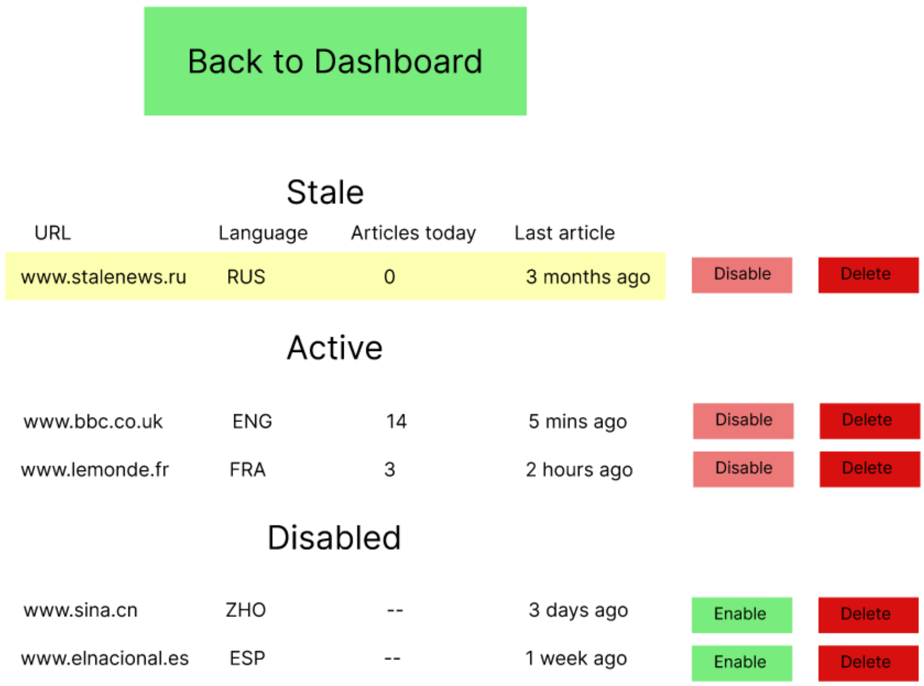

Figure B.3: *The wireframe for the manage sources page of the web interface. White space has been removed for clarity.*

C List of data sources for real-world dataset

Entertainment/Arts:

EN:

http://feeds.bbci.co.uk/news/entertainment_and_arts/rss.xml
https://rss.nytimes.com/services/xml/rss/nyt/Arts.xml
https://rss.nytimes.com/services/xml/rss/nyt/Books.xml
https://rss.nytimes.com/services/xml/rss/nyt/Movies.xml
https://rss.nytimes.com/services/xml/rss/nyt/Music.xml
https://rss.nytimes.com/services/xml/rss/nyt/Television.xml
https://rss.nytimes.com/services/xml/rss/nyt/Theater.xml
https://rss.nytimes.com/services/xml/rss/nyt/Dance.xml
http://rss.cnn.com/rss/edition_entertainment.rss
https://www.theguardian.com/uk/film/rss
https://www.theguardian.com/uk/tv-and-radio/rss
https://www.theguardian.com/books/rss
https://www.theguardian.com/artanddesign/rss
https://www.theguardian.com/stage/rss
https://rss.cbc.ca/lineup/arts.xml
https://globalnews.ca/entertainment/feed/
https://abcnews.go.com/Entertainment
https://www.stuff.co.nz/entertainment
https://www.jamaicaobserver.com/rssfeed/entertainment/
https://www.thenews.com.pk/rss/1/10
https://www.rte.ie/entertainment/

FR:

https://www.lefigaro.fr/rss/figaro_cinema.xml
https://www.lefigaro.fr/rss/figaro_musique.xml
https://www.lefigaro.fr/rss/figaro_livres.xml
https://www.lefigaro.fr/rss/figaro_theatre.xml
https://www.lefigaro.fr/rss/figaro_arts-expositions.xml
https://www.lemonde.fr/en/cinema/rss_full.xml

https://www.lemonde.fr/en/music/rss_full.xml
https://www.lemonde.fr/en/television-radio/rss_full.xml
https://www.lemonde.fr/en/books/rss_full.xml
https://www.lemonde.fr/en/arts/rss_full.xml
https://www.liberation.fr/cinema/
https://www.liberation.fr/arts/
https://www.liberation.fr/musique/
https://www.liberation.fr/livres/
https://www.lexpress.fr/theatre/
https://www.lexpress.fr/musique/
https://www.lexpress.fr/livre/
https://www.lexpress.fr/art/
https://www.lexpress.fr/cinema/
https://www.lapresse.ca/arts/rss
https://www.ledevoir.com/rss/section/culture.xml?id=48
https://lapresse.tn/category/culture/feed/
https://www.europe1.fr/rss/culture.xml
ES:
https://e00-elmundo.uecdn.es/elmundo/rss/television.xml
http://www.abc.es/rss/feeds/abc_Cultura.xml
http://www.abc.es/rss/feeds/abc_Libros.xml
http://www.abc.es/rss/feeds/abc_Teatro.xml
http://www.abc.es/rss/feeds/abc_Arte.xml
https://elpais.com/cultura/
https://elpais.com/television/
https://www.lanacion.com.ar/cultura/
https://www.elnacional.com/entretenimiento/
https://pulsoslp.com.mx/cultura
https://www.jornada.com.mx/rss/cultura.xml
https://www.eltiempo.com/rss/cultura.xml
https://www.elespectador.com/entretenimiento/
PT:
https://www.publico.pt/culturaipsilon
https://www.dn.pt/cultura/
http://g1.globo.com/dynamo/pop-arte/rss2.xml
http://g1.globo.com/dynamo/musica/rss2.xml
https://www.cartacapital.com.br/cultura/

