The Alan Turing Institute

Data Study Group Final Report: WWF

9–13 December 2019

Smart monitoring for conservation areas

https://doi.org/10.5281/zenodo.3878457

This work was supported by The Alan Turing Institute under the EPSRC grant EP/N510129/1

Smart Monitoring for Conservation Areas

WWF Data Study Group team
(December 2019)

1 Executive Summary

1.1 Challenge Overview

WWF (World Wide Fund for Nature) monitors over 250,000 protected areas (e.g. national parks and nature reserves) and thousands of other sites and critical habitats. These sites are the foundation of global natural assets and are central to the preservation of biodiversity and human well-being. Unfortunately, they face increasing pressures from human development.

In this challenge, we explore various data science techniques to automatically detect news articles that report emerging threats to key protected areas. We describe a system that identifies such news stories near real-time. This is vital to enable the wider machinery of WWF and the conservation community to engage with governments, companies, shareholders, insurers, and others to help halt the degradation or destruction of key habitats.

1.2 Data Overview

A dataset containing approximately 45,000 JSON files describing news stories relevant to UNESCO World Heritage Sites was provided. The news articles were scraped using Google News API over two years (from January 2018 to October 2019). The news articles were in English.

In addition, a training dataset of 135 news articles annotated by the WWF experts was supplied to facilitate the news' relevance assessment in 3 classes: level 0 (no threat), level 1 (threat detected but no actors), and level 2 (threat and actors detected). However, as it will be discussed later, this dataset was expanded by the data providers during the DSG week to 1,000 labelled data.

Several other datasets describing the details of assets within World Heritages Sites were also provided, including the coordinates (latitude and longitude) of the centroids of the World Heritage Sites and the coverage area in square kilometers. Several additional datasets describing the details of assets (e.g. lists of powerplants, oil and gas assets and concessions, global ports and airports, hydroelectric dams, etc.) within World Heritage Sites were also supplied. Due to time constraints, we could not incorporate them into the analysis, but there are plans to use them as part of future work.

1.3 Main Objectives

The main aim of the data study week was to understand how data science can help to answer the following question: can we detect and geolocate public

news stories describing emerging threats to key protected areas using a small number of labelled data? In order to achieve this goal, we compared various classification techniques for detecting threats reported in news articles and tried different feature extraction strategies using natural language processing (NLP) and other text mining techniques.

1.4 Approach

Our labelled dataset consists of 1,000 news articles with three labels: level 0 (no threat), level 1 (threat detected but no actors), and level 2 (threat and actors detected). See Section 3 for a detailed description of the dataset. In our workflow, we combine levels 1 and 2 into one group and classify news articles into one of the following two classes: relevant or not-relevant. Here, relevant means that a news article reports on a potential threat (such as human activities) around a World Heritage Site regardless of the mentions of actors.

We formulated the threat detection task as a (semi-)supervised learning problem. Two classes of supervised classification techniques were studied during the DSG: (i) text feature extraction combined with off-the-shelf classifiers/regressors including logistic regression, Naive Bayes, K-Nearest Neighbours (KNN), Random Forest, Ridge Regression and Support Vector classifiers, and (ii) deep neural networks including recurrent neural networks (RNN) and transformer architectures.

As depicted in Figure 1, our proposed approach can be divided into two main steps: training and prediction. In the first step, a new binary classifier is trained, and its performance (i.e. its accuracy, precision, recall, and F1-score) is measured on the validation set. In Section 4, we will discuss the preprocessing and feature extraction steps in details. This includes language detection, linguistic processing, sentiment analysis, topic modeling, embeddings and splitting the labelled data into two balanced datasets (i.e. training and validation sets). The outputs of this step are articles in vectorised representations of the news articles. These, together with the labelled data, are then used to train the classifier and evaluate its performance. We will present our best performing classifier based on BERT in Section 5. We experimented with a range of classifiers, such as, off-the-shelf non-deep-learning classifiers, Recurrent Neural Networks as implemented in TensorFlow [3] and transfer learning with the *fast.ai* library [8]. The architecture of these models and their performance are discussed in Section 8.6.

In the second step, the selected trained classifier is used for prediction. Similar to the previous step, first a set of features are extracted from a batch of inputs (i.e. unlabelled news articles). Next, based on the assigned labels, a set of inputs are labelled as "relevant" and passed to the postprocessing step. In the postprocessing, mentions of places, organization, facilities, and datetimes in each news article are detected and resolved (refer to Section 6 for more discussion). According to the probabilities assigned to each article, a set of "uncertain" articles is created. This is a list of articles that the classifier was most uncertain about their labels (i.e. the predicted probabilities or scores of the two classes is close to 50%). This set is passed to our "Smart Annotation" algorithm described in Section 7, which outputs the data that has been newly labelled by the user. We add these labelled articles to our train and validation sets in the "training step". Finally, the results of the trained classifier are reported.

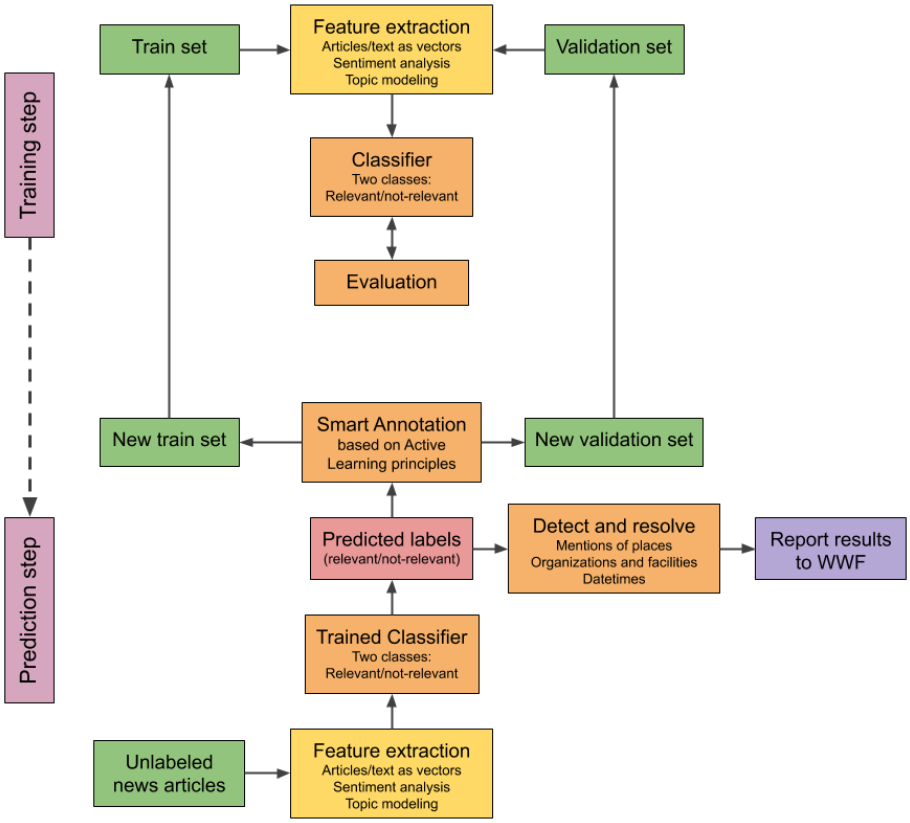

Figure 1: Our proposed approach for smart monitoring for conservation areas. This approach consists of two main steps: training and prediction. See text for discussion. Each aspect of our algorithm will be discussed in detail in the subsequent sections.

1.5 Main Conclusions

A deep learning method based on the BERT architecture [6] showed the best performance in the binary classification task. The performance metrics measured on the validation dataset were 96.35% for recall, 81.48% for precision, 88.29% for F1-score and 86.38% for accuracy. This model significantly outperformed all the other classifiers that we tried, including various RNN (Recurrent Neural Network) models and off-the-shelf non-deep-learning classifiers. As features, the BERT model uses embedded content, sentiment score and topics extracted from each news article. The performance of BERT was affected significantly by these features. For each article labelled as relevant, we detect and resolve mentions of places, organizations, facilities and dates in the postprocessing step.

Since we had a limited number of labelled data, we developed a "Smart Annotation" technique to find the most informative unlabelled news articles. This way, we could considerably reduce the cost of labelling new training data by focusing only on those news articles that our model is most uncertain about.

This forms a workflow in which the performance of a trained classifier can be improved iteratively.

2 Introduction

WWF is one of the world's largest independent conservation organisations, active in nearly 100 countries. WWF monitors over 250,000 protected areas (e.g. national parks and nature reserves) and thousands of other sites and critical habitats (e.g. coral reefs and mangroves). These sites are the foundation of our global natural assets and are central to the preservation of biodiversity and human well-being. Unfortunately, they face increasing pressures from human development, with mines, oil and gas operations destroying and degrading habitats; dams altering river flows; agriculture causing habitat loss; road and rail expansion fragmenting and opening areas to further degradation.

