Deep Learning for Sentiment Analysis of News Articles

Bradley Fowler - Emmanuel College

May 28, 2018

Technical Abstract

Background

Approximately 90% of all the news published are negative in their sentiment, usually because negative stories allow for a more eye-catching headline that grabs the reader's attention and gets the click. This has two undesirable consequences. Firstly, genuinely interesting articles are hidden away and missed entirely. Secondly, repeated exposure to negative news has proven detrimental psychological effects. An explicit example of the former occurred on the day of the Paradise papers; there was overwhelming coverage of how Lewis Hamilton, the Queen, and many others avoided paying tax. Yet no-one heard about the NHS trialling smartphone GP appointments, Facebook messenger allowing payments between friends, or Scotland offering free abortions to women in Northern Ireland, all published on the same day. This highly relevant coverage was overshadowed by hundreds of articles about tax evasion, where just one would have been sufficient.

This project aims to remove negative click bait to address the aforementioned problems and restore the balance of sentiment in news. This amounts to a two part solution. Firstly, an algorithm that is capable of identifying and filtering out these articles must be designed and trained. Secondly, a platform that displays live news articles post filtering must be built to help reduce peoples exposure to negative click bait.

Methodology

The filtering algorithm performs binary classification, with the articles of very negative sentiment that will be discarded forming one class, 'hide', and the remaining articles forming the 'show' class. For this method, prediction errors are not necessarily correlated to any specific sentiment values, but instead focused around articles that exhibit features from both the 'hide' and 'show' class. In user perception terms, we care less about small errors in sentiment prediction for articles far from the threshold than for those around it.

This project explores the novel approach of combining both text and image features of sentiment to give an improved classification accuracy. This approach has not been explored in the literature and will require three distinct architectures to be implemented; a text, image and then ensemble classifier.

A Recurrent Neural Network (RNN) was trained on news article summaries to extract sentiment based text features, while a Convolutional Neural Network (CNN) was trained on the corresponding news article's image to extract sentiment based image features.

Various techniques were applied to these networks to reduce overfitting and increase their performance, with this report analysing these techniques to explain why they work. The text and image classifiers were then combined together to form an ensemble classifier that could yield a prediction of higher accuracy, with a lower bound on performance that is equal to the best accuracy achieved by either the text or image classifier.

All three architectures consist of a supervised learning model and so their performance is heavily determined by the quality of the dataset used for training. Since there is no existing dataset for news article sentiment available in the literature, a custom built one was created.

To address the final part of the aim which is to 'help reduce peoples exposure to negative click bait', a website that runs the filtering algorithm on live news articles was created. Through extensive usability tests, a number of key features were added to help ensure it truly fulfilled this aim and the resulting platform can be found at justnews.io. The website was also a crucial part to the construction of a custom dataset, since it provided the platform for people to classify articles.

Conclusions

This project brought together both image and text features through an ensemble classifier, proving that there is independence between these features and combining them yields an increase in performance. It developed a method for augmenting textual data, where for this problem augmenting nouns, verbs and adjectives with highly relevant words of the same type improves performance. The investigation into challenging datapoints that caused erroneous predictions demonstrated appropriate behaviour for both individual text and image classifiers, but highlighted the difficulty in combining just two classifiers as the strength of their outputs are difficult to compare when no constraints are placed on them during training. An overall accuracy of 85% was achieved for this algorithm, a good result for a challenging task with many debatable borderline cases.

The second part of the problem was addressed through a website named Just News. The platform successfully enabled the efficient construction of a high quality dataset consisting of 1636 datapoints. The homepage, shows a constant stream of live news articles from 12 popular providers with realtime filtering using the developed algorithm. It boasts a range of features to help reduce peoples exposure to negative click bait including: topic filtering, search bar, trending items, news provider filtering, share buttons and notifications. A series of user tests were undertaken to identify these features and therefore demonstrate a certain level of effectiveness in addressing a solution to the problem.

Taken together, this project fulfils its aim of developing a comprehensive solution to negative click bait in the news, reducing peoples exposure by 85%.

Contents

1 Introduction

1.1 Motivation

Approximately 90% of all the news published are negative in their sentiment, usually because negative stories allow for a more eye-catching headline that grabs the reader's attention and gets the click. This has two undesirable consequences. Firstly, genuinely interesting articles are hidden away and missed entirely. Secondly, repeated exposure to negative news has proven detrimental psychological effects. An explicit example of the former occurred on the day of the Paradise papers; there was overwhelming coverage of how Lewis Hamilton, the Queen, and many others avoided paying tax. Yet no-one heard about the NHS trialling smartphone GP appointments, Facebook messenger allowing payments between friends, or Scotland offering free abortions to women in Northern Ireland, all published on the same day. This highly relevant coverage was overshadowed by hundreds of articles about tax evasion, where just one would have been sufficient.