https://expressodasilhas.cv/cultura
ID:
https://www.jawapos.com/entertainment/
https://www.pikiran-rakyat.com/entertainment
https://www.suara.com/rss/entertainment
https://www.tribunnews.com/seleb
https://www.suara.com/entertainment
ZH:
https://ent.sina.com.cn/
https://ent.huanqiu.com/
https://art.huanqiu.com/
https://news.mingpao.com/rss/pns/s00016.xml
https://ent.ltn.com.tw/
https://www.zaobao.com.sg/entertainment/
Sports:
EN:
http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/front_page/rss.xml
http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/football/rss.xml
http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/cricket/rss.xml
http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/rugby_union/rss.xml
http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/rugby_league/rss.xml
http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/golf/rss.xml
http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/boxing/rss.xml
http://newsrss.bbc.co.uk/rss/sportonline_uk_edition/athletics/rss.xml
https://rss.nytimes.com/services/xml/rss/nyt/Sports.xml
http://rss.cnn.com/rss/edition_sport.rss
http://rss.cnn.com/rss/edition_football.rss
http://rss.cnn.com/rss/edition_motorsport.rss
http://rss.cnn.com/rss/edition_tennis.rss
http://rss.cnn.com/rss/edition_golf.rss
https://www.theguardian.com/uk/sport/rss
https://rss.cbc.ca/lineup/sports.xml
https://globalnews.ca/sports/feed/
https://abcnews.go.com/Sports
https://www.stuff.co.nz/sport
https://www.jamaicaobserver.com/rssfeed/sport/
https://www.thenews.com.pk/rss/1/3

https://www.rte.ie/sport/

FR:

https://www.lefigaro.fr/rss/figaro_football.xml

https://www.lefigaro.fr/rss/figaro_football-transfert.xml

https://www.lefigaro.fr/rss/figaro_rugby.xml

https://www.lefigaro.fr/rss/figaro_tennis.xml

https://www.lefigaro.fr/rss/figaro_formule-1.xml

https://www.lefigaro.fr/rss/figaro_basket.xml

https://www.lemonde.fr/en/sports/rss_full.xml

https://www.lemonde.fr/en/football//rss_full.xml

https://www.lemonde.fr/en/rugby/rss_full.xml

https://www.lemonde.fr/en/basket/rss_full.xml

https://www.liberation.fr/sports/

https://www.lapresse.ca/sports/rss

https://www.ledevoir.com/rss/section/sports.xml?id=85

https://lapresse.tn/category/sport/feed/

https://www.europe1.fr/rss/sport.xml

ES:

https://e00-elmundo.uecdn.es/elmundodeporte/rss/portada.xml

https://e00-elmundo.uecdn.es/elmundodeporte/rss/futbol.xml

https://e00-elmundo.uecdn.es/elmundodeporte/rss/baloncesto.xml

https://e00-elmundo.uecdn.es/elmundodeporte/rss/ciclismo.xml

https://e00-elmundo.uecdn.es/elmundodeporte/rss/golf.xml

https://e00-elmundo.uecdn.es/elmundodeporte/rss/tenis.xml

http://www.abc.es/rss/feeds/abc_Deportes.xml

http://www.abc.es/rss/feeds/abc_Futbol.xml

http://www.abc.es/rss/feeds/abc_Baloncesto.xml

http://www.abc.es/rss/feeds/abc_Tenis.xml

http://www.abc.es/rss/feeds/abc_Automovilismo.xml

https://elpais.com/deportes/

https://www.lanacion.com.ar/deportes/

https://www.elnacional.com/deportes/

https://pulsoslp.com.mx/meta

https://www.jornada.com.mx/rss/deportes.xml

https://www.eltiempo.com/rss/deportes.xml

https://www.elespectador.com/deportes/

PT:

https://www.publico.pt/desporto
https://www.dn.pt/desporto/
http://feeds.jn.pt/JN-Desporto
https://feeds.folha.uol.com.br/esporte/rss091.xml
https://expressodasilhas.cv/desporto
ID:
https://bola.kompas.com/
https://sport.republika.co.id/
https://www.jawapos.com/sports/
https://www.jawapos.com/sepak-bola-indonesia/
https://www.pikiran-rakyat.com/olahraga/
https://www.pikiran-rakyat.com/bola/
https://www.suara.com/rss/bola
https://www.tribunnews.com/sport
https://sport.detik.com/
https://www.suara.com/sport
ZH:
https://sports.sina.com.cn/
https://sports.huanqiu.com/
https://news.mingpao.com/rss/pns/s00015.xml
https://sports.ltn.com.tw/
https://www.zaobao.com.sg/sports
http://www.people.com.cn/rss/sports.xml
Politics:
EN:
http://feeds.bbci.co.uk/news/politics/rss.xml
https://rss.nytimes.com/services/xml/rss/nyt/Politics.xml
https://www.statnews.com/category/politics/feed/
https://www.theguardian.com/politics/rss
https://globalnews.ca/politics/feed/
https://abcnews.go.com/Politics
http://www.scoop.co.nz/storyindex/index.rss?s.c=PO
https://www.stuff.co.nz/politics
https://rss.cbc.ca/lineup/politics.xml
https://www.thenews.com.pk/rss/1/8
FR:
https://www.lefigaro.fr/rss/figaro_politique_le-scan.xml