With a vast array of threats emerging on a daily basis across 250,000+ sites, a growing challenge for WWF and the wider conservation movement has been to consistently and timely identify emerging or proposed human developments within key sites and the stakeholders involved. The timely provision of this actionable information is vital to enable the wider machinery of WWF and the conservation community to engage with governments, companies, shareholders, insurers, and others to help halt the degradation or destruction of key habitats. Earlier engagement is often critical in gaining a positive outcome for the environment before projects are well established and significant investment has already occurred.

To help achieve this, the Conservation Intelligence (CI) team at WWF pools vast volumes of spatial data, running GIS assessments and applying remote sensing to define the extent of threats and degradation to scale the issues, inform the public and guide WWF's interventions. Wherever possible, the CI team attempts to identify the actors involved, that is, governments, companies, shareholders, insurers, and others, as this provides an actionable means to effectively engage to help limit negative impacts.

As part of their work, the CI team developed WWF-SIGHT[1]. SIGHT is a global mapping platform based on ESRI software. Uniquely, it integrates both commercial and non-commercial spatial data, satellite imagery and various functionalities. As a result, this data portfolio provides WWF with unparalleled insights into who is operating where, and with basic remote sensing functionality to check the sites in near real-time.

This study is a first step towards building a monitoring system based on web scraping news stories to identify current and potential human pressures across protected areas. As a case study, we focus on the flagship protected areas, natural World Heritage Sites (244 sites, such as Serengeti or the Great Barrier Reef). The monitoring system developed here is able to classify news stories retrieved using Google News API into two groups of relevant and not-relevant. Moreover and in order to reduce the human-time, we developed a "smart annotation" system that helps to identify the most relevant news articles that need to be annotated manually. For each article flagged as relevant, we extract key terms including involved stakeholders, dates and locations.

[1] https://www.wwf.org.uk/updates/wwf-sight-conservation-intelligence

3 Data Overview

3.1 News Articles

Our dataset consists of 44,746 news articles in English mentioning World Heritage Sites. These articles were scraped from the web using Google News API[2] over the period between January 2018 and October 2019. The selection of articles was based on keyword matching of the English names of the World Heritage Sites.[3] For example, the query used to extract articles mentioning the 'Waterton Glacier International Peace Park' is `("Waterton" AND Glacier International Peace Park) OR ("Waterton" AND Lakes National Park)`. This resulted in 44,746 news articles from 2,974 different sources with mentions of at least one of 244 World Heritage Sites.

The news articles were provided in a JSON format with the following data fields:

- *Source:* Id and name of the source where the article has been published.
- *Title:* The headline or title of the article.
- *Description:* A description or snippet from the article.
- *URL:* The direct URL to the article.
- *Published at:* The date and time when the article was published.
- *Content:* The full text of the article.

Each article was also accompanied with the unique identifier of the World Heritage Site mentioned in it, which can be used to link it back to external WWF databases (see Section 3.3 for a description of the supplementary data on World Heritage Sites provided by WWF).

3.2 Labelled Data

A subset of 135 news articles annotated by WWF experts was provided to facilitate the relevance assessment of the news articles. Each article was labelled according to a threat level, as follows:

- *Threat level 0:* No threat to the World Heritage Sites is discussed in the article.
- *Threat level 1:* A threat to the World Heritage Sites has been identified in the article, but the actors associated to the threat (e.g. gas companies, airports, or new cruise lines) are not mentioned in the article.
- *Threat level 2:* A threat to the World Heritage Sites has been identified in the article, and the actors causing the threat are also mentioned in the article. In such cases, an additional annotation layer was provided which carries the keywords for threats and actors mentioned in each article.

[2] `https://newsapi.org/s/google-news-api`.

[3] The UNESCO list of World Heritage Sites can be found here: `https://whc.unesco.org/en/syndication`.

During the Data Study Group, we collated threat levels 1 and 2 and focused on a binary classification task: *relevant* (i.e. threat level 0) versus *not-relevant* (i.e. threat levels 1 and 2). Threat-level ternary classification can be approached as a refinement to this and will be considered as part of future work.

One of the concerns posed at the beginning of the Data Study Group was that the size of our labelled dataset was possibly too small for a classifier to extract meaningful patterns. This limits the ability of the classifier to generalize and to correctly predict whether a news article is discussing a threat to a World Heritage Site. To have sizable training and validation sets, the WWF experts expanded the labelled dataset to 999 articles during the Data Study Group week.

To generate the labelled dataset, ≈1000 articles were selected randomly from the full sample while enforcing a level of representativeness by ensuring the presence of different World Heritage Sites in the articles. The final labelled dataset contains mentions to 224 different World Heritage Sites. Table 1 summarizes the number of articles in different classes and threat levels.

Table 1: Number of articles in each class. Note that the **not relevant** class matches exactly the **threat level 0** class; whereas the **relevant** class corresponds to **threat level 1** and **threat level 2** together. We use relevant/not-relevant labels for the binary classification tasks in Section 5.

Class	Number of articles
Not relevant	782
Relevant	217
Threat level: 0	782
Threat level: 1	149
Threat level: 2	68

3.3 World Heritage Sites Supplementary Data

Supplementary data on each World Heritage Site was provided by WWF. This includes the coordinates (latitude and longitude) of the centroids of the World Heritage Sites and the coverage area in square kilometers. This dataset was used in the post-processing of the data, and in particular in geocoding.

Additionally, several datasets describing the details of assets (e.g. lists of power plants, oil and gas assets, and concessions, global ports and airports, hydroelectric dams, etc.) within World Heritage Sites were also supplied. However, due to time constraints, we could not incorporate these into the analysis, but there are plans to use them as part of future work.

4 Preprocessing

In this step, the raw full text of each article is preprocessed and transformed to a vectorized representation which can then be consumed by the classifier. The vector representation may contain information about the content, sentiment or

topics. We performed the following preprocessing steps: language detection, linguistic processing, sentiment analysis, topic modelling, embedding, and splitting the data into balanced datasets (i.e. training and validation sets). Each step is discussed in detail in this section.

4.1 Language Detection

Even though most articles in our dataset were in English (as it was specified in the web scraping criteria), we found out that there were several articles in languages other than English. We used the `langdetect`[4] [14] library to automatically detect the language of each news article by applying it to each article title and description combined. 6,256 articles were identified as non-English. Table 2 shows the frequency of articles in the top five languages in our data. All non-English news articles were filtered out and were not used in the subsequent steps.

Table 2: Top five detected languages in the news articles dataset with their frequencies.

Language	**Frequency**
English	38,490
Portuguese	1,666
Italian	1,460
German	651
Spanish	613

4.2 Linguistic Processing

Linguistic preprocessing is an important step for many natural language processing tasks. For each article in the dataset, we removed all non-alphabetical characters and lower-cased the remaining text. We then used the `SpaCy`[5] [7] library for tokenisation (i.e. splitting the text into tokens, which are minimal units broadly corresponding to words) and for lemmatisation (i.e. transforming the tokens into their lemma or dictionary form). Finally, we filtered out all stop words (i.e. very common, function words, such as *the*, *a*, *but*, or *in*) from the lemmatised text.

Text expressions can often be multi-token, such as 'New York', where the tokens 'New' and 'York' separately have different meanings than the combined multi-token 'New York'. Sequences of two adjacent tokens that have a strong meaning together (such as 'New York') are called bigrams, and sequences of three adjacent tokens that have a strong meaning together (such as 'World Heritage Site') are called trigrams. As a last linguistic preprocessing step, we identify bigrams and trigrams and group them together. We used the `phrases`

[4] `https://pypi.org/project/langdetect/`

[5] SpaCy is a widely used Python library for natural language processing (`https://spacy.io/`). Throughout this report, we use the spaCy model for English `en_core_web_sm`, see: `https://spacy.io/models`.

module of the `Gensim` library[6] [18] with its default parameters, which bases its bigram scoring function on Mikolov et al. 2013 [13].

Table 3 shows an example text undergoing the different aforementioned preprocessing steps. The preprocessed texts are then passed as inputs to sentiment analysis (Section 4.3) and topic modelling (Section 4.4).

Table 3: Step-by-step linguistic preprocessing of the raw text.