1.2 Aim

This project aims to remove negative click bait to address the aforementioned problems and restore the balance of sentiment in news. Pictorially this is shown below in Figure 1, where a filtering threshold can be set to give a ratio of negative to non-negative news articles.

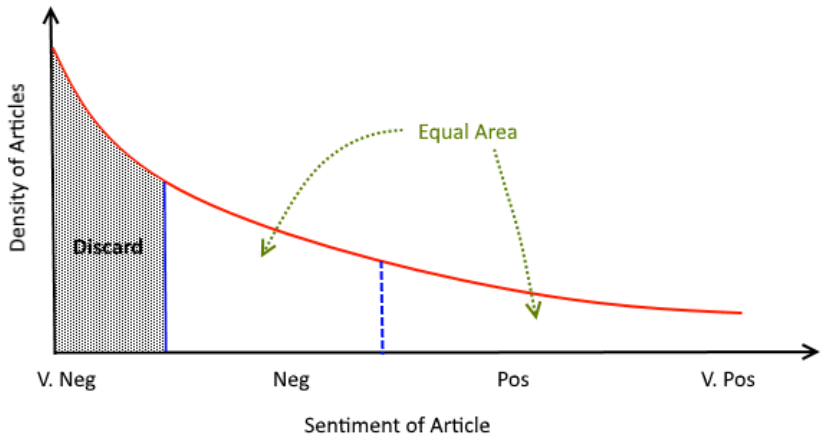


Figure 1: Data Filter

This requires a two part solution. Firstly, an algorithm that is capable of identifying and filtering out these articles must be designed and trained. Secondly, a platform that

displays live news articles post filtering must be built to help reduce peoples exposure to negative click bait.

For this solution, an objective value of the sentiment of an article must exist. This will be true for sensible bounds if sentiment is defined as follows:

The feeling towards the underlying matter which an article is about or if neutral, the effects described by said article.

A few examples are shown in Table 1 below to help clarify this definition.

	Example Statement	Sentiment	Reasoning
•	Three times rapist Person A due in court next Monday	V. Negative	Subject matter
•	Labour wins the general election	Neutral	Neutral effects described
•	Terrible day for UK with Labour win, god help the economy	Negative	Negative effects described

Table 1: Example Statements and their Sentiment

1.3 Performance Measures

There are two intuitive methods that can be employed to execute this aim.

Firstly, a regression based model can be trained to predict the precise sentiment of each news article. This will generate an even spread of prediction errors across the full range of sentiment classes, since the loss function will penalise the distance the predicted sentiment deviates from the true value. Articles with a predicted sentiment value below a certain threshold will then be discarded to complete the filtering operation.

The second method is to perform binary classification, with the articles being discarded forming one class, 'hide', and the remaining articles forming the 'show' class. For this method, prediction errors are not necessarily spread evenly across all sentiment values, but instead focused around articles that exhibit features from both the 'hide' and 'show' class.

For the regression based model, increasing the threshold will ultimately show only 'Very Positive' articles. However, this is undesirable since the model will be discarding lots of relevant non-click bait articles. The binary classification model does not suffer from this problem since increasing the threshold displays articles with greater certainty of not being from the click bait class. In user perception terms, we care less about small errors in sentiment prediction for positive articles than for those around the threshold. Thus, this project adopts the latter method of binary classification.

1.4 Background Literature Review

Sentiment analysis is a well studied field, starting from a simple bag-of-word model studied in the 20^{th} century to more recent research showing state-of-the-art performance achieved by Recurrent Neural Networks (RNN) at Stanford University[1]. Prior research define sentiment into two classes, 'positive' and 'negative'. Although this leads to higher quoted performance values, the technique discards many datapoints which do not fall into either classification. This limitation was stated by Alec Go et al in [2] and renders the trained algorithms useless for most practical purposes, since live streams of data will contain many points that do not fall into either of the classes. This project improves on this by using the aforementioned binary classification method that categorises the entire spectrum of news articles.

All existing methodologies for sentiment analysis use supervised learning and so their performance is heavily determined by the quality of the dataset used for training. The most common type of dataset used in the literature are only pseudo-sentiment ones, which are generated automatically from reviews and make the assumption that a review of a high rating will contain a phrase positive in sentiment, with the converse being true for low ratings. They have been constructed for amazon reviews[4][5] and movie reviews[1]. A sentiment dataset also exists for twitter[6], where tweets are classified based on if they are positive or negative about a company of interest. However, it is not clear if any of these datasets will be related to the sentiment analysis of news articles, where there exist no publicly available datasets.