https://www.liberation.fr/politique/
https://www.lexpress.fr/politique/
https://www.lapresse.ca/actualites/politique/rss
https://www.ledevoir.com/rss/section/politique.xml?id=51
https://www.europe1.fr/rss/politique.xml
ES:
https://www.lanacion.com.ar/politica/
https://www.jornada.com.mx/rss/politica.xml
https://www.eltiempo.com/rss/politica.xml
https://www.elespectador.com/politica/
PT:
https://www.publico.pt/politica/
https://www.dn.pt/politica/
https://feeds.folha.uol.com.br/poder/rss091.xml
http://g1.globo.com/dynamo/politica/mensalao/rss2.xml
https://www.cartacapital.com.br/politica
https://expressodasilhas.cv/politica
https://jornalnoticias.co.mz/politica/
ID:
https://www.jawapos.com/politik/
https://www.tribunnews.com/mata-lokal-memilih
https://news.detik.com/pemilu
https://sorotpolitik.kompas.com/
https://kilasparlemen.kompas.com/
ZH:
https://news.ltn.com.tw/politics/
http://www.people.com.cn/rss/politics.xml
http://cpc.people.com.cn/
https://politics.gmw.cn/
Science/Technology:
EN:
http://feeds.bbci.co.uk/news/technology/rss.xml
https://rss.nytimes.com/services/xml/rss/nyt/Technology.xml
https://rss.nytimes.com/services/xml/rss/nyt/PersonalTech.xml
http://rss.cnn.com/rss/edition_technology.rss
https://www.theguardian.com/uk/technology/rss
https://rss.cbc.ca/lineup/technology.xml

https://globalnews.ca/tech/feed/
https://abcnews.go.com/Technology
https://www.thenews.com.pk/rss/1/11
http://www.scoop.co.nz/storyindex/index.rss?s.c=SC
https://www.newscientist.com/subject/technology/feed/
https://www.newscientist.com/subject/physics/feed/
https://www.newscientist.com/subject/space/feed/
FR:
https://www.lefigaro.fr/rss/figaro_secteur_high-tech.xml
https://www.lapresse.ca/affaires/techno/rss
https://www.ledevoir.com/rss/section/societe/science.xml
https://www.europe1.fr/rss/technologies.xml
https://www.europe1.fr/rss/sciences.xml
ES:
https://e00-elmundo.uecdn.es/elmundo/rss/ciencia.xml
http://www.abc.es/rss/feeds/abc_Ciencia.xml
http://www.abc.es/rss/feeds/abc_Tecnologia.xml
https://elpais.com/ciencia/
https://elpais.com/tecnologia/
https://www.lanacion.com.ar/tecnologia/
https://www.lanacion.com.ar/ciencia/
https://www.elnacional.com/ciencia-tecnologia/
https://pulsoslp.com.mx/cienciaytecnologia
https://www.jornada.com.mx/rss/ciencias.xml
https://www.eltiempo.com/rss/tecnosfera.xml
https://www.elespectador.com/tecnologia/
https://www.elespectador.com/ciencia/
PT:
https://www.publico.pt/tecnologia
https://www.dn.pt/ciencia/
https://feeds.folha.uol.com.br/tec/rss091.xml
https://feeds.folha.uol.com.br/ciencia/rss091.xml
http://g1.globo.com/dynamo/tecnologia/rss2.xml
https://www.cartacapital.com.br/tecnologia
ID:
https://tekno.kompas.com/
https://www.kompas.com/sains

https://tekno.republika.co.id/
https://www.suara.com/rss/tekno
https://zetizen.jawapos.com/rss/science
https://zetizen.jawapos.com/rss/techno
https://www.pikiran-rakyat.com/teknologi/
https://www.tribunnews.com/techno
https://www.suara.com/tekno
ZH:
https://tech.sina.com.cn/
https://tech.huanqiu.com/
AR:
https://www.aljazeera.net/tech/
https://www.aljazeera.net/science/
https://alwafd.news/https://www.albayan.ae/technology
https://www.eremnews.com/sciences-technology
Business/Finance:
EN:
http://feeds.bbci.co.uk/news/business/rss.xml
https://rss.nytimes.com/services/xml/rss/nyt/Business.xml
https://rss.nytimes.com/services/xml/rss/nyt/SmallBusiness.xml
https://rss.nytimes.com/services/xml/rss/nyt/Economy.xml
http://rss.cnn.com/rss/money_news_international.rss
https://www.theguardian.com/uk/business/rss
https://www.theguardian.com/uk/money/rss
https://rss.cbc.ca/lineup/business.xml
https://globalnews.ca/money/feed/
https://abcnews.go.com/Business
http://www.scoop.co.nz/storyindex/index.rss?s.c=BU
https://www.stuff.co.nz/business
https://www.jamaicaobserver.com/rssfeed/business/
https://www.thenews.com.pk/rss/2/15
https://www.rte.ie/business/
FR:
https://www.lefigaro.fr/rss/figaro_societes.xml
https://www.lefigaro.fr/rss/figaro_bourse.xml
https://www.lemonde.fr/en/economy/rss_full.xml
https://www.lemonde.fr/en/money-investments/rss_full.xml

