Tracking Brand Sentiment via Public News
— Final Report —

Matthew Collins
Max Greenwood
Mario Lavina
Louis Manestar
Maksym Tymchenko
Charlize Yang

{matthew.collins21
max.greenwood21
mario.lavina21
louis.manestar18
maksym.tymchenko15
yushi.yang19}
@imperial.ac.uk

Supervisor: Professor Alessandra Russo
Course: COMP70048, Imperial College London

May 3rd, 2022

1 Introduction

1.1 Background

Reputation management is a complex issue in the corporate world. The way a person feels about a brand depends on all the means and contexts in which they encounter it: through interactions with its products and employees, direct commercial messaging, and absorbing what other people think about them. But while it is nebulous and not easily quantifiable, reputation is certainly not taken lightly by business directors. In a recent survey, corporate reputation is imagined to prop up an average of 63% of company market value in the minds of 100 global executives [12]. As such, it features highly on many corporate agendas; in a 2014 study, 87% of executives rated reputation risk as the most significant strategic business risk [3].

However, reputation is precarious: it can take "20 years to build, five minutes to ruin" [2]. Influential companies are under constant scrutiny and flurries of online media activity can amplify negative sentiment in the public conscience quickly. Disapproval is never welcome, but it can provide organisations with important learning opportunities and a temporary platform to respond to the controversy. Indeed, 58% of global executives from a 2020 study identified how a company addresses crises and issues it faces as a contributing factor to its reputation: the most significant factor relating to marketing and communications [12].

With the importance of brand reputation and the potential risks posed by media commentators established, two questions arise: how do companies stay in touch with what is being said about them, and how can they react to it? Leveraging big data and artificial intelligence to identify trends and associations in public discourse is likely to be a key development area in corporate reputation management. With deeper insight, companies can more confidently shape the messages they want to send to the public. One of the most direct ways of reacting to public opinion is to communicate to them explicitly through an advertising campaign. These campaigns are increasingly being focused online, with a predicted 60% of global ad spend given to digital channels in 2022 [8]. Digital advertising is the area in which The Trade Desk operates.

1.2 The Trade Desk

The Trade Desk (TTD) is a multi-national media buying organisation that competes to purchase digital advertisement space and optimise online campaigns for their clients. They process over 13 million auction requests per second, serve over 2 billion ads per day and collect over 1 billion user events such as clicks, conversions and video playing per day. They are also the world's second-biggest user of Amazon's **S3** storage service, given the vast amount of data they deal with.[6]

TTD procures digital ad space through Real-Time Bidding (RTB): an auction where website owners, also known as publishers, sell ad impressions to advertisers. Auctions are initiated whenever a user visits a publisher's website and are completed in milliseconds before the page loads. Figure 1 displays the RTB ecosystem and the actors involved. TTD plays two key roles in the online display advertising market: they are a demand-side platform (DSP) that connects their clients to the digital market, as well as a data management platform (DMP) that stores information about users' online activities, habits, interests and demographics.

TTD is actively exploring new ways to adopt AI to provide advertisers with insights. Tapping into interests around brand reputation and the portrayal of corporations in the media has been identified as a prospective venture. As a product concept, TTD have envisaged a software that will allow users to track and investigate the sentiments attached to brands mentioned in online articles. Dr. Jiefei Ma, Head of AI at TTD, and data scientist Christopher Hawkes worked with us to kickstart this new product.

1.3 Project Specifications

We were asked to build a minimum viable product (MVP) to track the sentiment attached to brands in public news over time. A database containing time-stamped sentiment information will be updated

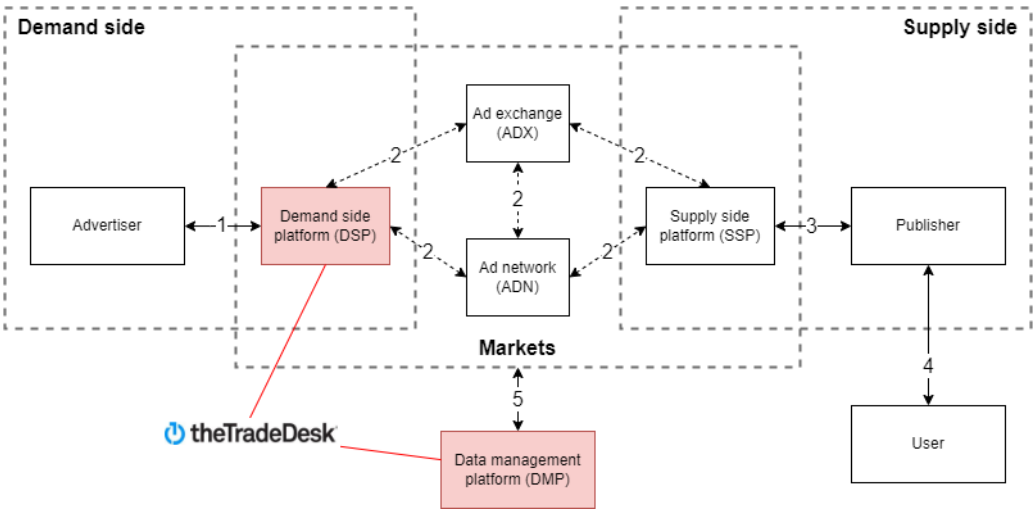

Figure 1: Summary of RTB ecosystem adapted from Wang, J. et al.: "1. Advertiser creates campaigns in the market. 2. The market trades campaigns and impressions to balance the demand and supply for better efficiency. 3. The publisher registers impressions with the market. 4. The user issues queries or visits webpages. 5. The markets can query data management platform user profiles in real-time."[14]

daily, facilitated by an automated software pipeline deployed at scale. We had three specific criteria for the product and one extension deliverable:

1. Cloud-based data processing and NLP pipelines for brand news article identification and sentiment extraction;
2. A repository for time series sentiment data;
3. An interactive dashboard for sentiment tracking of brands over time.
4. Extension: An open-source repository for extracted news articles.

In this report, we explain how we successfully delivered the MVP and the extension to specification and how well the product fulfils its function. In Section 2, we outline the final architecture of the deployed software pipeline before elaborating on the development of its three main components: data extraction, NLP modelling and visualisation. We also discuss the cloud services we used to deploy it at scale. In Section 3, we reflect on the qualities and limitations of the MVP and suggest how it might be adapted in the future. Prior to these sections, we briefly overview some of the key resources and technologies that enabled our development.

1.4 Resources and Technology

Data Public news was obtained from the Common Crawl News Dataset (CC-NEWS) [1], a news-specific archive of web crawl data. Containing terabytes of data, CC-NEWS is updated 15-20 times every day, with around 25,000 articles each update, in a Web Archive (WARC) format. While articles appear in 55 different languages, for the MVP we restricted ourselves to processing only English articles.

Natural language models For the tasks of brand identification and sentiment analysis, we utilised off-the-shelf models from John Snow Labs: a popular Natural Language Processing (NLP) for healthcare company. Their open-source Spark NLP libraries – built on Apache Spark and the Spark ML library – include pre-trained models and can run jobs on clusters in parallel and at industry scale.