Preprocessing	**Output**
raw text	The Mount Etna volcano was among 19 sites that have been added to UNESCO's World Heritage list of places of "outstanding universal value."
Non-alphabetical characters removal	The Mount Etna volcano was among sites that have been added to UNESCO s World Heritage list of places of outstanding universal value
Lower-casing	the mount etna volcano was among sites that have been added to unesco s world heritage list of places of outstanding universal value
Tokenization	[the, mount, etna, volcano, was, among, sites, that, have, been, added, to, unesco, s, world, heritage, list, of, places, of, outstanding, universal, value]
Lemmatization	[the, mount, etna, volcano, be, among, site, that, have, be, add, to, unesco, s, world, heritage, list, of, place, of, outstanding, universal, value]
Stop-words removal	[mount, etna, volcano, site, add, unesco, world, heritage, list, place, outstanding, universal, value]
Bigram and trigram processing	[mount_etna, volcano, site, add, unesco_world_heritage, list, place, outstanding, universal, value]

4.3 Sentiment Analysis

In natural language processing, sentiment analysis is the process of classifying texts as positive, negative, or neutral. Following the intuition that sentiment polarity may be an indicator of the threat that is present in the article's content, we have used sentiment analysis as a preprocessing step to the classification task. We performed sentiment analysis on headline, description, and overall content of each article using the `TextBlob` package[7] [2]. `TextBlob` uses a predefined dictionary for each word in a text, and the overall sentiment of the text is determined by averaging the polarity of all its words.

Sentiment is often calculated from two variables, namely polarity and subjectivity. In `TextBlob`, both are floating points. Polarity is in the interval [-1, 1] where -1 refers to a 'negative' sentiment and 1 refers to a 'positive' sentiment. Values close to 0 are normally referred to as 'neutral'. Subjectivity lives in the interval [0, 1] where 0.0 refers to an objective input (closer to a fact) and 1.0 refers to a subjective input (closer to an opinion). For the sentiment analysis, we

[6] `https://pypi.org/project/gensim/`.
[7] https://textblob.readthedocs.io/en/dev/.

considered polarity > 0.1 as positive and polarity < -0.1 as negative. Polarity in the interval $[-0.1, 0.1]$ is considered neutral.

Figure 2 shows the measured polarity of article's title, description and content in three panels. The majority of articles were neutral, that is, their sentiment score falls within ± 0.1 interval. The frequency of news articles grouped by their sentiment level (i.e., positive, neutral, negative) is shown in Figure 3. Here we find that most news articles are grouped as neutral, regardless of the text (title, description or content) used in the analysis. It also seems that there are more positive news articles in our dataset than negative ones. However, upon inspection it was found that looking at title, description and content sentiment individually could lead to misleading results due to contradicting sentiments between the headline and the text. Therefore, we assigned an average score to the combined headline and content.

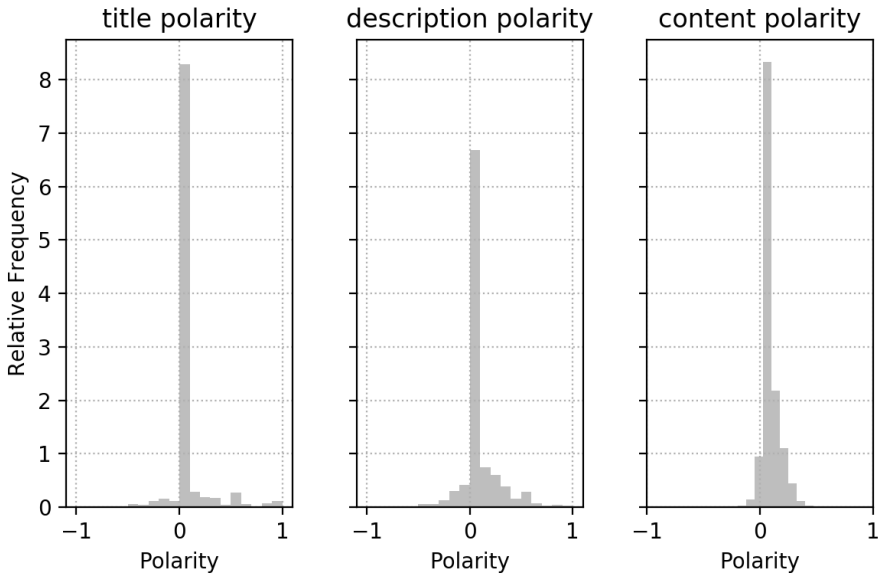

Figure 2: The distribution of phrase polarity for article title (left), description (middle) and content (right). The majority of articles are neutral (between -0.1 and 0.1) for all three categories.

`TextBlob` was found to successfully identify negative articles; however, it was less successful in identifying positive articles. A reason for that could be its relatively simplistic approach of averaging the sentiment for each word individually. In Section 8, we discuss potential next steps towards improving the sentiment analysis application to this task.

4.4 Topic Modelling

We implemented the Latent Dirichlet Allocation (LDA) [5] technique to topic modelling as a way of estimating the topic mixture of the news articles in an unsupervised way. LDA posits that each text is a distribution over `N` topics,

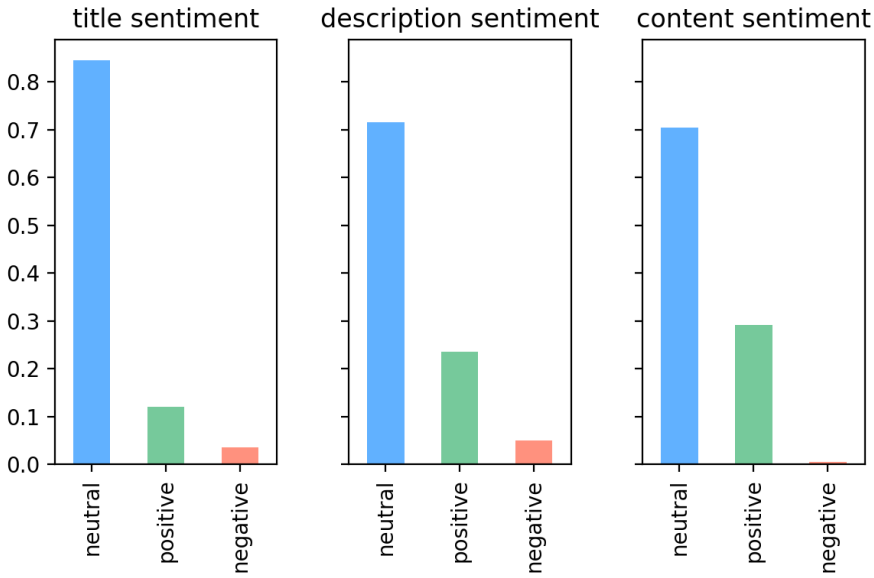

Figure 3: The relative frequency of all sentiments for title (left), description (middle) and content (right). In all cases, the majority of articles are neutral and with more positive instances than negative.

while each topic is in turn a distribution over `M` words. Topics are defined by their top words which are the most probable ones to occur given the topic. Each topic can then be used as a feature in the classifiers.

4.4.1 Implementation

We use the preprocessed texts as inputs (see Section 4 for details) and restrict the analysis to texts in English. If there was no overlap between the vocabularies of English and other languages, non-English words would simply form clusters which would not overlap with the English-language ones. However, this is obviously not the case since modern languages include many borrowings. To avoid spurious effects arising from this, we remove entries in other languages.

We perform an additional cleaning step in which most frequent tokens (presumably least informative ones [10]) are excluded from the analysis. The words are then converted into vectors of the size of the entire vocabulary with frequency entries (as implemented in `scikitlearn`[8] [17]). These vectors are then used to train an LDA model on the full set of articles. We compare two different implementations, via the `sckikit-learn` and `gensim`[9] [18] libraries, finding comparable results. We opted for `sckikit-learn` as it offers a slight advantage in terms of computational speed. Finally, the performance of the model is evaluated using the topic coherence score [15, 19] as implemented in the `gensim` library.

[8] `https://scikit-learn.org/stable/`
[9] `https://radimrehurek.com/gensim/`

4.4.2 Hyperparameter Optimisation

The number of topics is determined based on the coherence score. Several models with different number of topics (ranging from 5 to 90 in steps of 5) are fit to the data using the `gensim` package. We observe a distinct maximum around 50 topics although the values are all in a fairly narrow range between 0.4-0.5 in coherence score, indicating that they may not be very semantically cohesive. We note that this comparison is carried out with only a single pass through the corpus due to the computational cost. The choice of 50 topics is vindicated by the observation that the families of words arising from such a run are more naturally clustered than for smaller or larger numbers of topics. We choose to fit the model in 10 epochs. This is a compromise between an improved disambiguation of the topics and computational cost.

4.4.3 Visualisation

A simple approach to visualize the distribution of terms in topics, as well as their weights, is by means of word clouds. We use the Python library `wordcloud`[10] for this purpose and are able to identify topics relating to 'Russia/USSR', 'film', 'climate change', 'cruise holidays', 'conservation in Africa', 'representative democracy', 'evolution and the Galapagos', 'paleonthology', 'Harry and Meghan', 'Climate change/policy', 'weather forecasting', 'volcanoes', 'electrical power', 'conservation in south-east Asia', and 'marine life', among many others. One example is shown in Figure 4 related to conservation in Africa.