https://www.liberation.fr/economie/
https://www.lapresse.ca/affaires/economie/rss
https://www.lapresse.ca/affaires/marches/rss
https://www.lapresse.ca/affaires/entreprises/rss
https://www.ledevoir.com/rss/section/economie.xml?id=49
https://lapresse.tn/category/economie/feed/
https://www.europe1.fr/rss/economie.xml
ES:
https://e00-elmundo.uecdn.es/elmundo/rss/economia.xml
http://www.abc.es/rss/feeds/abc_Economia.xml
https://elpais.com/economia/
https://www.lanacion.com.ar/economia/
https://www.elnacional.com/economia/
https://www.jornada.com.mx/rss/economia.xml
https://www.eltiempo.com/rss/economia.xml
https://www.elespectador.com/economia/
PT:
https://www.publico.pt/economia
https://www.dn.pt/dinheiro/
http://feeds.jn.pt/JN-Economia
https://feeds.folha.uol.com.br/mercado/rss091.xml
http://g1.globo.com/dynamo/economia/rss2.xml
https://www.cartacapital.com.br/economia
https://expressodasilhas.cv/economia
https://jornalnoticias.co.mz/economia/
https://jornalnoticias.co.mz/desporto/
ID:
https://money.kompas.com/
https://ekonomi.republika.co.id/
https://www.jawapos.com/ekonomi/
https://www.pikiran-rakyat.com/ekonomi/
https://www.suara.com/rss/bisnis
https://www.tribunnews.com/bisnis
https://finance.detik.com/
https://www.suara.com/bisnis
ZH:
https://finance.sina.com.cn/

https://finance.huanqiu.com/
https://news.mingpao.com/rss/pns/s00004.xml
https://ec.ltn.com.tw/
https://www.zaobao.com.sg/finance/
http://www.people.com.cn/rss/finance.xml
AR:
https://www.aljazeera.net/ebusiness/
https://alwafd.news/https://www.albayan.ae/economy
https://www.eremnews.com/economy
https://shafaq.com/ar/https://www.okaz.com.sa/economy

Health/Welfare:

EN:
https://rss.nytimes.com/services/xml/rss/nyt/Health.xml
https://www.statnews.com/tag/coronavirus/feed
https://www.statnews.com/category/health/feed
https://www.statnews.com/category/pharma/feed
https://rss.cbc.ca/lineup/health.xml
https://globalnews.ca/health/feed/
https://abcnews.go.com/Health
http://www.scoop.co.nz/storyindex/index.rss?s.c=GE
https://www.stuff.co.nz/health
https://news.un.org/feed/subscribe/en/news/topic/health/feed/rss.xml
https://www.thenews.com.pk/rss/1/12
https://www.rte.ie/health/
https://www.rte.ie/coronavirus/
https://www.ecdc.europa.eu/en/taxonomy/term/2794/feed
https://www.ecdc.europa.eu/en/taxonomy/term/2942/feed
https://www.ecdc.europa.eu/en/taxonomy/term/1310/feed
https://www.ecdc.europa.eu/en/taxonomy/term/1505/feed
https://www.newscientist.com/subject/health/feed/
FR:
https://www.lefigaro.fr/rss/figaro_sante.xml
https://www.lexpress.fr/sante/
https://www.lapresse.ca/actualites/sante/rss
https://www.lapresse.ca/actualites/covid-19/rss
https://www.ledevoir.com/rss/section/societe/sante.xml
https://www.europe1.fr/rss/sante.xml

https://news.un.org/feed/subscribe/fr/news/topic/health/feed/rss.xml
ES:
https://e00-elmundo.uecdn.es/elmundosalud/rss/portada.xml
http://www.abc.es/rss/feeds/abc_SociedadSalud.xml
https://elpais.com/salud-y-bienestar/
https://www.lanacion.com.ar/salud/
https://www.eltiempo.com/rss/salud.xml
https://www.elespectador.com/salud/
https://news.un.org/feed/subscribe/es/news/topic/health/feed/rss.xml
PT:
https://feeds.folha.uol.com.br/equilibrioesaude/rss091.xml
http://g1.globo.com/dynamo/ciencia-e-saude/rss2.xml
https://www.cartacapital.com.br/saude
https://news.un.org/feed/subscribe/pt/news/topic/health/feed/rss.xml
ID:
https://health.kompas.com/
https://www.jawapos.com/bersama-lawan-covid-19/
https://www.jawapos.com/kesehatan/
https://www.suara.com/rss/health
https://www.tribunnews.com/kesehatan
https://health.detik.com/
https://www.suara.com/health
ZH:
https://news.un.org/feed/subscribe/zh/news/topic/health/feed/rss.xml
https://health.huanqiu.com/
https://health.ltn.com.tw/
https://www.zaobao.com.sg/health/
https://www.jkb.com.cn/publicHealth/
http://health.people.com.cn/
https://health.gmw.cn/