Recent research indicates the most promising results for sentiment analysis are achieved by deep learning approaches, with the University of Colorado using RNNs and sentiment analysis to help identify fake news[3]. All of the approaches to-date assume sentiment is determined by one sensory input however, this project will improve on that by investigating the novel approach of combining both text and images together through an ensemble classifier to make a better prediction.

2 Sentiment Analysis

2.1 Algorithm Methodology

When constructing the dataset by classifying news articles, it was identified that for many cases the image associated with each news article was a strong indicator, and sometimes determining factor, as to which sentiment classification an article should be given. The image associated with an article is an otherwise unused dimension of the input and so to exploit this, an architecture designed specifically for image classification will be implemented alongside a text classifier. The text and image classifiers will then be combined together to form an ensemble classifier that can yield a prediction of higher accuracy, with a lower bound on performance that is equal to the best accuracy achieved by either the text or image classifier. All three architectures will consist of supervised learning models and make use of a custom built dataset discussed in the next section. The following subsections outline these models in depth along with the reasons behind their design.

2.1.1 Text Classification

2.1.1.1 RNN LSTM Architecture

The Recurrent Neural Network Long Short Term Memory (RNN LSTM) Architecture was first introduced by Gers in his PhD thesis[8]. It was adopted for this project for the following reasons:

1. The structure of the LSTM unit enables it to bridge long time lags, an important feature of sentiment which is determined by a complex arrangement of words.
2. RNN LSTM networks achieve state-of-the-art results for many sequential problems such as sentiment analysis and machine translation.
3. RNNs make use of word-to-vector embeddings, allowing for an intuitive insight into their operation.

The RNN takes as its input an arbitrary length sequence of vectors $\{\mathbf{x}_i\}_{i=1}^{T}$, processing them sequentially to generate a series of latent variables $\{\tilde{\mathbf{h}}_t\}_{t=1}^{T}$. For binary classification, the final latent variable $\tilde{\mathbf{h}}_T$ passes through a fully connected layer with a sigmoid activation function to produce a scalar output $y = \sigma(\mathbf{v}^T\tilde{\mathbf{h}}_T + b)$ contained within the range $[0, 1]$. The unrolled diagram view of this is shown by Figure 2 and expression (1).

$$predicted\ class = \begin{cases} show & \text{if } y > 0.5 \\ hide & \text{otherwise} \end{cases} \qquad (1)$$
$$where\ \ \sigma(u) = \frac{1}{1+e^{-u}}$$

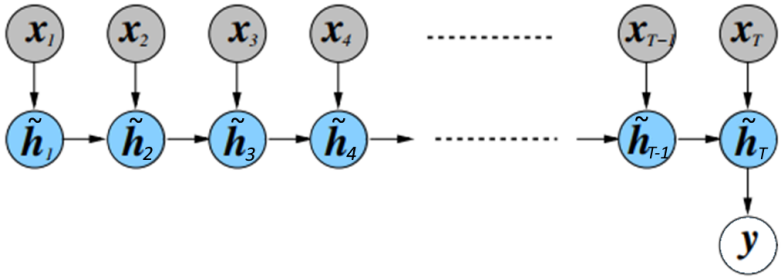


Figure 2: RNN Architecture

Mathematically, this relationship can be written in terms of a non-linear function g, which describes the behaviour of the RNN cell used by the network. For LSTM, the latent variable vector is the concatenation of two distinct parts, a history vector $\mathbf{h}_t$ and memory vector $\mathbf{c}_t$. $\tilde{\mathbf{h}}_t = (\mathbf{h}_t, \mathbf{c}_t)$.

$$\tilde{\mathbf{h}}_t = g(\mathbf{x}_t, \tilde{\mathbf{h}}_{t-1}) \qquad (2)$$

A visual representation and the equations governing an LSTM cell are shown by Figure 3 and expressions (3) & (4) respectively