Cloud resources In order to process the large amounts of data required for a useful MVP, all our software was prepared and containerised for deployment on **Amazon Web Services (AWS)**.

AWS incorporates a suite of over 200 cloud-computing products and services (highlighted in bold throughout), including data storage (**S3**), compute engines (**Fargate and EC2**), resource management software (**AWS Batch**) and business intelligence tools (**Quicksight**). We used Docker to build images that run our software on computing instances rented from AWS, each allocated a specific number of virtual CPUs (vCPUs) and memory. Our budget for the project was $1000.

Programming languages The software we built was coded entirely in Python. A small amount of SQL was also required to query our output database for visualisation.

2 Project Design and Decisions

`https://github.com/Brand-Sentiment-Tracking`

2.1 Project Architecture

Figure 2 displays the high-level architecture of our final software. There are three broad stages to the deployment pipeline:

1. Data extraction and processing of articles from raw HTML into a format suitable for modelling.
2. Machine learning (ML) modelling to identify brands and sentiment from the processed article.
3. Visualisation of our model output on an interactive dashboard for business insight.

We started with the CC-NEWS archive hosted by AWS on **S3**. Running our extraction tool on AWS servers, we requested articles based on the date they were scrapped before processing them with relevant Python libraries. The features of each article were appended to a tabular Parquet file stored in a public S3 bucket. From this bucket, we requested the processed articles based on the extraction date and passed them to two NLP models: brand identification and sentiment analysis, run in sequence as part of a Python package deployed on AWS servers. The output of these models was appended to the article information and written to another Parquet file stored in a final S3 bucket. This database was connected to the interactive dashboard that allows users to interpret the data.

In this section, we structure our discussion of the project design around the three stages of the pipeline mentioned above, plus an additional segment focusing on decisions relating to AWS platforms and application scaling to the cloud. For each stage, we discuss how we developed our software packages, the technology and libraries used to execute the tasks and our key design decisions. Figures 3 and 4 depict the class diagrams describing our software architecture and the stored data features at each stage of the process.

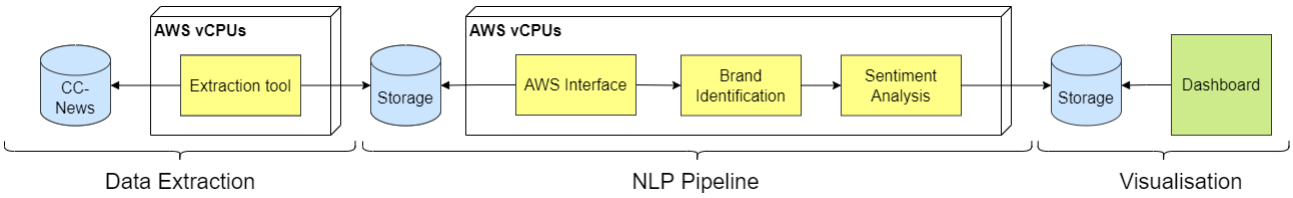

Figure 2: Visual overview of our software architecture.

2.2 Data extraction

`https://github.com/Brand-Sentiment-Tracking/prod-batch-extractor`

The data extraction tool was designed to pull web crawl data from a Common Crawl repository and transform it into a useful structure for downstream modelling tasks. Common Crawl is a non-profit organisation dedicated to building an archive of web pages by crawling the internet. They

maintain two archives hosted by AWS on **S3**: CC-MAIN, which contains records from all kinds of web requests/responses; and CC-NEWS, which is made up of the web requests for, and the responses containing, news articles, typically as `text/html`. Each record is stored as a Web ARChive (WARC) file, containing the headers, payload and metadata about the crawl. Each WARC file is approximately 1GB in size when compressed, holding up to 60,000 records.

There are two possible approaches to extract news articles from Common Crawl. The first is to use the Common Crawl Index API (CDX) to query CC-MAIN and extract articles from a discrete set of reputable news sites. This method also returns the WARC URL and the byte index of each record that matched the query, meaning only relevant data needs to be downloaded. The second is to use a package such as `news-please` [4] to download and extract articles from CC-NEWS yielding a large pool of downloaded data to either process or discard.

While we opted for the second approach, both were tested in the course of our development. Initially, we anticipated that using a specified set of reputable news websites would be advantageous for the MVP, given the potential to immediately filter out many thousands of websites that would not provide relevant data for the tracker and avoid downloading excess WARC files. With this in mind, a custom CC-MAIN WARC loader seemed like the best option; we created a package that would request articles through the CDX API. However, discussions with TTD persuaded us that the CC-NEWS approach would benefit other groups performing modelling tasks involving a large corpus of news. The extension task of creating an open-source, multilingual repository of public news was achieved by building a 'greedy' extraction tool to download and process a complete set of daily articles from the CC-NEWS repository.

Due to a change to how WARC URLs in CC-NEWS are retrieved, using the dedicated CC-NEWS extraction package `news-please` proved unreliable. Instead, we developed our own versatile lightweight version summarised in Figure 3, drawing on the same HTML extraction package, `newspaper3k`. We found `newspaper3k`'s language detection predicts nearly every article to be written in English regardless of its actual language. Despite this error, `newspaper3k` is still able to effectively extract the article. Utilising `langdetect`, a python implementation of Google's Java language detection algorithm [10], gave a more accurate prediction from the extracted title and text.

On implementing our version of `news-please`, we observed slow article extraction using serial processing; it took about 48 hours to extract a single day's worth of articles. As a CPU intensive task, we leverage Python's `multiprocessing` library to extract articles from multiple WARC files at once by submitting jobs to a process pool. Multiprocessing also facilitates running the extraction at scale on AWS by requesting larger instances with a greater number of vCPUs. While JSON files were originally thought to be an appropriate file format for extracted article data, discussions with TTD prompted a switch to Parquet files – a compressed tabular format – for two reasons:

- JSON documents are not very space-efficient, while Parquet files are, especially when using snappy compression.
- Using Parquet files allows us to partition the extracted articles into different categories, automatically filtering them by language and date crawled. Therefore, the pipeline does not need to filter before the ML modelling stage. Instead, it simply downloads articles from the relevant partitions.

Our final extraction program is able to extract a day's worth of online news (0.5M - 1.5M articles) in approximately 2-3 hours when deployed to a 16 vCPU instance on AWS.