D Optimal model hyperparameters

Dataset	nBERT-base			XLM-RoBERTa-base		
	Batch size	**Epochs**	**Learning rate**	**Batch size**	**Epochs**	**Learning rate**
English Data	32	4	5e-05	32	3	2e-05
Multilingual Data	32	3	5e-05	32	4	3e-05
Real-world Data	16	5	3e-05	16	4	2e-05

Table D.1: *Table containing the optimised batch size, number of epochs and learning rate of each model with each dataset after a grid search hyperparameter optimisation*

E JavaScript implementation of adding sources to table

```
function getDateString(date){
  let now = Date.now()
  let then = Date.parse(date)
  // Return never for default value
  if (then <= 946684800000){
    return ["Never", true]
  }

  let msDifference = now - then
  // Calculate if source is stale
  msUntilStale = Math.floor(daysUntilStale * 24 * 60 * 60 * 1000)
  let is_stale = (msUntilStale < msDifference)

  // Get date string
  let mins_difference = Math.floor(msDifference / (1000 * 60))
  if (mins_difference === 0){
    return ["Just now", is_stale]
  } else if (mins_difference === 1){
    return ["1 minute ago", is_stale]
  }
  // Articles scraped in future timezones
  else if (mins_difference < 0){
    return ["Today", is_stale]
  }
  // If less than an hour ago
  else if (mins_difference < 60){
    return [mins_difference + " minutes ago", is_stale]
  }
  // If less than a day ago
  let hours_difference = Math.floor(mins_difference / 60)
  if (hours_difference === 1){
    return ["1 hour ago", is_stale]
  } else if (hours_difference < 24){
    return [hours_difference + " hours ago", is_stale]
  }
  // If less than a week ago
  let days_difference = Math.floor(hours_difference / 24)
  if (days_difference === 1){
    return ["Yesterday", is_stale]
  }
  else if (days_difference < 7){
    return [days_difference + " days ago", is_stale]
  }
  // If less than a month ago
  let weeks_difference = Math.floor(days_difference / 7)
  if (weeks_difference === 1){
    return ["Last week", is_stale]
  }
  else if (days_difference < 30){
    return [weeks_difference + " weeks ago", is_stale]
  }
```

```
  // If less than a year ago
  let months_difference = Math.floor(days_difference / 30)
  if (months_difference === 1){
    return ["Last month", is_stale]
  }
  else if (days_difference < 365){
    return [months_difference + " months ago", is_stale]
  }
  // If a year agp pr more
  return [Math.floor(days_difference / 365) + " years ago", is_stale]
}

// Get the position to insert the row in, to maintain alphabetical order
function getIndex(table_id, new_string){

  var table = document.getElementById(table_id);

  // Adapted from https://stackoverflow.com/a/3065389
  let index = 0;
  for (var i = 1, row; row = table.rows[i]; i++) {
    if (new_string <= row.cells[0].firstChild.innerHTML){
      return i;
    }
 }
 return table.rows.length;
}
```

```
// Adapted from https://www.w3schools.com/jsref/met_table_insertrow.asp
// Adds a row to the table with the information given in the parameters
function addTableRow(source_id, url, name, lang, srcType, last, isEnabled) {

  // Get date string and if source is stale
  date_string_data = getDateString(last)
  date_string = date_string_data[0]
  is_stale = date_string_data[1]

  // Determine correct table
  let table_id = "active-table"
  if (!(isEnabled)){
    table_id = "disabled-table"
  }
  else if(is_stale){
    table_id="stale-table"
  }

  // Default for no language
  if(!lang){
    lang = "--"
  }

  // Get table to add source to, or create it if it doesn't exist
  var table = document.getElementById(table_id);
  if (!table){
    addTable(table_id)
    table = document.getElementById(table_id);
  }
  // Create rows
  var row = table.insertRow(getIndex(table_id, name));
  var srcCell = row.insertCell(0);
  var langCell = row.insertCell(1);
  var srcTypeCell = row.insertCell(2);
  var lastCell = row.insertCell(3);
  var disableCell = row.insertCell(4);
  var deleteCell = row.insertCell(5);

  // Populate rows
  srcCell.innerHTML = `<a href="${url}">${name}</a>`;
  langCell.innerHTML = lang.toUpperCase();
  srcTypeCell.innerHTML = srcType;
  lastCell.innerHTML = date_string;
  lastCell.className = is_stale;
  // Enable / disable and delete buttons
  if (isEnabled){
    disableCell.innerHTML = `<td><button id = "${source_id}-toggle" class="action-btn table-btn disable-btn" onclick="toggleSource(this)">Disable</button></td>`
  } else {
    disableCell.innerHTML = `<td><button id = "${source_id}-toggle" class="action-btn table-btn accent-btn" onclick="toggleSource(this)">Enable</button></td>`
  }
  deleteCell.innerHTML = `<td><button id = "${source_id}-delete" class="action-btn table-btn delete-btn" onclick="deleteSource(this)">Delete</button></td>`
}
```

F Signed ethics checklist

School of Computing Science

University of Glasgow

Ethics checklist form for assessed exercises (at all levels)

This form is only applicable for assessed exercises that use other people ('participants') for the collection of information, typically in getting comments about a system or a system design, or getting information about how a system could be used, or evaluating a working system.