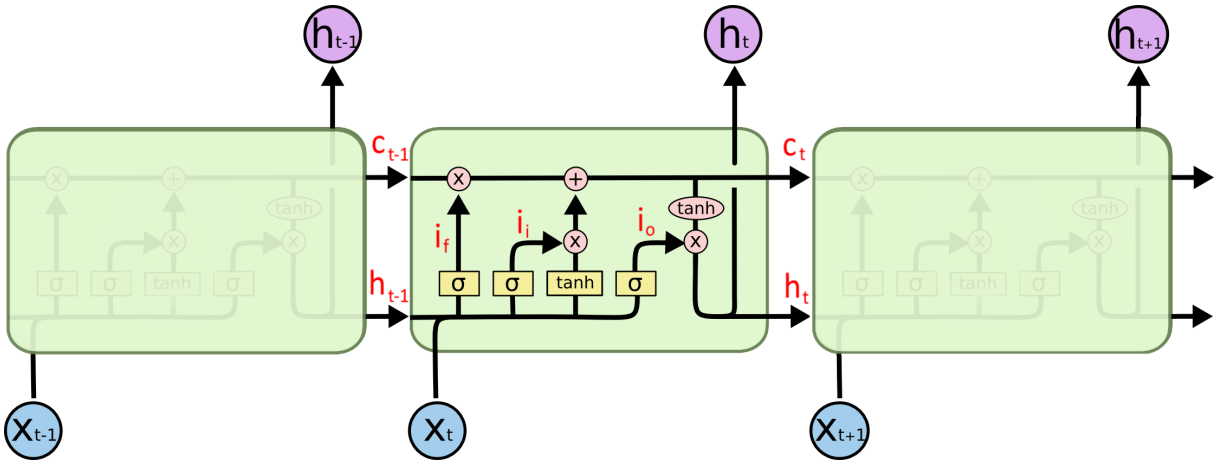


Figure 3: LSTM Unit[9]

$$
\begin{aligned}
\boldsymbol{i}_i &= \sigma(W_i^f \mathbf{x}_t + W_i^r \mathbf{h}_{t-1} + W_i^m \mathbf{c}_{t-1} + \mathbf{b}_i) \\
\boldsymbol{i}_f &= \sigma(W_f^f \mathbf{x}_t + W_f^r \mathbf{h}_{t-1} + W_f^m \mathbf{c}_{t-1} + \mathbf{b}_f) \\
\boldsymbol{i}_o &= \sigma(W_o^f \mathbf{x}_t + W_o^r \mathbf{h}_{t-1} + W_o^m \mathbf{c}_t + \mathbf{b}_o)
\end{aligned} \tag{3}
$$

$$
\begin{aligned}
\mathbf{c}_t &= \boldsymbol{i}_f \odot \mathbf{c}_{t-1} + \boldsymbol{i}_i \odot tanh(W_c^f \mathbf{x}_t + W_c^r \mathbf{h}_{t-1} + \mathbf{b}_c) \\
\mathbf{h}_t &= \boldsymbol{i}_o \odot tanh(\mathbf{c}_t)
\end{aligned} \tag{4}
$$

σ and $tanh$ functions compute element wise and return a vector.
$\odot$ represents element-wise multiplication between vectors.
Connections between $\mathbf{c}_{t-1}$ and gate inputs are not shown on the diagram.

The core of the LSTM cell is the memory unit $\mathbf{c}_t$, which is the vector that maintains a running store of the important features identified up to time point t. The input gate $\boldsymbol{i}_i$, and forget gate $\boldsymbol{i}_f$, are responsible for the addition and removal of data from the memory cell respectively. The output gate $\boldsymbol{i}_o$, controls the ratio at which data is extracted from the memory unit $\mathbf{c}_t$ and placed into the history vector $\mathbf{h}_t$.

There are 11 weight matrices and 4 bias vectors that optimise this process. These are updated via back-propagation, the derivation for which is shown in Section A.1 of the Appendix.

The size of the latent variable vector and dimensionality of the input completely determine the number of model parameters, calculated to be $l(7l + 4d + 4)$ where d is the dimensionality of the input and l the length of either the history vector or memory vector, noting that they have to be the same size. d and l were chosen such that the number of model parameters is less than the number of datapoints to reduce overfitting and improve generalisation. The exact values are described in the next section.

2.1.1.2 Word-to-Vector Embedding

The RNN relies on vector inputs; a standard way of achieving this for text data is through a word-to-vector embedding. Every word in an article is one-hot encoded (a vector of zeros with 1 at a single position), with words not contained in a pre-specified vocabulary being assigned the all zero vector. Mathematically this can be written in the form given in (5), where $\mathbf{s}_i$ represents the one-hot encoding of word i, $\mathbf{x}_i$ is the word vector in the embedded space, and W the embedding matrix to be learned.

$$\mathbf{x}_i = W\mathbf{s}_i \tag{5}$$

The vocabulary is determined by taking the most common words within the dataset up to a maximum number of words. The embedding dimension determines the complexity of relationships that can be modelled between words. The size of W is equal to the product of the two, which for a modest vocabulary size of 400,000 and embedding dimension of 50, gives 20 million model parameters in the embedding matrix alone. The dataset used in this project has far fewer datapoints, such that if W were to be learned then the model would be highly susceptible to overfitting and not generalise well. Therefore, to reduce the number of model parameters, a transfer learning approach has been implemented that utilises a pre-trained embedding.