2.3 NLP model pipeline

`https://github.com/Brand-Sentiment-Tracking/prod-sentiment-package-ec2`

NLP models underlie the core functionality of the MVP: they provide the brand sentiment data that gives the product value to the end-user. However, this data is not useful in small amounts; we therefore use Apache Spark, a large open-source analytics engine, to facilitate modelling in bulk. One major advantage of Spark is that its inbuilt dataframes can be manipulated through User-Defined

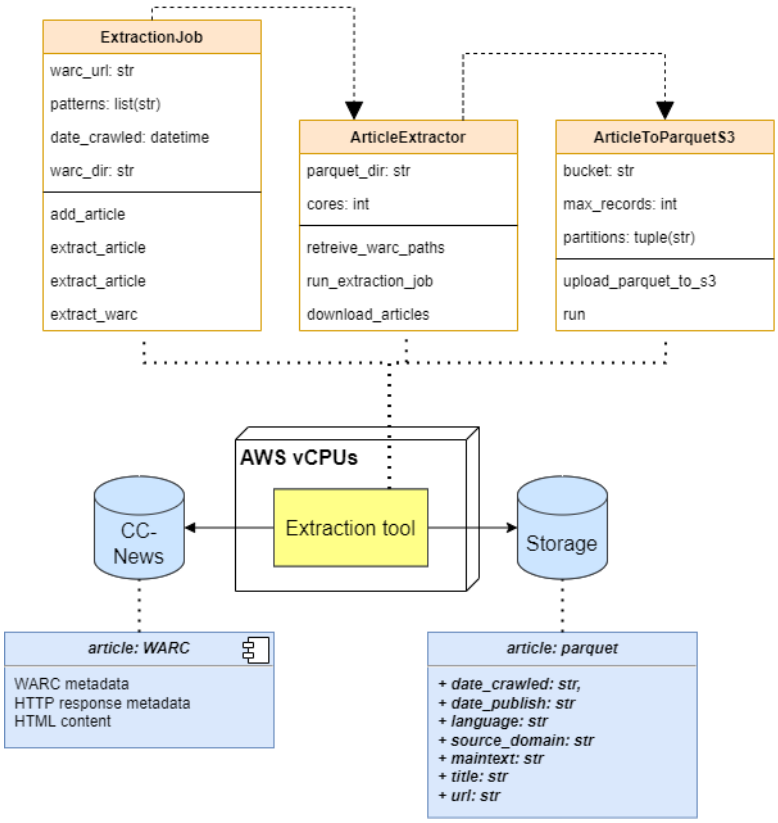

Figure 3: The data extraction package labelled with class diagrams and key data features.

Functions, allowing row operations to be parallelised over clusters of vCPUs to speed up modelling tasks. Ordinary Pandas dataframes do not have this feature. The John Snow curated Spark NLP library – as the only NLP library built natively on Spark – was the natural choice for experimenting with models to perform our tasks.

As depicted in Figure 4, our final NLP modelling program is split into three modules: AWS Interface, Brand Identification, and Sentiment Analysis. The AWS interface acts as a selection filter that reads a specified range of articles into a Spark dataframe based on extraction date. After substituting missing values for published dates with the extraction date, the data is ready for NLP treatment: the remainder of this section focuses on the brand identification and sentiment analysis tasks.

2.3.1 Choice of text

We decided to use only the headlines of articles for brand identification and sentiment analysis. While it might be argued that this negates valuable data, using the article's main text introduces certain problems. Firstly, the main text can be difficult to extract from the HTML; the generic parser from `newspaper3k` in the extraction program struggles with certain layouts of websites. Often only partial or irrelevant text is extracted from a WARC file.

Secondly, dealing with multiple references to the same entities within one article would become an issue. It is improbable that the same entity is mentioned more than once in a headline, so it is easy to extract all the entities from a headline and treat them as independent data points. However, in the body of an article, an entity may be mentioned many times, introducing more involved contextual considerations and the need to aggregate sentiment over the article.

Using only headlines for the MVP increases the likelihood of quality data points and makes the sentiment analysis more computationally efficient by avoiding additional data wrangling requirements. However, we have ensured that it is feasible to work with the main text of articles in future iterations of the product; the input to the NLP model pipeline includes the main text as a feature as depicted in Figure 4.

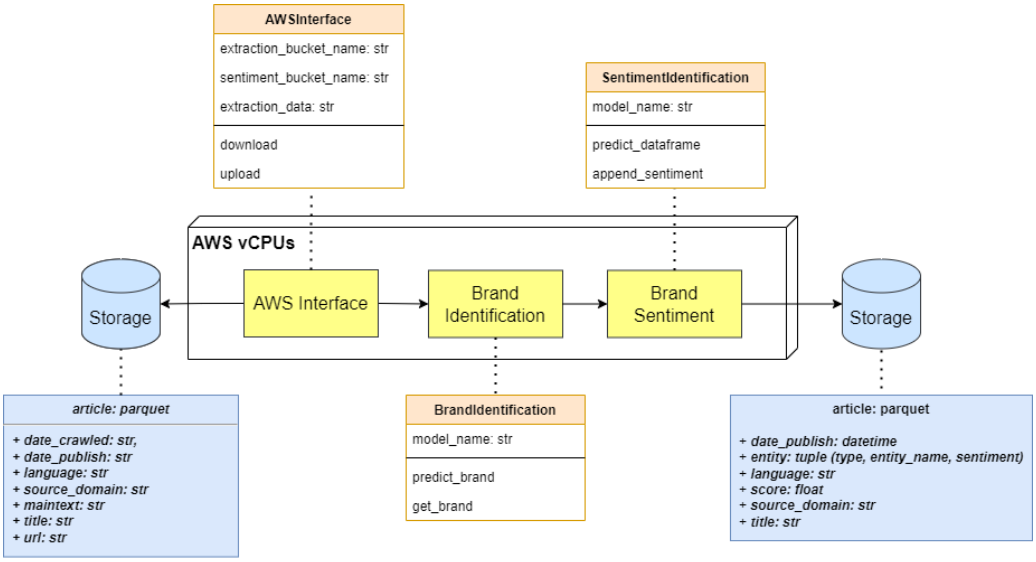

Figure 4: The NLP model pipeline labelled with class diagrams and key data features.

2.3.2 Brand identification

The Brand Identification class in Figure 4 encompasses the methods written to detect the entities in our articles using Name Entity Recognition (NER). The class uses Spark NLP to create a Pipeline object defined as a set of sequential stages: a tokenizer, a pre-trained sentence embedding stage and a pre-trained NER Transformer. Defining a Pipeline object allows us to control the different stages of the pipeline and, more importantly, the structure and format of the output dataframe. When the class is initialized, Spark downloads the pre-trained stages and assembles them in a single object to perform NER predictions on a Spark dataframe with news headlines. For each headline, the output gives a list of tuples in the format of `(entity_type, entity_name)`.

Although TTD is primarily interested in detecting brands for our MVP, to give the package a wider scope the NER model identifies all the entities in the text and categorises them into four types: Organisation (ORG), Person (PER), Location (LOC), and Miscellaneous (MISC).