If no other people have been involved in the collection of information, then you do not need to complete this form.

If your evaluation does not comply with any one or more of the points below, please contact the Chair of the School of Computing Science Ethics Committee (matthew.chalmers@glasgow.ac.uk) for advice.

If your evaluation does comply with all the points below, please sign this form and submit it with your assessed work.

---

1. Participants were not exposed to any risks greater than those encountered in their normal working life.

Investigators have a responsibility to protect participants from physical and mental harm during the investigation. The risk of harm must be no greater than in ordinary life. Areas of potential risk that require ethical approval include, but are not limited to, investigations that occur outside usual laboratory areas, or that require participant mobility (e.g. walking, running, use of public transport), unusual or repetitive activity or movement, that use sensory deprivation (e.g. ear plugs or blindfolds), bright or flashing lights, loud or disorienting noises, smell, taste, vibration, or force feedback

2. The experimental materials were paper-based, or comprised software running on standard hardware.

Participants should not be exposed to any risks associated with the use of non-standard equipment: anything other than pen-and-paper, standard PCs, laptops, iPads, mobile phones and common hand-held devices is considered non-standard.

3. All participants explicitly stated that they agreed to take part, and that their data could be used in the project.

If the results of the evaluation are likely to be used beyond the term of the project (for example, the software is to be deployed, or the data is to be published), then signed consent is necessary. A separate consent form should be signed by each participant.

Otherwise, verbal consent is sufficient, and should be explicitly requested in the introductory script.

4. No incentives were offered to the participants.

The payment of participants must not be used to induce them to risk harm beyond that which they risk without payment in their normal lifestyle.

5. No information about the evaluation or materials was intentionally withheld from the participants.

Withholding information or misleading participants is unacceptable if participants are likely to object or show unease when debriefed.

6. No participant was under the age of 16.

Parental consent is required for participants under the age of 16.

7. No participant has an impairment that may limit their understanding or communication.

Additional consent is required for participants with impairments.

8. Neither I nor my supervisor is in a position of authority or influence over any of the participants.

A position of authority or influence over any participant must not be allowed to pressurise participants to take part in, or remain in, any experiment.

9. All participants were informed that they could withdraw at any time.

All participants have the right to withdraw at any time during the investigation. They should be told this in the introductory script.

10. All participants have been informed of my contact details.

All participants must be able to contact the investigator after the investigation. They should be given the details of both student and module co-ordinator or supervisor as part of the debriefing.

11. The evaluation was discussed with all the participants at the end of the session, and all participants had the opportunity to ask questions.

The student must provide the participants with sufficient information in the debriefing to enable them to understand the nature of the investigation. In cases where remote participants may withdraw from the experiment early and it is not possible to debrief them, the fact that doing so will result in their not being debriefed should be mentioned in the introductory text.

12. All the data collected from the participants is stored in an anonymous form.

All participant data (hard-copy and soft-copy) should be stored securely, and in anonymous form.

Course and Assessment Name: COMPSCI402SP: Level 4 Individual Project

Student's Name: Adam Fairlie

Student Number: 2461352

Student's Signature: Adam Fairlie

Date 19/03/2023

G Questionnaire

BioCaster Scraper Dashboard Usability Testing

Background Information:
I am creating a system for BioCaster to collect news articles from multiple sources, using a configurable scraper. The features of this project are:

- News article curation through a web scraping system, which can collect from RSS Feeds and news websites.
- Topic classification of news article category (e.g. health, sport, entertainment) using a ML classifier (in this current implementation, classification of articles is random).
- A database to store news source information (name, url, country) and article information (headline, body, language, category, publish date), and a live visualisation to show the information in this database.
- A web interface for controlling the web scraping system (enabling/disabling the system and managing the sources used).

This questionnaire will evaluate the visualisation and the web interface.
Motivation:
The results of this questionnaire will be used (with consent) in my dissertation evaluation. In particular, I will conduct analyses of the responses (aggregations and qualitative coding) to answer the research questions "Does the visualisation provide a complete and effective overview of the data collected?" and "Does the interface allow users to easily and effectively manage the underlying web scraping system?"
Mock interface:
A mock version of the interface (not connected to the scraping system) can be found at https://www.adamfairlie.com.
This includes a kibana visualisation which is connected to the scraping system, but with random topic classification.
To view the visualisation of this interface, you will have to log in (through kibana) using the details below.
Kibana login:
Username: biocaster
Password: questionnaire2023

Thank you for your participation. If you need to contact the investigator for any reason, including to withdraw your results, please feel free to send an email to **2461352f@student.gla.ac.uk**

Consent information

By ticking the following box, you:

- Confirm that you are over the age of 16.
- Understand that you may withdraw from the questionnaire at any time.
- Understand the nature of the experiment (use in analysis of the interface and visualisation for evaluation section of dissertation).
- Consent to your results to this questionnaire being used **anonymously** as part of the evaluation.
- Understand that you may contact the investigator (at the email address **2461352f@student.gla.ac.uk**) at any time before or after the experiment with any questions, or to withdraw your participation.