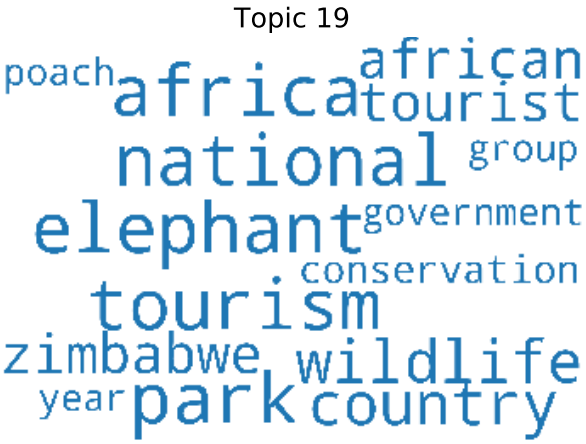

Figure 4: Visualisation of the "conservation in Africa" topic extracted from news articles using the Latent Dirichlet Allocation (LDA) technique. Words related to the topic are clustered around each other.

[10] `https://pypi.org/project/wordcloud/`

4.4.4 Results

Our trained LDA model is applied to the news articles yielding the topic content of each article normalised such that the sum over the mixture of all 50 topics for a given article is unity. Consequently, each topic can be interpreted as a feature to be fed to the classifier with the intuition that there is a correlation between the labels of the training data and (at least some of) the topics.

4.5 Embeddings

In Sections 4.3 and 4.4, we discussed sentiment analysis and topic modelling as two ways to extract features from news articles. Here, we use (word) embeddings to represent the news article texts (title, description or content) in a machine-readable format. Pretrained neural network embeddings can map the words (i.e., discrete/categorical variables) in a corpus to dense vectors of continuous values. These low-dimensional representations, combined with other extracted features, can then be used as inputs to classifiers.

Static, non-contextualized language models (e.g. Word2Vec [13]) generate unique embeddings for each word. One obvious downside is that these approaches assign, for example, the same representation for 'bank' in both 'bank deposit' and 'river bank'. Contextualized language models such as BERT, on the other hand, generate different representations for the same word depending on the other words in the sentence (i.e., context).

In this study, we experiment with both types of embeddings. In Section 5, we use BERT contextual embeddings [6], which are dynamically informed by the words around them. In Section 8.6, we experiment with three context-free methods to encode text: a pretrained Word2Vec model, token-counts and TFIDF (term frequency–inverse document frequency).

4.6 Splitting the Data

We split the dataset into two subsets, train and validation sets. It is important to have similar label distributions in these sets as the provided labelled dataset in this study is imbalanced; the relevant articles (22% of the rows) are greatly outnumbered by the not-relevant articles (78% of the rows. See Section 3.2 for the absolute numbers and the definition of threat levels). The classifiers in Sections 5 and 8.6 are trained on the training set and their performance (e.g., accuracy, precision, recall and F1-score) is measured on the validation set. This enables us to check the generalizability of the model and prevent overfitting.

5 Classification Using BERT

This section explores the application of BERT (Bidirectional Encoder Representations from Transformers) [6] in classifying news articles into relevant/not-relevant classes. BERT is a popular approach that makes use of the Transformer architecture to learn contextual relations between words (or sub-words) in a text. BERT is designed to pretrain deep bidirectional representations from unlabeled text, and it makes extensive use of recent architectural developments in NLP, such as multi-headed self-attention. Compared to rivaling approaches, BERT is trained bidirectionally, from preceding and subsequent context (in practice,

Table 4: Performance of six classifiers in `scikit-learn` library on the validation set and measured by the average precision, recall and F1-scores. In all classifiers, titles were embedded using BERT and used as features. MLP: Multilayer perceptron; K-NN: k-nearest neighbors.

	Naive Bayes	Logistic Regression	K-NN	MLP	Random Forest	Ridge Regression
Precision	0.3867	0.6539	0.5207	0.6208	0.5868	0.4889
Recall	0.6212	0.5307	0.4549	0.4908	0.2537	0.5615
F1	0.4738	0.5841	0.4841	0.5380	0.3538	0.5202

this is done by masking out a subset of words in each sentence, and predicting these missing words from the remaining ones). We use the HuggingFace implementation of BERT in this work [23].

We will discuss (1) feature extraction using BERT (with fixed parameters) and using those features in a range of classifiers (K-Nearest Neighbour, naive Bayes); (2) end-to-end fine-tuning of BERT. Algorithms 1 and 2 list the main steps in these tasks.

Algorithm 1 Feature extraction and Scikit learn classifiers

1: Fix parameters in BERT, use forward pass to generate sentence level embeddings:
2: *embedding = bert(input_sentence)*
3: Prediction using a range of classifiers (e.g., K-NN and naive Bayes):
4: *prediction = classifier(embedding)*

Algorithm 2 End-to-end fine-tuning of BERT and classification

1: Fine-tune BERT for text classification.
2: Prediction using BERT:
3: *prediction = classifier(embedding)*

5.1 Feature Extraction and `Scikit Learn` Classifiers

Table 4 outlines the performance of Algorithm 1 in which first embeddings using BERT are generated, and then these features are used in various models in `scikit-learn` library to classify articles into relevant/not-relevant categories. The dataset was split into train and validation sets with 80% and 20% of all labelled news articles, respectively. In these classifications, only the titles were used, and not the news content. To measure the statistical performance of the models, a Monte Carlo cross validation approach was used and the averaged values of precision, recall and F1 are reported in Table 4.

Table 5: Performance of BERT trained on titles only. The results are significantly better compared to Algorithm 1, see Table 4.

	BERT + MLP
Precision	0.5054
Recall	0.8704
F1	0.6395

5.2 End-to-End Training of BERT

5.2.1 Using Titles

In Algorithm 2, BERT is used to transform titles into embeddings, and after that, a linear layer maps the features to the target space (two classes, relevant and not-relevant). To offset the class imbalance, class-weights of (approximately) inverse frequency are included in the cross-entropy loss. Table 5 lists the results which are significantly better compared to Algorithm 1 in Section 5.1.

5.2.2 Using Content

While fine-tuning BERT in Section 5.2.1 lead to promising results from news titles alone, significant performance gains are possible when embedding the entire content of articles instead. To take this into account, we first concatenate the title and the content, and then we use the first 300 words to generate an embedding (this is the maximum sentence length for deploying HuggingFace's BERT implementation on a TITAN X GPU at a reasonable batch size). Dropout with probability of 0.2 is employed for regularization, and the training is done with batch size of 8. This classifier resulted in a recall of 0.9444 with a precision of 0.5795. The confusion matrix shown in Table 6.

Table 6: Confusion matrix for the BERT model trained on titles and part of the content. This model resulted in a recall of 0.9444 and a precision of 0.5795.

	Predicted No	Predicted Yes
Labelled No	159	37
Labelled Yes	3	51

5.2.3 Leveraging Sentiment and Topics

The performance of the model in Section 5.2.2 can still be improved by adding more features. Here, we added three sets of new features: (1) sentiment scores (Section 4.3) assigned to title; (2) and content; (3) topics extracted using unsupervised LDA described in Section 4.4.

In this experiment, the MLP (multilayer perceptron) in the end-to-end BERT architecture was expanded to account for additional dimensions. In total, the feature vector has 768 dimensions for the BERT embeddings, 50 dimensions for the topics and 2 dimensions for the sentiment analysis. Best results were achieved when applying dropout only to the BERT embeddings while leaving the other dimensions unperturbed.

Table 7: Comparison between the performance of four BERT models trained using different feature vectors. The best model includes embeddings of the title and part of the content, sentiment scores (Sent.) and topics extracted from each article.

	BERT Title	BERT Content	BERT Content + Sent.	BERT Content + Sent. + Topics
Precision	0.5054	0.5795	0.7602	0.8148
Recall	0.8704	0.9444	0.9489	0.9635
F1	0.6395	0.7183	0.8441	0.8829

Table 7 compares the performance of four BERT models trained on different feature vectors: only title; title and content; title, content and sentiment score; title, content, sentiment score and topics (from our LDA topic modelling). Including only sentiment features to the content embeddings improves the recall only marginally; however, the precision score improves significantly and reaches to 0.7602. This is $\approx 18\%$ increase in the precision compared to only using content embeddings (Section 5.2.2). This improvement is unexpected and needs further investigation.

Our best performing model uses content embeddings along with features from sentiment analysis and topic modeling as shown in Table 7. The model was able to detect 132 out of 137 threats and returning only 30 erroneous predictions in not-relevant class. Table 8 shows the confusion matrix of this model.

Table 8: Confusion matrix for our best performing BERT model.