Model selection We evaluated five pre-trained NER models from John Snow Labs and compared their performance based on F1 scores and running times. These include four state-of-the-art transformer-based models (XLNet, RoBERTa, BERT, DistilBERT) and one non-transformer deep learning model (Char CNNs - BiLSTM - CRF). While XLNet and RoBERTa were trained with a more complex network structure and an improved training methodology compared to a standard BERT model, DistilBERT improves inference speed with only half of the parameters while retaining BERT performance [5]. The five pre-trained models were specifically selected such that they are all trained and tuned on the same dataset of CoNLL-2003: a benchmark dataset for NER consisting of 22,137 news sentences from the Reuters RCV1 corpus between August 1996 and August 1997.[11]

As shown in Table 1, all five pre-trained models achieved similar performance on the CoNLL-2003 test set (left out in training) despite their architectural differences, with the XLNet and BERT models performing slightly better than the other models in identifying organisations. The F1 scores for 'I-PER' are consistently higher than 'B-PER' across all models, suggesting that while the models are good at identifying the initial token of a person's name (the first name), they are slightly worse at recognising the full name.

We also randomly selected and annotated 200 headlines from our own dataset and used 100 of them as a validation set to compare NER model performance. The other half was used as a separate test set to evaluate the final NER model in Section 3.1.1. Table 2 summarises the entity-level F1 scores for each entity type except 'MISC'. All F1 scores are calculated as a weighted sum of precision and recall, where the precision is the percentage of entities found that were correct, and the recall

Model	All	**B-ORG**	**I-ORG**	B-PER	I-PER	B-LOC	I-LOC
ner_conll_bert_base_cased	72	87	89	77	84	82	82
xlnet_base	71	87	87	77	83	82	81
ner_conll_roberta_base	71	85	87	76	83	80	80
ner_dl	70	86	84	76	83	82	77
ner_conll_distilbert_base_cased	70	84	86	75	83	80	77

Table 1: Token-level F1 scores (%) of NER models evaluated on the CoNLL-2003 test set. 'All' gives the F1 score for all tokens. 'B-ORG' refers to the beginning token of an organisation name; 'I-ORG' refers to the 'inside' tokens. For example, 'The Trade Desk' would be labelled as a [B-ORG, I-ORG, I-ORG] sequence. PER and LOC suffixes correspond to person and location entities respectively.

is the percentage of entities in the labelled set that were found. As shown in Table 2, the XLNet model gives the highest F1 scores across almost all entity types, followed by RoBERTa and BERT. In particular, XLNet has a significantly higher F1 score for identifying organisations, meaning it is more likely to identify brand names in news headlines correctly.

Model	F1(All)	**F1(ORG)**	F1(PER)	F1(LOC)
xlnet_base	60	61	72	50
ner_conll_roberta_base	53	55	62	47
ner_conll_bert_base_cased	51	47	68	52
ner_dl	45	38	58	52
ner_conll_distilbert_base_cased	38	33	42	47

Table 2: Entity-level F1 scores (%) of NER models evaluated on self-labelled data (100 headlines).

The running times of models when processing a large amount of data were also compared and used as a criterion for model selection. Table 3 shows the time to download the pre-trained NER models and apply them on a subset of headlines released on March 8, 2022. Using the running time for BERT as a benchmark, the non-transformer model is the fastest with 37% less time taken, while XLNet and DistilBERT took 30% and 40% more time. We chose the 'xlnet_base' pipeline as our final model given that it exhibits the best performance on both datasets and has a relatively fast processing speed.

Model	**Download + Processing Time (s)**	Proportional Increase wrt BERT
ner_dl	43.9	0.63
ner_conll_bert_base_cased	69.4	1
xlnet_base	90.2	1.30
ner_conll_distilbert_base_cased	96.8	1.40
ner_conll_roberta_base	207.4	2.99

Table 3: Average running times (in seconds) for NER models on 16,736 headlines.

Pre-trained models vs training from scratch While John Snow Labs offers functionalities to train a NER model from scratch, we did not do so for two reasons: Firstly, manually labelling data for NER is time-consuming. Training a NER model in SparkNLP requires labelling the part-of-speech (POS) tags and the entity type for each token, which is prone to error and not feasible within the project's time constraint. Secondly, the pre-trained models are trained on a sizeable news-based dataset (CoNLL-2003) with over 14k sentences. Even after concatenating a limited number of labelled data to the CoNLL-2003 dataset to train the model, this would not be expected to yield significantly better results than the well-tuned pre-trained models.

2.3.3 Sentiment analysis

The Sentiment Analysis class in Figure 4 takes input the output dataframe from Brand Identification and predicts the sentiment for each headline with corresponding confidence scores. It then appends the predicted sentiment to all the tuples of detected entities and outputs in the form `(entity_type, entity_name, sentiment)`. In addition, we record a basic numeric `score` feature for each datapoint: this measure is calculated using the probability distribution over the sentiment class predictions. Specifically, we subtract the 'negative' probability from the 'positive' probability to reflect the model confidence toward one sentiment or the other.

For sentiment prediction, we compared several Spark NLP pre-trained models. To choose the best model, we compared the performance of different models on the FinancialPhraseBank dataset [7] downloaded from Kaggle [13], before we had access to data from CC-NEWS. This dataset consists of 4,840 datapoints with two columns, 'Sentiment' and 'News Headline', where the sentiment can be 'negative', 'neutral' or 'positive'. The performances of different models on this dataset are shown in Table 4.

Model	Acc	F1(neg)	F1(pos)	F1(neu)
classifierdl_bertwiki_finance_sentiment	90	89	85	93
bert_sequence_classifier_finbert	88	88	85	90
analyze_sentimentdl_glove_imdb	26	35	37	5
analyze_sentimentdl_use_imdb	29	27	44	1
analyze_sentimentdl_use_twitter	29	38	41	5

Table 4: Metrics (%) of sentiment models on the FinancialPhraseBank dataset [7].

Note that the first two models in Table 4 predict three labels: 'positive', 'negative' and 'neutral', whereas the last three predict mostly 'positive' and 'negative' labels. They only output a 'neutral' label when the confidence score for both the 'positive' and 'negative' labels is less than 0.6, causing low neutral F1 scores for the last three models. Moreover, these models performed worse than or close to random (33%) as the majority of data points in the datasets have the actual label 'neutral', but these models tend to mislabel them as 'positive' or 'negative'. Even after repeating the evaluation with all the 'neutral' labels removed from the dataset, the performance of the first two models was still superior to the last three models.

The 'classifierdl_bertwiki_finance_sentiment' model was chosen for deployment. It produced the highest accuracy (90%) on the FinancialPhraseBank dataset (Table 4) and processes data approximately six times faster than the second-best option, 'bert_sequence_classifier_finbert'.

Testing on our own labelled data We also annotated 1,000 of our extracted articles with sentiment labels. While we left out half of them for final evaluation model in Section 3.1.1, 500 of them (65 negative, 385 neutral and 50 positive) were used to compare the predictions of our chosen model, 'classifierdl_bertwiki_finance_sentiment', as well as of other models. This resulted in the metrics shown in Table 5.

Model	Acc	F1(neg)	F1(neu)	F1(pos)
classifierdl_bertwiki_finance_sentiment	76	41	86	40
bert_sequence_classifier_finbert	75	49	85	25
analyze_sentimentdl_glove_imdb	19	32	6	22
analyze_sentimentdl_use_imdb	16	40	0	19
analyze_sentimentdl_use_twitter	19	34	6	20

Table 5: Metrics (%) of sentiment models on the self-labelled data (500 headlines).