Consent *

☐ I understand, and consent to my data being used as part of the evaluation

Task

Open the mock version of the interface: https://www.adamfairlie.com
Try using the interface, particularly the following features:

- Enabling/disabling the scraping system
- Showing and viewing/interacting with the visualisation
- Enabling/disabling sources
- Deleting sources
- Creating a new source

Then answer the following questions:

Please check which of the following actions you were able to perform

- [ ] Toggling the scraping system on/off
- [ ] Showing/hiding the visualisation
- [ ] Adding a new source
- [ ] Enabling a source
- [ ] Disabling a source
- [ ] Deleting a source

Did you encounter any bugs/technical issues while using the website?

Long answer text

Visualisation

Questions on the visualisation of scraper information (the kibana dashboard, showing the world map and other charts/tables)

How **easy to understand** do you feel the visualisation is?

	1	2	3	4	5	
Very difficult	○	○	○	○	○	Very easy

Which parts of the visualisation, if any, were difficult to understand/interpret?

Short answer text

In what ways, if any, do you feel the visualisation could be made easier to understand?

Long answer text

How effectively do you feel the visualisation **represents** the **data collected?**

	1	2	3	4	5	
Not very effectively	○	○	○	○	○	Very effectively

How do you feel the data could be better represented by the visualisation (e.g. do you feel any important statistics/information is missing)?

Long answer text

Which parts of the visualisation, if any, do you feel are effective?

Long answer text

Interface

Questions on the overall web interface for managing the scraper

How would you rate the web interface on how **intuitive** it is?

	1	2	3	4	5	
Very unintuitive	○	○	○	○	○	Very intuitive

Which functions/parts of the interface, if any, felt unintuitive or difficult to use?

Long answer text

How would you rate the web interface on how **clearly** the information is presented?

	1	2	3	4	5	
Very unclear	○	○	○	○	○	Very clear

Which parts of the interface, if any, were unclear or reduced the clarity of information presented?

Long answer text

How would you rate the website on how **aesthetically appealing** it is?

	1	2	3	4	5	
Very unappealing	○	○	○	○	○	Very appealing

Which parts of the web interface, if any, do you feel are effective?

Long answer text

General feedback

Description (optional)

Are there any other general feedback/comments you would like to give?

Long answer text

7 Bibliography

L. Abdulraheem, J. Monehin, A. Akanbi, A. Onajole, and A. Bamgbala. Disease notification among physicians in a nigerian tertiary health institution. *Nigerian Medical Practitioner*, 45(6): 111–115, 2004.

M. A. Al-Masni, D.-H. Kim, and T.-S. Kim. Multiple skin lesions diagnostics via integrated deep convolutional networks for segmentation and classification. *Computer methods and programs in biomedicine*, 190:105351, 2020.

T. Alam, A. Khan, and F. Alam. Bangla text classification using transformers. *arXiv preprint arXiv:2011.04446*, 2020.

X. Chen, P. Cong, and S. Lv. A long-text classification method of chinese news based on bert and cnn. *IEEE Access*, 10:34046–34057, 2022.

Z. Chen, L. J. Zhou, X. Da Li, J. N. Zhang, and W. J. Huo. The lao text classification method based on knn. *Procedia Computer Science*, 166:523–528, 2020.

N. Collier, S. Doan, A. Kawazoe, R. M. Goodwin, M. Conway, Y. Tateno, Q.-H. Ngo, D. Dien, A. Kawtrakul, K. Takeuchi, et al. Biocaster: detecting public health rumors with a web-based text mining system. *Bioinformatics*, 24(24):2940–2941, 2008.

A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*, 2019.

L. Deping, W. Hongjuan, L. Mengyang, and L. Pei. News text classification based on bidirectional encoder representation from transformers. In *2021 International Conference on Artificial Intelligence, Big Data and Algorithms (CAIBDA)*, pages 137–140. IEEE, 2021.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

I. Dilrukshi and K. De Zoysa. Twitter news classification: Theoretical and practical comparison of svm against naive bayes algorithms. In *2013 International Conference on Advances in ICT for Emerging Regions (ICTer)*, pages 278–278. IEEE, 2013.

I. Dilrukshi, K. De Zoysa, and A. Caldera. Twitter news classification using svm. In *2013 8th International Conference on Computer Science & Education*, pages 287–291. IEEE, 2013.

F. Fanny, Y. Muliono, and F. Tanzil. A comparison of text classification methods k-nn, naïve bayes, and support vector machine for news classification. *Jurnal Informatika: Jurnal Pengembangan IT*, 3(2):157–160, 2018.

Z. Haddad, L. Madoff, E. Cohn, J. Olsen, A. Crawley, J. Brownstein, M. Smolinski, J. Shao, M. Pollack, and D. Herrera-Guibert. The epicore project: using innovative surveillance methods to verify outbreaks of emerging infectious diseases. *International Journal of Infectious Diseases*, 45:19, 2016.

L. B. Hutama and D. Suhartono. Indonesian hoax news classification with multilingual transformer model and bertopic. *Informatica*, 46(8), 2022.

D. Kakwani, A. Kunchukuttan, S. Golla, N. Gokul, A. Bhattacharyya, M. M. Khapra, and P. Kumar. Indicnlpsuite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for indian languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4948–4961, 2020.