	Predicted No	Predicted Yes
Labelled No	90	30
Labelled Yes	5	132

6 Postprocessing

In this step, we describe a series of tasks that have been applied to the output of the classifier. In particular, in Section 6.1 we describe detection of facilities and organizations on articles identified as describing threats, in Section 6.2 we describe dates detection and parsing, and in Section 6.3 we describe the process of finding mentions of place names and resolving them to their geographic coordinates.

6.1 Facility and Organization Detection

As a first step towards identifying the actors in articles classified as relevant, we use `spaCy` to detect named entities that correspond to the `FAC` class (i.e. buildings, airports, highways, bridges, etc.) or to the `ORG` class (i.e. companies, agencies, institutions, etc.).[11] Even though the automatically detected names of facilities and organisations does not necessarily yield only the actors that

[11] See `https://spacy.io/api/annotation#named-entities` for more information on `spaCy`'s named entity classes.

pose a threat to the World Heritage Site,[12] this approach allows us to capture most actors that do pose a threat when they are mentioned in the article.

6.2 Date Parsing

We have used `spaCy`'s statistical entity recognition system to identify contiguous spans of tokens that refer to dates in the text, and label them accordingly as `DATE`. The detected entities are then parsed by the `dateparser`[13] [1] Python package, which parses dates in almost any string format and converts them into `datetime` objects.

The following example is from an article published on 29 June 2018, at around 8pm:

> *In a statement last month, Pingtan rejected specific allegations it had made false statements to East Timor during the licensing process in 2016.*

In it, there are two dates: an absolute date (`'2016'`) and a relative date (`'last month'`). We detect that both dates are temporal expressions with `spaCy`, and reformat them into a machine-readable format with `dateparser`, which takes the publication date of the article as an anchor from which all dates are calculated. In this example, `2016` is converted into `2016-06-29 00:00:00` and `last month` is converted into `2018-05-29 20:10:19`. This is of course an approximation to the date where the event in question truly took place, but serves as an indicator nevertheless.

6.3 Geocoding

Geocoding is the task of detecting mentions of place names in a text and resolving them to their real-world referents. In the following example from our data, a good geocoder should be able to detect that 'Kakadu' is a place, and that its approximate latitude and longitude are -12.67 and 132.47 respectively:[14]

> *Unfinished business: Kakadu needs a new approach to cleaning up an old mine.*

Each article in our dataset has a metadata field that refers to the World Heritage Site that is mentioned in the article. As discussed in Section 3.3, the challenge owners provided us with geographic information about each World Heritage Site, including the latitude and longitude of its centroid and the area in squared kilometres. Given these two variables, we approximated a circle for each World Heritage Site and we derived its radius. We consider as relevant

[12] In the article titled "Critically endangered grey nurse shark mapped for the first time in landmark study" (`https://tinyurl.com/ujdk7s9`), the only organisations mentioned in it (and correctly detected by `spaCy`) are 'the Department of Biological Sciences' and 'Macquarie University'. In the article "Mexico finds illegal avocado plantation in Monarch reserve" (`https://tinyurl.com/wfmfa9n`), the relevant actor 'Association of Avocado Export Packers' is correctly identified as an organization.

[13] `https://dateparser.readthedocs.io/en/latest/`

[14] It is common practice in geocoding to reduce a location to an approximate pair of coordinate points. While this can for obvious reasons be problematic when dealing with places with an extended area (as in the example, where Kakadu refers to the Kakadu National Park, which has an area of 19,804 km2), this is a pragmatic solution that is adapted by most geoparsers.

any location that falls within radius distance from the centroid of the World Heritage Site.

In order to provide approximate coordinates to the places mentioned in the articles, we need a gazetteer (i.e. an external knowledge base containing knowledge of the geography of the world). We used Geonames,[15] a geographical database that is available for download free of charge, and which is widely used for geocoding [22]. Geonames, which is edited and maintained by the online community, contains over 11 million unique locations from around the world that belong to different geographical feature classes, including administrative divisions, water bodies, populated places, roads, land forms, etc.[16] Each location in the database contains a list of names that can be used to refer to it, its latitude and longitude, feature type, and other geographic information.

We created a subgazetteer for each World Heritage Site, containing all locations within approximate radius distance from the centroid of the World Heritage Site (see table 9 for an example). We combined the alternate names of these locations with `spaCy` Named Entity Recognition to identify the relevant locations mentioned in the article and assign geographical coordinates to them. Due to the limited timeframe, we worked on the assumption that ambiguous place names in articles about a World Heritage Site would refer to the location in the vicinity of the World Heritage Site. In the discussion section (Section 8), we provide possible directions to disambiguate place names that can refer to multiple locations.

Table 9: Example rows of locations within approximate radius distance from the centroid of the Galápagos Islands. We only show the relevant columns used in this work: *geonameid* contains the unique Geonames id of the location, *name* is the principal name according to Geonames while *alternatenames* lists potential alternate names this location can be referred to, *latitude* and *longitude* are the coordinates that approximate the position of the location on the Earth's surface, and *fclass* is the type of location, where `T` corresponds to topographical features and `H` corresponds to bodies of water.

geonameid	name	alternatenames	latitude	longitude	fclass
3650144	Punta Wreck	Punta Breck, Punta Naufragio, Punta Negro, Punta ...	-0.92144	-89.62328	T
3650145	Cabo Woodford	Cabo Madera de Vado, Cabo Woodford	-0.76113	-90.78320	T
3650146	Volcán Wolf	Mount Whiton, Volcan Wolf, Volcán Wolf	0.01954	-91.34373	T
3650147	Isla Wolf	Isla Wenman, Isla Wenmans, Isla Wolf, Wenmen Isla...	1.38271	-91.81474	T
3650148	Caleta Webb	Bahia Webb, Bahía Webb, Caleta Webb, Ensenada de ...	-0.75449	-91.38688	H

[15] `https://www.geonames.org/`

[16] See the full list of feature classes here: `https://www.geonames.org/export/codes.html`.

7 Smart Annotation

Labelling articles as relevant or not relevant according to the threat conveyed in them is a costly task. It took ≈40 hours to obtain 1000 labels. In this section, we propose a strategy to ease the annotation process that uses an active learning principle based on heterogeneous uncertainty sampling [12]. The classifier described in Section 5 returns the confidence score alongside the prediction of whether an article is relevant or not. We have created a Jupyter notebook with the `ipyannotate` Jupyter widget for data annotation in which the output of the classification on unseen data is provided to an annotator. The articles are sorted from low to high classification confidence, so that articles for which the classifier was less certain are annotated first. This should force the classifier to learn new more discriminative features that help refine the classification further.

This strategy, which we call smart annotation as it depends on selective human feedback, significantly reduces the number of manual annotations by actively querying the user only when the label assignation is less certain. Figure 5 shows an example of an article in the annotation interface: the article mentions the "Monarch Butterfly Biosphere Reserve" and is titled "Mexico finds illegal avocado plantation in Monarch reserve", and the link to the article is provided for the user to read the content if needed. The article has been identified as containing a threat to the World Heritage Site with a probability of 0.74. The annotation interface allows the user to tag the article as 'threat' (i.e. containinig a threat) or 'no theat' (i.e. not containing a threat). At any point, the annotator can stop labelling data, and the resulting annotations will automatically be evenly incorporated as training and validation data, which can then be used to retrain and reassess our classifier, and to iteratively improve its accuracy.

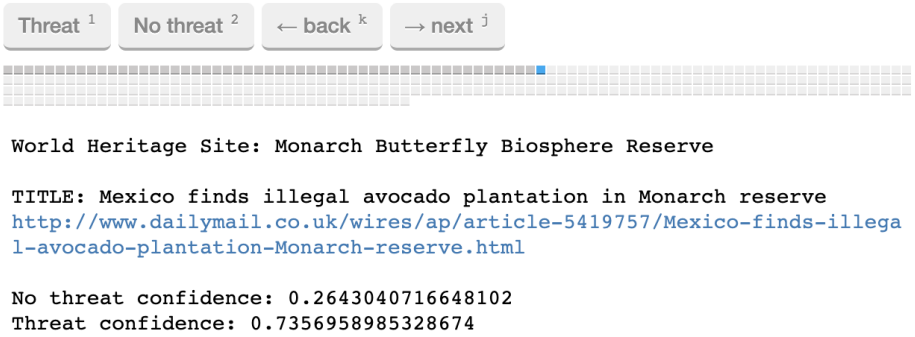

Figure 5: Screenshot of the annotation interface on a Jupyter notebook using the `ipyannotate` widget. The progress bar under the annotation buttons indicates the number of articles already annotated out of the total number of articles to annotate. Optionally, dates, places and organizations mentioned in the article can be printed on screen to assist the annotator choose a label.