Table 5 confirms that the 'classifierdl_bertwiki_finance_sentiment' model has the best performance on our dataset if compared to similar models available on John Snow labs. This is because it exhibits

the highest accuracy over all other models and the highest neutral and positive F1 scores over the model 'bert_sequence_classifier_finbert'. The discrepancy in performance with the FinancialPhraseBank dataset stems from the fact that the 500 randomly extracted news headlines span the full spectrum of current events, rather than financial events alone on which the model was fine-tuned.

Training our own sentiment model To challenge the performance of the pre-trained sentiment model, we attempted to train a Spark NLP model on our own labelled data by defining a model architecture consisting of a state-of-the-art pre-trained BERT sentence embedder 'sent_bert_use_cmlm_en_base' followed by a deep learning model defined using the ClassifierDLApproach object. We used our manually labelled dataset consisting of 500 datapoints to train on data specific to our application. To achieve a compromise between a balanced dataset and a large enough number of points, we downsampled the number of neutral points to 115 (50% of the downsampled dataset). The data was split into 70% for training, 15% for validation and 15% for testing while preserving the same ratio of positive, negative and neutral points in each set. After optimizing the hyperparameters, the trained model achieves a 68% accuracy on the test set, where our pre-trained pipeline achieves 71% on the same test set. This is a very promising result considering the limited number of labelled datapoints that were utilised. Therefore, manually labelling a larger number of points and training on that has the potential to achieve higher performance and should be considered in future iterations of the software.

2.4 Visualisation

The dashboard component of the MVP communicates the information within the sentiment time-series repository populated after the NLP modelling stage. We use the business insight (BI) platform, **Amazon Quicksight**, to create analyses and publish them to a dashboard.

We produced two graph templates to centre our analysis around. The first is a stacked bar chart of article counts for a particular brand over a range of dates, grouped by sentiment. The second is a line graph displaying the average sentiment score for one or multiple brands over a range of dates. From these graphs, clients can swiftly get a sense of the volume of articles associated with their brand over a given period, how historical events have been portrayed by the media and how brand sentiment has evolved over time. In the two case studies below, we demonstrate how our dashboard captures real-world events and provides insight relevant to reputation management.

Case study: Tesla At time of writing, the last month has seen a good deal of media activity relating to Tesla, the electric vehicle company, and it's CEO, Elon Musk. Figure 5 displays the two aforementioned graphs when filtered on the organisation 'Tesla'. Analysing the contributions to these graphs on the interactive dashboard give us confidence that our software successfully extracts a representative sample of online English-language news and interprets the sentiment of headlines adequately. Three notable features stood out from these graphs:

- Our database contains 916 neutral Tesla related articles published on the 14th April. This spike corresponds Elon Musk's offer to buy Twitter, a story that received a lot of press but without much sentiment attached. Interestingly, the increase in negative articles on the same day were unrelated, instead mostly referring to a racial discrimination lawsuit.
- The sentiment score graph shows a peak in positivity on the 21st April. Articles from this time highlight Tesla's strong Q1 revenue results.
- The negative article increase in the record count graph and sharp decline in sentiment score on the 27th April correspond to the $126 billion loss Tesla's share price took after Elon Musk followed through on his offer to buy Twitter.

Case study: P&O Ferries To explore how our software might be used in a reputation management context, we investigate a company that has recently experienced a public scandal through the lens of our dashboard. P&O Ferries was condemned for laying off 800 employees without warning via a

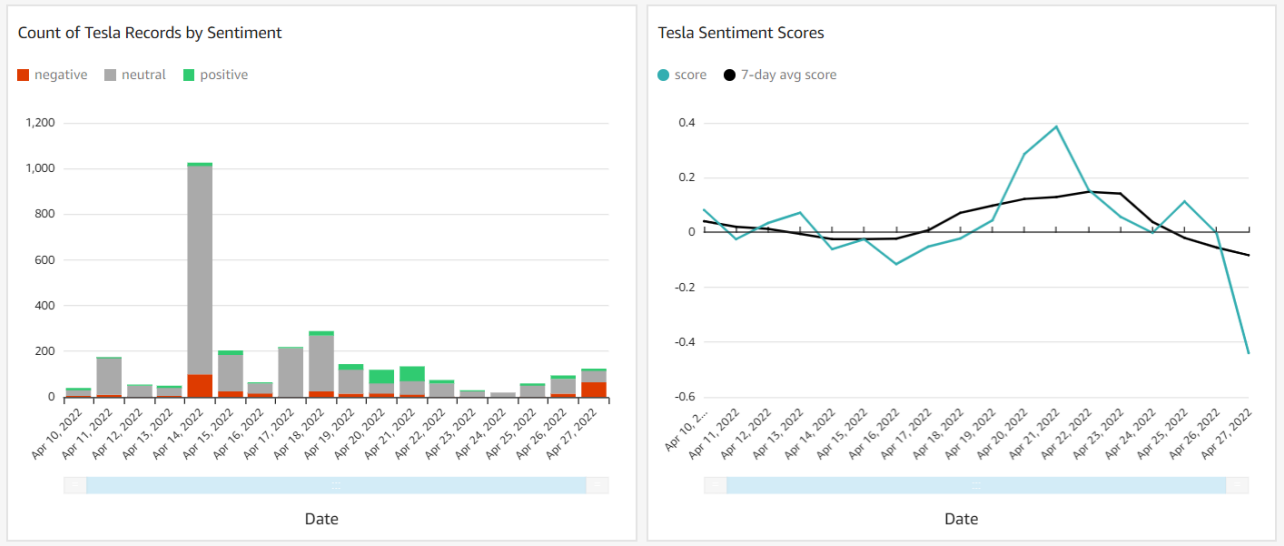

Figure 5: Screen capture of sentiment analysis graphs for Tesla over the month of April 2022 from our Quicksight dashboard. Count of records grouped by sentiment labels (left); daily and 7-day average sentiment scores (right).

prerecorded message towards the end of March. Figure 6 shows the P&O sentiment of public news in the aftermath of this event. We note two interesting features:

- While the high volume of negative articles declines after the 1st April when the scandal was still fresh, the proportion of negative articles remains high throughout the month. A 7-day average sentiment score well below zero could imply that P&O have not dealt with the situation well since the initial criticism. Reviewing some of the negative headlines over this time, we learn of numerous lawsuits, cancellations, safety failures and agency staff boycotts. It is clear that the ripples of discontent around P&O are still felt a full month after they came to prominence in the news, and that this particular scandal has made a significant dent in their corporate reputation from which they have not yet recovered.
- There appears to be an anomalous day midway through the month where P&O sentiment is mostly positive. Checking the contributing headlines for the 15th April reveals an area for improvement within our NLP modelling package. The most significant headline 'Cruise bookings boosted after P&O sackings' may be positive for the cruise industry in general, but is unclear whether this is good news for P&O explicitly. Future iterations of the product should ideally account for ambiguities like this.