P. Koehn and R. Knowles. Six challenges for neural machine translation. *arXiv preprint arXiv:1706.03872*, 2017.

P. Kostkova, F. Saigí-Rubió, H. Eguia, D. Borbolla, M. Verschuuren, C. Hamilton, N. Azzopardi-Muscat, and D. Novillo-Ortiz. Data and digital solutions to support surveillance strategies in the context of the covid-19 pandemic. *Frontiers in Digital Health*, 3:707902, 2021.

T. Kudo and J. Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018.

S. Kumar, A. Sharma, B. K. Reddy, S. Sachan, V. Jain, and J. Singh. An intelligent model based on integrated inverse document frequency and multinomial naive bayes for current affairs news categorisation. *International journal of system assurance engineering and management*, 13(3): 1341–1355, 2022.

W. Lan, Y. Chen, W. Xu, and A. Ritter. An empirical study of pre-trained transformers for arabic information extraction. *arXiv preprint arXiv:2004.14519*, 2020.

B. Li, Y. He, and W. Xu. Cross-lingual named entity recognition using parallel corpus: A new approach using xlm-roberta alignment. *arXiv preprint arXiv:2101.11112*, 2021.

D. Y. Liliana, A. Hardianto, and M. Ridok. Indonesian news classification using support vector machine. *World Academy of Science, Engineering and Technology*, 57:767–770, 2011.

D. Liparas, Y. HaCohen-Kerner, A. Moumtzidou, S. Vrochidis, and I. Kompatsiaris. News articles classification using random forests and weighted multimodal features. In *Multidisciplinary Information Retrieval: 7th Information Retrieval Facility Conference, IRFC 2014, Copenhagen, Denmark, November 10-12, 2014, Proceedings 7*, pages 63–75. Springer, 2014.

Z. Meng, A. Okhmatovskaia, M. Polleri, Y. Shen, G. Powell, Z. Fu, I. Ganser, M. Zhang, N. B. King, D. Buckeridge, et al. Biocaster in 2021: automatic disease outbreaks detection from global news media. *Bioinformatics*, 38(18):4446–4448, 2022.

K. Mujahid, S. Bhatti, and M. Memon. Classification of urdu headline news using bidirectional encoder representation from transformer and traditional machine learning algorithm. In *2021 6th International Multi-Topic ICT Conference (IMTIC)*, pages 1–5. IEEE, 2021.

Office for National Statistics. Population density - census maps, ons, 2022. URL `https://www.ons.gov.uk/census/maps/choropleth/population/population-density/population-density/persons-per-square-kilometre`.

Our World in Data. Population density, 2022, 2022. URL `https://ourworldindata.org/grapher/population-density`.

R. Pappagari, P. Zelasko, J. Villalba, Y. Carmiel, and N. Dehak. Hierarchical transformers for long document classification. In *2019 IEEE automatic speech recognition and understanding workshop (ASRU)*, pages 838–844. IEEE, 2019.

T. Pires, E. Schlinger, and D. Garrette. How multilingual is multilingual bert? *arXiv preprint arXiv:1906.01502*, 2019.

G. Qiang. An effective algorithm for improving the performance of naïve bayes for text classification. In *2010 Second international conference on computer research and development*, pages 699–701. IEEE, 2010.

T. Ranasinghe and M. Zampieri. Multilingual offensive language identification with cross-lingual embeddings. *arXiv preprint arXiv:2010.05324*, 2020.

D.-W. Seo and S.-Y. Shin. Methods using social media and search queries to predict infectious disease outbreaks. *Healthcare informatics research*, 23(4):343–348, 2017.

G. Septian, A. Susanto, and G. F. Shidik. Indonesian news classification based on nabana. In *2017 International Seminar on Application for Technology of Information and Communication (iSemantic)*, pages 175–180. IEEE, 2017.

T. B. Shahi and A. K. Pant. Nepali news classification using naive bayes, support vector machines and neural networks. In *2018 International Conference on Communication Information and Computing Technology (ICCICT)*, pages 1–5. IEEE, 2018.

G. Stanovsky, N. A. Smith, and L. Zettlemoyer. Evaluating gender bias in machine translation. *arXiv preprint arXiv:1906.00591*, 2019.

R. Steinberger, F. Fuart, B. Pouliquen, and E. van der Goot. Medisys: a multilingual media monitoring tool for medical intelligence and earlywarning. In *International Disaster and Risk Conference*, pages 25–29, 2008.

B. Swaminathan, T. J. Barrett, S. B. Hunter, R. V. Tauxe, and C. P. T. Force. Pulsenet: the molecular subtyping network for foodborne bacterial disease surveillance, united states. *Emerging infectious diseases*, 7(3):382, 2001.

S. Ting, W. Ip, A. H. Tsang, et al. Is naive bayes a good classifier for document classification. *International Journal of Software Engineering and Its Applications*, 5(3):37–46, 2011.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

A. Villanova and O. Abdulmatin. valurank/news$_articles_categorization, 2022. URL$.

I. Vogel and M. Meghana. Detecting fake news spreaders on twitter from a multilingual perspective. In *2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 599–606. IEEE, 2020.

S. Wu and M. Dredze. Are all languages created equal in multilingual bert? *arXiv preprint arXiv:2005.09093*, 2020.