8 Discussion

Due to time constraints, not all avenues that were explored made it to the integrated final pipeline. In Section 8.1 we introduce our approach to selecting relevant content from the news article; Section 8.2 describes `Flair` as an alternative to `TextBlob` for sentiment analysis; in Section 8.3 we describe our attempt to disambiguate place names; we propose a strategy to detect threat actors in Section 8.4; and discuss ways of expanding the training set in Section 8.5. Finally, during the Data Study Group week, we experimented with a range of classifiers, such as non-deep-learning classifiers, Recurrent Neural Networks implemented in `TensorFlow`, and transfer learning with the `fast.ai` library. The architecture of these models and their performance are discussed in 8.6.

8.1 Relevant Content Detection

Some articles can be very long, and the World Heritage Site may not necessarily be the focus of the article. As an example, the article titled "100 best movies to watch on Netflix right now"[17] contains the following paragraph:

> *While some are Netflix originals like "To All the Boys I've Loved Before" and "Private Life," others are classic films, like "The Terminator" and "Good Will Hunting." A number of documentaries, like "Virunga," are also in the top 100, as are some Marvel movies, including "Black Panther."*

The article, which mentions "Virunga" referring to the Virunga National Park in the Democratic Republic of the Congo, should be tagged as non-relevant. This is a very long article (above 7,000 words), but the section referring to the World Heritage Site is very short. The great majority of the article is therefore a source of noise that may have a negative impact on the classification of the article. The task of detecting relevant content consists of, given an article, selecting the sections in the article that are related to the World Heritage Site.

We first detected relevant sentences in the article. Here, we consider a sentence to be relevant if it mentions any of the keywords used to query and extract the articles related to the World Heritage Sites (e.g. the keywords used to find articles related to the Waterton Glacier International Peace Park are 'Waterton', 'Glacier International Peace Park', and 'Lakes National Park'), or it mentions any place name referring to a location inside the World Heritage Site or within approximated radius distance (as discussed in Section 6.3). Once the relevant sentences have been identified, we also obtain their context, which in this case is a window of 5 sentences to the left and 5 sentences to the right of the relevant sentence. The relevant sentences with their corresponding contexts are considered as the **relevant content** of the article.

The work described in this section to detect relevant content was not completed in time to integrate it into the pipeline, but it is a preprocessing step that could help other components, such as sentiment analysis or the classification itself.

[17] `https://www.businessinsider.com.au/netflix-top-rated-critics-movies-2019-4`

8.2 Sentiment Analysis

In the final pipeline, we used sentiment analysis as a preprocessing step to the classification task. The choice of the `TextBlob` library to perform sentiment analysis was motivated by a series of factors: our limited timeframe, its ready availability and relatively straightforward use, and the fact that it is implemented in Python. `TextBlob` has a very simplistic approach: it looks up each word in a phrase into a predefined dictionary and averages the polarity of all words in the text. This raises problems with phrases which use multiple emotive words, which are averaged to a neutral score. Ideally, given a token-level approach, phrases which contain as many positive words as it does negative should be labelled as ambiguous rather than neutral, whether sentences without clear positive or negative words (such as 'I commute to work') should be selected as true neutrals. Unfortunately, `TextBlob` is unable to distinguish between these two types of neutrality.

Because of all the observed limitations of using `TextBlob`, we explored an alternative approach to sentiment analysis. We used `Flair`[18] [4] to observe any differences in the sentiment analysis outputs using a pre-trained model to see whether it better reflects the sentiment context of the article heading, description, and overall content of the articles. `Flair` is an embedding-based natural language processing framework introduced by Zalando in 2018. It allows to combine different embeddings together, and use both traditional word embeddings and `Flair` contextual string embeddings. Flair is context-sensitive, and therefore likely to capture more semantic and complex features than the token-based model of `TextBlob`. The library is implemented in Python on top of the popular `PyTorch`[19] [16] deep learning framework. Its latest version (version 0.4) includes a pre-trained sentiment analysis model that is based on the Internet Movie Database (IMDb) movie review dataset. It is a binary sentiment analysis dataset consisting of 50,000 reviews labelled as either positive or negative. Running the pre-trained model gives a value between 0 and 1 along with a sentiment label (i.e. `positive` or `negative`).

`Flair` takes considerably more computing time than `TextBlob`. We did not have time to have a full evaluation for any of the two methods, so we only assessed the quality of the two by manually comparing the two methods on a small set of headlines. `TextBlob` seemed to detect negative sentiments much better than positive sentiments, whereas `Flair` seemed to be worse at detecting negative titles. From the results obtained, it is difficult to conclude which method performs better for the purposes of our task without performing a thorough evaluation. As part of future work, it would be more insightful to apply sentiment analysis to the content of the article that is relevant to the World Heritage Site (as discussed in Section 8.1).

8.3 Geocoding Ambiguous Place Names

While the original intention was to use geocoding as a first filtering step to identify articles that mention locations within close proximity to a World Heritage Site (and therefore avoiding relying solely on limiting keyword-based query

[18] `https://github.com/flairNLP/flair`
[19] `https://github.com/pytorch/pytorch`

searches), the format in which the data was provided to us (with the World Heritage Site as a field given in the metadata) meant that geocoding in our pipeline was not needed as a filtering step. Here, we discuss potential directions to adapt geocoding for filtering geographically-relevant articles.

We approached geocoding as a postprocessing task in which, given an article and the World Heritage Site that is mentioned in it, we identify the different mentions of places situated within a certain distance from the centroid of the World Heritage Site and provide them with coordinates given a gazetteer built for this particular World Heritage Site. A more interesting option would be to use geocoding as a filtering step through which to find articles mentioning a location situated within certain distance of the protected area. However, place names are highly ambiguous (e.g. there are over 25 locations in the world that can be known as 'London', the most prominent ones being the English capital and the city in Ontario), and we must disambiguate them (i.e. identify the right entity referred to by a mention) before resolving them to their geographic coordinates.

We tested two different options[20] for disambiguating place names:

- The **Edinburgh Geoparser**[21] [21] is a widely-used geoparser software package that is implemented in modules, as a sequence of steps arranged in a pipeline,[22] allowing switching off unrequired modules and tweaking the pipeline. We adapted the geoparser by having our own place name recognizer based on `spaCy` and our own gazetteer based on `Geonames`. We tested tweaking the disambiguator module to favour locations that are closer to World Heritage Site centroids[23], but this did not seem to provide better results, even though we did not have time to perform a full evaluation. One of the `Edinburgh Geoparser`'s disadvantages is its slowness.

- The **Mordecai** geoparser is a Python library that extract the place names from a text, resolves them, and returns their coordinates and structured geographic information. Mordecai requires a running Elasticsearch service with Geonames in it. Based on observation, results were disappointing and generally worse than those of the `Edinburgh Geoparser`.

In our scenario, in which the World Heritage Site mentioned in the article was part of the metadata, disambiguating towards the closest location to the centroid of the World Heritage Site provided the best results. In future scenarios in which we may not have the information on which World Heritage Site is mentioned in the article, toponym disambiguation will be a required step before geocoding.

[20] Other potential options to geoparsing (such as the Google Geocoding API (`https://developers.google.com/maps/documentation/geocoding/start`)) were dismissed because they required an Internet connection, which we initially did not have. We will explore this option as part of future work.

[21] `https://www.ltg.ed.ac.uk/software/geoparser/`

[22] `http://groups.inf.ed.ac.uk/geoparser/documentation/v1.1/html/pipeline.html`

[23] `https://programminghistorian.org/en/lessons/geoparsing-text-with-edinburgh#giving-preference-to-a-geographical-area`.

8.4 Actors Detection

During the Data Study Group week, the challenge owners provided us with lists of relevant keywords to WWF, among which names of environmental organisations and of known actors involved in environmental threats (such as power plants, oil and gas assets and concessions, global ports and airports, and hydroelectric dams). As discussed in Section 6.1, we used `spaCy` Named Entity Recognition to identify the mentions of organisations or facilities mentioned in the articles. This was a pragmatical solution that did not involve any training or preprocessing. We considered during the Data Study Group week a more tailored approach to the problem of identifying actors, by using `spaCy` to train a new named entity recognizer with custom entities, which in this case would be (1) environmental organisations and (2) actors causing a threat,[24] learnt from sentences in which the keywords provided by WWF are mentioned.

8.5 Retrieving Related News

For the purpose of expanding the training set, it could be useful to find news articles associated with a certain event within a certain time window. We implemented an event detection module[25] that takes the `spaCy`-processed vectorised representations of the headlines of the news articles (we used only articles in English) as input. Then we grouped the headlines using the clustering algorithm DBSCAN[26] from the machine learning library `scikit-learn`[27] [17]. We used the `epsilon` (i.e. the maximum distance between two samples for one to be considered as in the neighborhood of the other) that produced the largest number of clusters.