Visualising our final database through the dashboard vindicated our approach to article extraction and NLP modelling. While the database is by no means perfect (see Section 3.1.3), its value is evident. The product has the potential to be shown to prospective users and improved upon within a Lean Startup-type feedback loop.

2.5 Deployment on cloud services

Deploying our software at scale on AWS proved the steepest learning curve for us. Many hours were put into experimenting with the services and debugging the errors which arose. Figure 7 depicts the infrastructure of our pipeline on the cloud with reference to the three stages discussed before.

AWS has a wide range of services capable of running the article extraction and sentiment analysis applications. For example, **Elastic MapReduce (EMR)** is specialised for deploying spark clusters, while **Lambda** allows us to rapidly deploy short functional scripts. Ultimately, **AWS Batch** was chosen as our main platform for deploying the article extractor and sentiment package as it allowed us to control the resources such as memory, vCPUs and GPUs. **AWS Batch** also has good integration

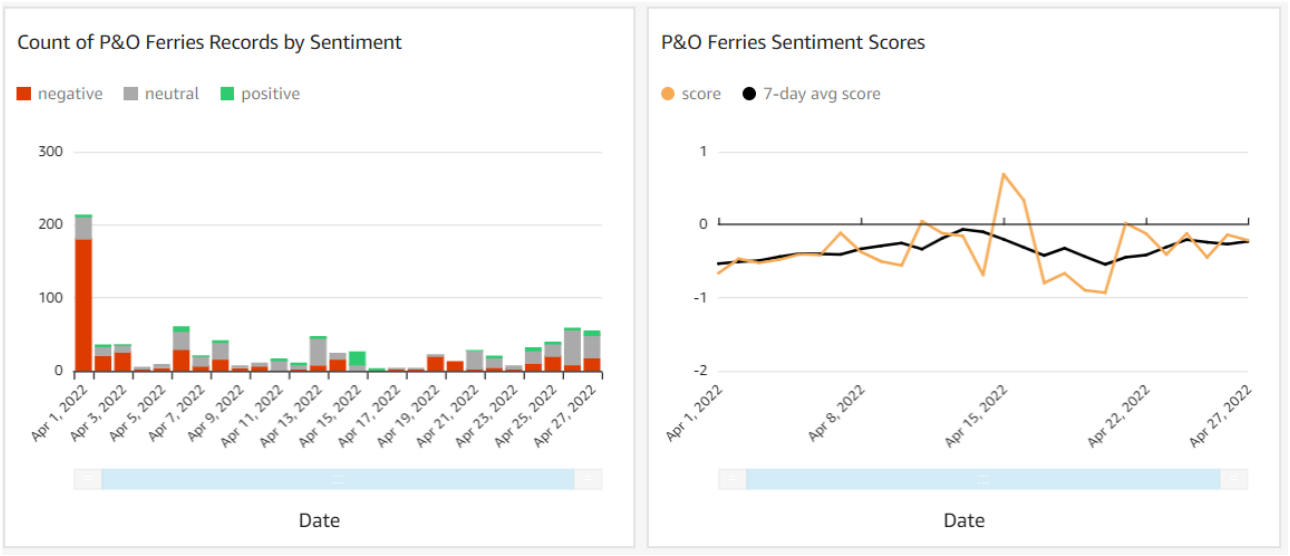

Figure 6: Screen capture of sentiment analysis graphs for P&O Ferries over the month of April 2022 from our Quicksight dashboard. Count of records grouped by sentiment labels (left); daily and 7-day average sentiment scores (right).

with **EventBridge** and **AWS Step Functions**, so deploying a pipeline that could extract and analyse the sentiment of articles daily was simple to implement. Lastly, because Batch is built on top of **Elastic Container Registry (ECR)**, all code is pushed to AWS as Docker images. Therefore, by testing images locally we guarantee that images (given they are built for AMD64) will run on AWS.

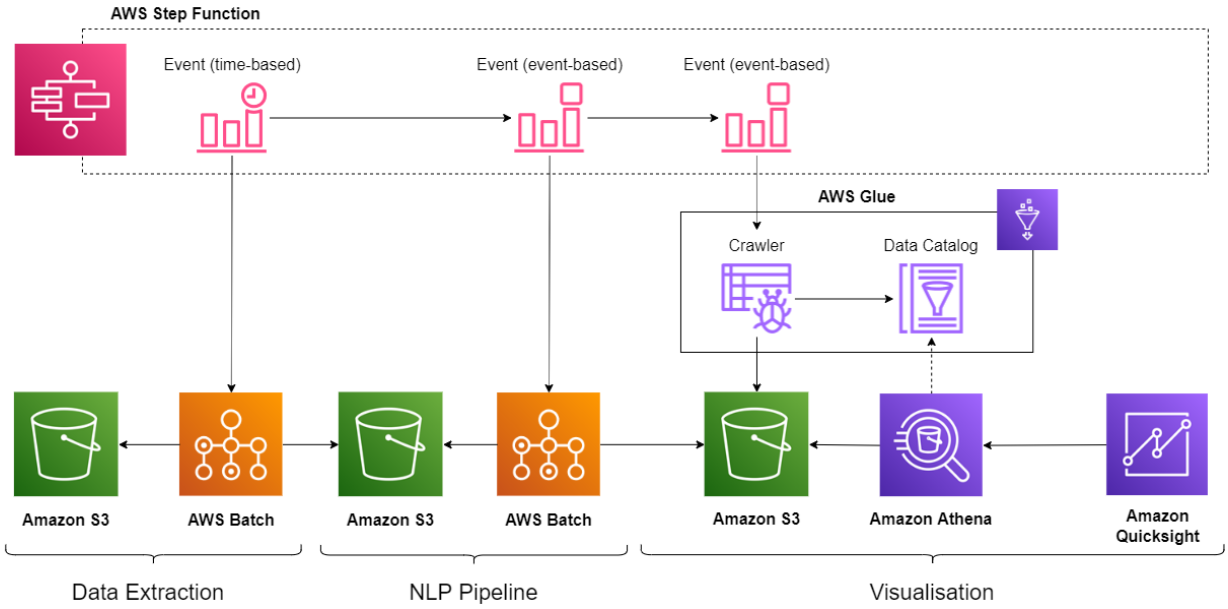

Figure 7: Schematic of our AWS infrastructure.

2.5.1 Python and Spark Base Image

`https://github.com/Brand-Sentiment-Tracking/aws-python-pyspark-base-image`

Since both the extractor and sentiment package run on Python v3.8 and Apache Spark, an base image built with both these tools was developed on top of `amazoncoretto:8`, a Linux OS with Java

8 preinstalled. This base image help to speed up deployment since updates to the applications didn't require reinstalling Spark and Python every time.

2.5.2 Extraction package

The Docker images are stored for the data extraction processes in AWS's **ECR**. These images are built for AMD64 chips and the `amazoncoretto:8` system. This registry is private for security purposes.