These are some example of clusters:

1. (a) "Australia Says Great Barrier Reef Has 'Very Poor' Outlook, Climate Change To Blame"
   (b) "Great Barrier Reef now has 'very poor' outlook due to climate change"
   (c) "Australia lowers outlook for Great Barrier Reef to 'very poor"'
   (d) "Great Barrier Reef 'now close to collapse' thanks to climate change - The Times"
   (e) "Australia downgrades outlook for Great Barrier Reef to 'very poor"'
   (f) "Australia lowers Great Barrier Reef outlook to 'very poor"'
2. (a) "Cow walks on wild side with Polish bison"
   (b) "Adorable Cow Says Screw It! And Runs Away To Live With Wild Bison"

[24] See `https://spacy.io/usage/training#example-new-entity-type` for the official guide on how to train a new entity type.

[25] We implemented event detection following the procedure described in this blog post: `https://towardsdatascience.com/natural-language-processing-event-extraction-f20d634661d3`

[26] `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html`.

[27] `https://scikit-learn.org/stable/`

By extracting the publication date of the news article from the metadata, we can also follow the timeline of the event in the news (see Figure 6 for a visual representation of the timeline of the event in table 10):

Table 10: Tracking of an event through time: this table is a cluster of articles detected as belonging to the same event (i.e. the spread and extinction of California's largest wildfire), sorted by the date of publication of the article.

Date	Headline
2018-08-07 17:45:35	California's Mendocino Complex fire just became the largest wildfire in state history
2018-08-07 17:52:09	California's Mendocino complex of fires now largest in state's history
2018-08-13 18:00:00	The Mendocino Complex Fire Has Now Spawned the Biggest Single Blaze in California History
2018-09-19 11:30:00	The Largest Bushfire In California's History Is Finally Out
2018-09-19 20:40:00	The Largest Wildfire in California's History Is Finally Out
2018-09-21 19:32:41	This summer's wildfires were the largest in California history

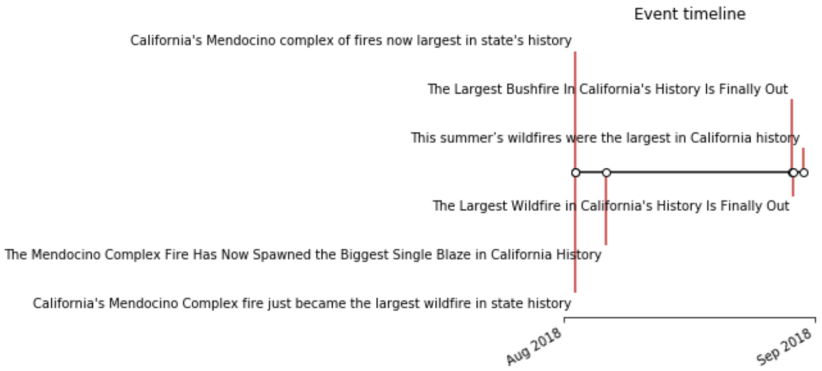

Figure 6: Tracking of an event through time.

This work was not completed in time to be part of the final pipeline. However, it would be interesting to add it as part of the smart annotation tool: given an article that has been positively tagged as relevant, this will allow seeing if previous and subsequent articles referring to the same event have also been tagged as relevant, and change its label if necessary.

8.6 Baseline Classifiers

8.6.1 Non-deep-learning Models

We have trained multiple non-deep-learning models as baselines for the classification task in Section 5. These classifiers have several advantages. First, they perform well even when not large training data is provided. Second, efficient and easy-to-use implementations of these algorithms exist and have been widely used by the ML community, such as the `scikit-learn` (`sklearn`) library used in this study. We built a pipeline to easily test different classifiers including a random forest classifier, multinomial logistic regression, and support vector classifier (SVC). Due to the modular design of the `sklearn` library, further classifiers could be readily added to the pipeline.

The input features can be a combination of tokenised texts, sentiments, and topics extracted from each news article. Three different ways to vectorize the text into a matrix of features have been implemented, by using pretrained `word2vec` vectors [13], token-counts or TFIDF (term frequency–inverse document frequency). In both token-counts and TFIDF methods, the text is converted into a sparse vector while `word2vec` vectors are dense with dimensionality of 300[28]. When working with document-level text, each 300-dimensional vectors for tokenized words are averaged to produce one vector.

The pipeline also includes a random grid search for hyperparameter tuning. We used the `sklearn RandomizedSearchCV` function to loop over various parameters to fine-tune the models according to a given number of iterations (20 in our experiments). It calculates the 5-fold cross validation score of each of the models and outputs a plot of the accuracy across the five folds. The model with the highest F1 score is chosen as best model. We also tried the `GridSearchCV` function to tune the models, however this was too computationally intensive and the approach was abandoned.

After selecting the hyperparameters in the previous step, we fit the model on the training set and run the model on the validation set. In the last step, the pipeline returns the target metrics (recall, precision and F1 score), and it plots the confusion matrix. Based on our experiments, the logistic regression model had the best performance, as measured by precision and recall among our non-deep-learning baseline classifiers.

8.6.2 Classification Using RNN and `tensorflow`

As one of our baselines, we implemented an RNN network in `TensorFlow`[29] [3]. In this model, the article contents were tokenized up to trigrams and were used in both training and validation sets. Vectors from a pretrained `Word2Vec` model were used to embed the article contents.

We also tried to balance the labels in our training and validation sets by oversampling the underrepresented classes. We oversampled the positive (relevant) class using random sampling with replacement, such that the number of labels of the two classes is the same, that is, 547 not-relevant articles and 547 oversampled relevant articles (from 152 articles originally labelled as relevant).

[28] We use the pretrained GoogleNews word2vec vectors from `https://code.google.com/archive/p/word2vec/`

[29] `https://www.tensorflow.org/`

Unfortunately, the oversampled training dataset did not improve the results. In what follows, we will report the results on the original, imbalanced dataset.

The `TensorFlow` implementation of the Bidirectional Long Short-Term Memory (LSTM) model is as follows:

```
model_tf = tf.keras.Sequential([
    tf.keras.layers.Embedding(
        input_dim = model_word2vec.wv.vectors.shape[0],
        output_dim = model_word2vec.wv.vectors.shape[1],
        input_length = 200,
        weights = [model_word2vec.wv.vectors],
        trainable=False),
    tf.keras.layers.Bidirectional(\
        tf.keras.layers.LSTM(128,
        recurrent_dropout=0.1)),
    tf.keras.layers.Dropout(0.25),
    tf.keras.layers.Dense(64),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

with binary cross-entropy as the loss, minimized using the Adam optimization algorithm [11]:

```
model_tf.compile(loss='binary_crossentropy',
    optimizer=tf.keras.optimizers.Adam(),
    metrics=['accuracy'])
```

We experimented with different model setups as listed in Table 11. Unfortunately, the results on all metrics are significantly lower than what the BERT model could achieve (see Section 5 for discussion). This was still the case after changing the model architecture to `GRU` with the following implementation:

```
model_tf2 = tf.keras.Sequential([
    tf.keras.layers.Embedding(
        input_dim = model_word2vec.wv.vectors.shape[0],
        output_dim = model_word2vec.wv.vectors.shape[1],
        input_length = 500,
        weights = [model_word2vec.wv.vectors],
        trainable=False,
        batch_input_shape=[300, None]
    ),
    tf.keras.layers.GRU(1024,
        return_sequences=False,
        stateful=False,
       recurrent_initializer='glorot_uniform'),
    tf.keras.layers.Dropout(0.25),
    tf.keras.layers.Dense(64),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

8.6.3 Transfer Learning with `fast.ai`

Transfer learning is a technique where a pre-trained model is used and fine-tuned on training data instead of generating a new model from scratch. We have already used this concept in Section 5 where a set of pre-trained BERT models were fine-tuned for the binary classification task using the HuggingFace

Table 11: Performance of an RNN model implemented in TensorFlow on the binary classification task (*lr* is learning rate).

Model	lr	Embed. size	Epochs	Precision	Recall	F1
1	0.001	200	40	49%	40%	44%
2	0.0001	500	20	**65.3%**	25.16%	37.36%
3	0.0001	500	40	55.1%	**49.23%**	**52.03%**
4	0.0001	500	60	51.7%	46.15%	48.78%

library. Another widely used library is the `fast.ai`[30] [8] which provides a convenient set of tools to efficiently fine-tune a set of pre-trained models on the problem at hand [9]. Here, we used the AWD-LSTM model architecture with pre-trained word embeddings from *Wikitext 103*.

Our training consists of the following steps: (1) load in training and validation sets, preprocess the data (tokenize using *fast.ai*'s built-in tokenizer, lowercase, remove NA values) and create a DataBunch object; (2) Training the classifier with `dropout = 0.5` and `weight decay = 0`. We used the one cycle policy [20] for optimization. After the first fine-tuning step, we unfreezed all the layers and trained the whole model using a lower learning rate. This work is in progress, and similar to Section 5.2.3, we expect improvements in the performance metrics when adding additional features (e.g. sentiment analysis and topics extracted by unsupervised LDA).

9 Further Work

The work achieved during the Data Study Group week has opened several avenues. In Section 8, we have discussed some of the strands of work that could not be integrated into the final pipeline due to time limitation or lack of proper evaluation. They are points worth pursuing as part of future work.

In particular, using geolocation in the articles content not only as a postprocessing step, but also as a filtering mechanism to identify relevant news stories that occur anywhere in the protected area (especially in articles in which the name of the protected area is not mentioned) is an important next step: this can potentially widen the scope of our threat detector in the news significantly. In order to improve the coverage, we will also explore the use of WWF's commercial datasets to help locate relevant actors within sites. The classifier is likely to be improved by the inclusion of the additional keywords provided by WWF describing relevant assets (see Section 3.3).

We plan to scale this work to a wider group of conservation assets (250,000+) and provide open access to the relevant geolocated news stories. This would allow news stories to be tagged within WWF-SIGHT, and would allow WWF staff to click and quickly consider relevant news stories in their areas of operation. This will, hopefully, support the wider conservation community and provide greater transparency on the stakeholders driving biodiversity loss.

In terms of infrastructure, one of the limitations that was faced during the preparation for the Data Study Group was that Google News API has a limitation of 250,000 term searches with a maximum return of 100 records. We want

[30] https://www.fast.ai/

to explore different methods to overcome this which include exploring other APIs to extract news articles and find more local stories.

It would be also extremely interesting to be able to find relevant articles from other languages, in order to have access to news that are more immediate to the protected areas in places where the main language is not English. Even though our approach is largely language independent, it requires using some language-specific processing functions and manually-annotated labelled data, for which we would like to explore an approach that combines machine translation with our smart annotation strategy (see Section 7).

Finally, due to the intense and multi-strand nature of the challenge, the codes developed during the Data Study Group week need more work to be publishable as a package. There are sections that due to time constraints could not be properly evaluated. In the future, we will package the existing codes into a user-friendly product for WWF and release it under an open-source licence.

10 Team Members

- **Kasra Hosseini** (The Turing Co-PI). Kasra is a research data scientist at the Alan Turing Institute. He is interested in artificially intelligent systems, machine learning, data analysis and visualisation.
- **Mariona Coll Ardanuy** (The Turing Co-PI). Mariona is a computational linguist at the Alan Turing Institute. Her research interests lay in the intersection between the humanities and language technology.
- **David J. Patterson** (Challenge Owner).
- **Laura García-Vélez** (Challenge Owner).
- **Leonardo Castro-Gonzalez**. Leonardo is a PhD student at the University of Birmingham. He is interested in Urban Analytics, Complex Systems, and its application to Economics and Urban Scaling Laws. For the WWF project, Leonardo worked mainly in the sentiment analysis and in the postprocessing work.
- **Lucas Deecke**. Lucas is a PhD student in machine learning at the University of Edinburgh, studying themes around multi-domain learning and the topology of deep learning architectures. For the group he worked on end-to-end finetuning of BERT for sequence classification.
- **Dan Saattrup Nielsen**. Dan is a PhD student in Mathematical Logic at the University of Bristol. He studies virtual large cardinals, being axioms of large infinities to add to ZFC, the defacto standard foundation of Mathematics. He was facilitating the group, which entailed maintaining an overview, making sure that everything is on track, helping out with NLP methods and libraries, as well as working with Anton on the Latent Dirichlet Allocation topic modelling.
- **Selina Cho**. Selina is a PhD student in Cyber Security at the University of Oxford. Her work focuses on online game cheating, and understanding its ecosystem from the notion of self-governance. In the DSG, Selina participated in the sentiment analysis using TextBlob and Flair
- **Lesley Dwyer**. Lesley is a recent graduate with an MSc in Data Science from City, University of London. During the DSG, she worked on feature extraction using Word2Vec and helped with the baseline scikit-learn classifiers.
- **Victoria Auyeung**. Victoria is a PhD student in Metabolic and Cardiovascular Disease at the University of Cambridge. Her work focuses on using genomics and metabolomics to understand mechanisms of complex diseases. In the DSG, Victoria worked on feature extraction and recursive neural networks in fast.ai.
- **Amy Krause**. Amy is a Data Architect at EPCC, The University of Edinburgh. She is a research software and data engineer and works with domain experts to architect and implement software solutions for large-scale data intensive applications. She is a key architect for the Edinburgh International Data Facility (EIDF). As part of the DSG, Amy evaluated geoparsing tools, created balanced training and validation data for modelling, and implemented the event extraction.

- **Alejandro Coca-Castro**. Alejandro is a PhD student in Geography at King's College London. His work focuses on modelling the following land cover and land-use over deforested areas in the tropics usig large volumes of high temporal satellite data and machine learning. In the DSG, Ale-jandro helped in tokenization techniques, augmenting environmental- and development-based keywords, co-leading a framework of different scikit-learn classifiers and a general support to Alina's approach using a RNN-based classifier in TF v2.0.

- **Reka Vonnak**. Reka is an MSc student in Urban Analytics at the University of Glasgow. Her research interests focus on urban data science and spatial analysis. During the Data Study Group she worked on pre-processing, tokenization techniques, and recursive neural networks with Fast.ai.

- **Anton Baleato Lizancos**. is a PhD student in Cosmology at the University of Cambridge. His research explores ways in which the gravitational lensing of cosmic microwave background light can be used to test the cosmological model and fundamental physics. As part of the DSG, An-ton contributed some low-level pre-processing functions, co-led the topic modelling pipeline with Dan, and helped with the Active Learning efforts led by Kasra.

- **Alina Miron**. Alina is an Associate Lecturer at Brunel University London. Her work focuses on machine vision and machine learning for modelling and predicting human motion in rehabilitation exercises. As part of DSG, Alina worked on various data preprocessing tasks, with a focus on text preprocessing, and also developed the RNN-classification approach based on TensorFlow and Word2Vec.

- **A. Barbara Metzler**. is a PhD student in Machine Learning and Global Health at Imperial College London. Her current work focusses on map-ping poverty with high resolution satellite imagery and machine learning. During the DSG she helped with creating two different vectorizations and creating a framework for the different sklearn classifiers.

- **Katriona Goldmann** is a PhD student in computational biology at Queen Mary University London. Her research uses statistical and machine learning methods to create multi-omic data integration models in auto-immune diseases with the aim of moving towards stratified medicine. For the WWF project, she performed data preprocessing and conducted sentiment analysis using TextBlob.

References

[1] dateparser – Python parser for human readable dates. `https://dateparser.readthedocs.io/en/latest/`. Accessed: 2019-12-10.

[2] Natural Language Basics with TextBlob. `http://rwet.decontextualize.com/book/textblob/`. Accessed: 2019-12-10.

[3] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[4] A. Akbik, D. Blythe, and R. Vollgraf. Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649, 2018.

[5] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[6] 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.

[7] M. Honnibal and I. Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. 2017.

[8] J. Howard et al. fastai. `https://github.com/fastai/fastai`, 2018.

[9] J. Howard and S. Ruder. Universal language model fine-tuning for text classification, 2018.

[10] K. S. Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 1972.

[11] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[12] D. D. Lewis and J. Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Machine learning proceedings 1994*, pages 148–156. Elsevier, 1994.

[13] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[14] S. Nakatani. Language detection library for java, 2010.

[15] D. Newman, J. H. Lau, K. Grieser, and T. Baldwin. Automatic evaluation of topic coherence. In *Human language technologies: The 2010 annual conference of the North American chapter of the association for computational linguistics*, pages 100–108. Association for Computational Linguistics, 2010.

[16] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

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

[18] R. Řehůřek and P. Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

[19] M. Röder, A. Both, and A. Hinneburg. Exploring the space of topic coherence measures. In *Proceedings of the eighth ACM international conference on Web search and data mining*, pages 399–408, 2015.

[20] L. N. Smith. A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay, 2018.

[21] R. Tobin, C. Grover, K. Byrne, J. Reid, and J. Walsh. Evaluation of georeferencing. In *Proceedings of the 6th Workshop on Geographic Information Retrieval (GIR'10)*, Zurich, Switzerland, 2010.

[22] D. Weissenbacher, A. Magge, K. O'Connor, M. Scotch, and G. Gonzalez. Semeval-2019 task 12: Toponym resolution in scientific papers. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 907–916, 2019.

[23] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew. Huggingface's transformers: State-of-the-art natural language processing, 2019.

turing.ac.uk
@turinginst