Typically, container images are deployed using the **Elastic Container Service (ECS)** by defining tasks which run the container. However, **AWS Batch** was used over **ECS** as it is easier to replicate tasks and run various jobs at the same time, a crucial feature for extracting historical data over a large time span. Another advantage is that with Amazon's **Auto Scaling Group**, the type of **EC2** instances required for a job to run are completely handled by AWS making it very cost-effective.

Running a Batch job can be performed through a **Fargate** or an **EC2** instance. **Fargate** is a flexible instance where the user does not need to manage the servers, however there is a limit of 4 vCPUs and 30GB of memory. **EC2** has no such limitations on resources. Both approaches have two types of provisioning models: On-Demand or Spot. With On-Demand the client is billed per second, whereas with Spot the client can save money by specifying the maximum percentage of On-Demand pricing they are willing to pay. The disadvantage of using Spot is that once the Spot price increases above the threshold the client is willing to pay, the job terminates. Hence, why **EC2** instances were used for the data extraction process due to the high resource requirements of the job.

The CC-NEWS archive is stored on **S3**: a object storage service widely regarded as cheap and secure, with unlimited digital storage capacity. As such **S3** is also appropriate for the outputs of our extraction process. We made this bucket public, following the recommendation by TDD that it would be a highly valuable resource for others wanting to perform analytical or modelling tasks on a large number of contemporary online articles.

2.5.3 Sentiment package

Similarly to data extraction, the package is deployed using a docker image and running it through **AWS Batch** using an On-Demand **EC2** instance. This is because the sentiment package is a very resource intensive so running the process on **Fargate** would take too long (17 hours compared with 3 hours on **EC2**).

2.5.4 Visualisation and event triggers

To prepare the data for analytics, an extract, transform, and load (ETL) job is required. An unstructured **S3** bucket was chosen as the final repository for time series data, preferred to a relational SQL database for consistency and storage efficiency. As compressed Parquet data cannot be loaded directly into **Quicksight**, an **AWS Glue Crawler** is required to interpret the contents of the **S3** bucket and write the metadata into a data catalog. The data catalog contains the schema and format of the tabular data. From this data catalog we use **Amazon Athena** to query the data. While there are many choices of BI platforms, Amazon's own **Quicksight** service is one of the cheaper and user friendly options, with Athena integrated into the service for efficient database querying.

With all cloud services under the AWS umbrella, we were able to automate our entire pipeline using **AWS Step Function**, a serverless service orchestrator. The extraction job on **AWS Batch** is triggered daily, with sentiment analysis, crawling and dashboard updates following on cue. The end result is a configurable pipeline feeding a dashboard that updates itself with data processed from the most recent articles posted to CC-NEWS everyday.

3 Evaluation

3.1 Product evaluation

In this section, we reflect on the merits and limitations of our final product by considering the performance of the NLP models on our dataset, the quality of our source code and the quality of our output

data. We conclude by discussing the prospects for our software in future applications.

3.1.1 Model evaluation

To evaluate our NLP models models, we manually annotated two datasets from our extracted article bucket and obtained metrics for: 1) NER; 2) sentiment extraction respectively. Our datasets are subsets of 10,000 shuffled articles from 10 different WARC files. We manually annotated 200 data points with entities and 1000 data points with our own sentiment labels. For both NLP tasks, while we used half of the labelled data points as a validation set in model selection in Sections 2.3.2 and 2.3.3, the other half of the dataset was used as a separate test set in the final model evaluation.

NER test metrics Table 6 shows the entity-level performance of our chosen model pipeline **xlnet_base**. The F1 score for ORG (51%) is slightly lower than that on the validation set (61% in Table 2) as this is highly data-dependent. The low precision on ORGs (40%) implies that more than half of ORGs found were not correctly matched, given that we do not consider any partial matches of entities, such as 'Google Art' (true entity) and 'Google' (detected entity). If we update the model metrics with a more relaxed matching strategy, accepting a correct match if the identified entity string is a sub-string of a true entity and vice versa, all the metrics improve drastically, with ORG F1 rising to 67%. We considered adopting a clustering algorithm to merge similar identified entities into one, but this may reduce the diversity of detected entities. Keeping partially matched entities can provide more information about, for example, company subsidiaries.

	Exact matching			**Partial matching**		
Entity type	Precision	Recall	F1	Precision	Recall	F1
All entities	52	69	60	67	90	77
ORG	40	70	51	53	92	67
PER	64	70	67	80	87	83
LOC	70	69	69	89	88	88

Table 6: Entity-level metrics (%) of the XLNet model on a test set of 100 manually labelled points without and with partial matching.

Sentiment identification metrics To evaluate the chosen sentiment model on our own extracted data, we used the manually-labelled test set of 500 headlines consisting of 33 positive, 79 negative and 388 neutrals. We present the performance of our chosen model 'classifierdl_bertwiki_finance_sentiment' with detailed metrics shown in Table 7, where the overall accuracy was 82%.

Class	Precision	Recall	F1
Negative	75	34	47
Neutral	84	94	89
Positive	57	52	54
Macro avg	72	60	63

Table 7: Metrics (%) of chosen classifierdl_bertwiki_finance_sentiment model on a test set of 500 manually labelled points.

As can be seen in Table 7, our model exhibits the best precision and recall on the neutral class. The recall scores imply that although we miss some non-neutral headlines, we correctly identified 34% of the actual negative headlines and 52% of the actual positive ones. The precision metrics tell us that when our model labels a headline as 'negative', it is correct 75% of the time and 57% of the time when it labels a headline as 'positive'. However, manual inspection of the data tells us that when a non-neutral headline is mislabelled, the predicted label is neutral almost every time. Although mislabelling occurs, it is extremely rare that a positive headline is labelled as negative or vice-versa.

It is worth briefly mentioning the inherent limitations of sentiment analysis. The concepts of 'positive', 'neutral' and 'negative' are vague, context dependent and subjective. Take a headline such as 'Texas killer found guilty': should this be classified as negative because it refers to a morbid event, positive because it represents justice being done, or neutral because it simply states a fact? While our final model attained an adequate sentiment test accuracy, the training, selection and test data unavoidably have a level of bias and inconsistency between annotators. Deeper understanding of foundational research around subjectivity in sentiment analysis would be useful going forward.[9]

3.1.2 Code quality

Given the quantity of information our pipeline must handle on a daily basis and the necessity to pass our product on to future development teams, our software code was required to be highly robust and readable.

To achieve this, we developed a suite of unit tests for each application deployed. There are 75 unit tests in total: 46 for the data extractor and 29 for the sentiment package. Every attribute and public method–aside from some logging methods–are tested to ensure we have captured as many edge cases as possible.

However, while testing is important for robust software, it is not impervious to subtle bugs and rare edge cases. When deploying packages on the cloud, re-running applications instantly to try catch bugs and errors by trial-and-error is not viable. To assist development, we ensured our applications have verbose logging and strict error handling. All classes in our packages have their own dedicated loggers that report everything from debug to error messages, as well as type and value checking for all configurable attributes. This is so our application will either fail fast if a crash is inevitable, or allow us to efficiently trace the logs to find unexpected bugs. Comprehensive logging worked well in practice, although a common bug we found challenging to fix using this approach was an from Java when running in AWS.

Since our packages need to be easy-to-use for others, we also put emphasis on making the code as readable as possible. We kept to the module Team Agreements by adding fully descriptive comments (according to Google's docstring guide) and linting the code (using `flake8`, which follows the PEP-8 guide). Since we are using Python v3.8, we also decided to make use of its type hinting library. Not only does it help users to understand the expected inputs, but also allows code editors to show available attributes and methods of variables.

To tie all of these practices together, we have deployed also pipelines using GitHub Actions that, on a push to the master branch, will lint the code using `flake8`, build the Docker image, run the unit tests and push to AWS ECR if they all pass. This way, we are confident that our images in ECR have all been tested, and the daily pipeline is unlikely to fail. Currently, one drawback with these pipelines is that they are dispatched onto GitHub-hosted runners. This is fine in most cases, however since we are using large models for NER and sentiment analysis, we are not able to check the main `predict_brand()` and `predict_sentiment()` methods. These tests pass when run locally, but in the future, we would look towards using self-hosted runners with a larger memory capacity to run all the unit tests at once as part of these pipelines.

3.1.3 Data quality

`https://extracted-news-articles.s3.amazonaws.com/`

The commercial viability of the product rests largely on producing useful sentiment information for third parties, which aside from effective NLP models means well-structured, well-labelled, consistent data. In an application built to handle terabytes of data, there are inevitably data losses and leakages where useful information is disregarded and unwanted information is propagated. We comment briefly on the state of our final database with regard to usefulness, completeness and reliability.

We believe that our final database fulfils the purpose of the MVP: it can be used to clearly illustrate trends in brand sentiment over time. CC-NEWS process around 0.5M to 1.5M articles every day. After some data analysis, we concluded that approximately 40% of the CC-NEWS articles are English meaning that the product goes through 200,000 to 550,000 articles every day. However due to

the nature of CC-NEWS, around 9% of these headlines prove to be of low quality. Some appear not to be real news headlines, often being very short (e.g. 'Toronto 103, Philadelphia 88', 'Subscribe here!). Others aren't in English. We use `langdetect` to recover the language of the main text, but sometimes although the main text contains a substantial amount of English, the headline is not actually English. However, any brands detected in these articles are unanimously assigned neutral, hence don't affect the sentiment score meaningfully.

Our NER model does not manipulate the entities after detecting them, hence possessive nouns, hyphenated phrases, and different cases are treated as separate entities. This is by design, as being able to differentiate between these may well be of interest; however, clustering or filtering some of the entities would improve the reliability of the data for visualisation. Searching the database for 'russia' yields 408 entity results, including 'Russia's', 'Russia-NATO', '9-Russia and 'RUSSIA'. 74% of these data points correspond to 'Russia' or 'Russian', suggesting that filtering out less common entities would be an easy way to clean up the database.

3.2 Future potential

We believe that the insight our software can provide has great commercial potential. We conclude by offering further enhancements for future iterations of the MVP.

Given that the current software only handles articles written in English, a natural extension would be to develop a product that could perform brand sentiment analysis in multiple languages and utilise more extracted data. Large interlingual or multiple language-specific NLP models would be required for this task. In addition to extending language capabilities, the country—or perhaps even city—from which each article originated could be incorporated into the database. Segmenting brand sentiment geographically could allow multinational companies to distinguish where they have favourable and unfavourable coverage.

Refinements could be made to the ML pipeline to generate more credible assessments. The most obvious would be to model over the body text of articles in addition to the headlines. One essential adjustment would be to distinguish between sentiments expressed towards different entities in the same text, possibly by combining the NER and sentiment models. Another would be to ensure the scoring system reflected the potential improvement or damage to brand perception an article might have based on the context in which a brand is mentioned.

Incorporating weighting factors to variegate the relevance of each article by source would also be a valuable addition. For example, a Washington Post article read by 500,000 people has a larger impact to corporate reputation than a similarly themed article from a regional journal read by 5,000 people. Factoring these disparities into the scoring system by organising sources into tiers of significance or even coupling with real-time website engagement metrics would improve insight.

Looking further ahead, linking the entities in our database to express media sentiment towards the network around a brand as well as the brand itself would be an exciting prospect. Aligning entities to a knowledge base, mapping relations between organisations, products, people and places, and developing further algorithms to characterise sentiment across these networks would increase the sophistication of our product. Such a tool could be highly valuable not only for TTD, but any institute conducting large-scale market analysis or trend forecasting.

References

[1] Common crawl news dataset, Oct 2016. Accessed at https://commoncrawl.org/2016/10/news-dataset-available/.

[2] Warren Buffett, 2014. As quoted in Corporate Survival: The Critical Importance of Sustainability Risk Management (2005) by Dan Robert Anderson, p. 138. Accessed at https://en.wikiquote.org/wiki/Warren_Buffett.

[3] Deloitte. Global survey on reputation risk, 2014. Accessed at https://www2.deloitte.com/global/en/pages/governance-risk-and-compliance/articles/reputation-at-risk.html.

[4] Felix Hamborg, Norman Meuschke, Corinna Breitinger, and Bela Gipp. news-please: A generic news crawler and extractor. In *Proceedings of the 15th International Symposium of Information Science*, pages 218–223, March 2017.

[5] Suleiman Khan. Bert, roberta, distilbert, xlnet — which one to use?, 2019. Accessed at https://towardsdatascience.com/bert-roberta-distilbert-xlnet-which-one-to-use-3d5ab82ba5f8.

[6] Jiefei Ma, April 2022. From personal communication.

[7] P. Malo, A. Sinha, P. Korhonen, J. Wallenius, and P. Takala. Good debt or bad debt: Detecting semantic orientations in economic texts. *Journal of the Association for Information Science and Technology*, 65, 2014.

[8] Zenith Media, Decemeber 2021. Accessed at https://www.zenithmedia.com/digital-advertising-to-exceed-60-of-global-adspend-in-2022/.

[9] Andrés Montoyo, Patricio Martínez-Barco, and Alexandra Balahur. Subjectivity and sentiment analysis: An overview of the current state of the area and envisaged developments. *Decision Support Systems*, 53(4):675–679, 2012.

[10] Shuyo Nakatani. Language detection library for java, 2010. Accessed at https://github.com/shuyo/language-detection.

[11] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *CoRR*, cs.CL/0306050, 2003.

[12] Weber Shandwick. The state of corporate reputation in 2020: Everything matters now, January 2020. Accessed at https://www.webershandwick.com/news/corporate-reputation-2020-everything-matters-now/.

[13] Ankur Sinha. Sentiment analysis for financial news, May 2020. Accessed at https://www.kaggle.com/datasets/ankurzing/sentiment-analysis-for-financial-news/.

[14] Jun Wang, Weinan Zhang, and Shuai Yuan. Display advertising with real-time bidding (rtb) and behavioural targeting. *Foundations and Trends® in Information Retrieval*, 2017.