Using the Word2Vec unsupervised learning algorithm[11], Google trained an embedding of 3 million word vectors each with a dimensionality of 300 from its own news dataset[12]. This would be ideal, however there are two main problems with this embedding. Firstly, the word vectors matrix is large at 3.6 GB, requiring an excessive amount of RAM as this matrix will need to be stored multiple times. Secondly, for an embedding of dimension 300 the number of model parameters for an LSTM size of $l = 20$ is 26,880; still far too big to have good generalisation with the dataset used in this project.

Thus, a more manageable matrix embedding trained by Stanford University[13] that contains only 400,000 word vectors, each with a dimensionality of 50 will be used. Choosing $l = 20$ sets the number of model parameters to be 6,880, which is less than the number of augmented datapoints.

A 2D t-distributed stochastic neighbour embedding (t-SNE) visualisation of this embedding can be seen below in Figure 4,

2.1.1.3 Dropout

Another way to improve generalisation is through dropout, when random nodes are turned off (output set to zero) during each training step. This helps to reduce the overspecialisation of neurons and encourage the discovery of multiple identifying features for each class. This project followed the variational method outlined in Z. Ghahramani's paper[14], which improves on the naive implementation of using different masks at each time step, with no dropout on the recurrent layers.

Variational inference based dropout uses the same dropout mask at each time step, including the recurrent layers. When used with discrete inputs, as is the case with this project

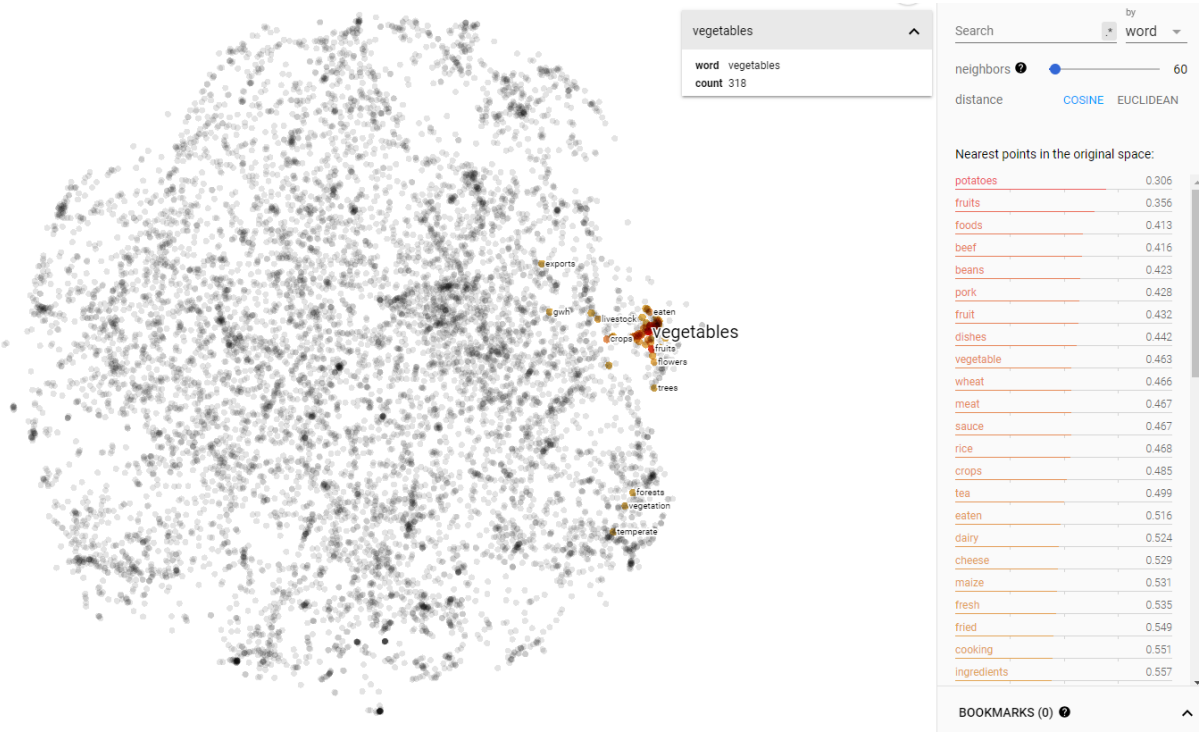
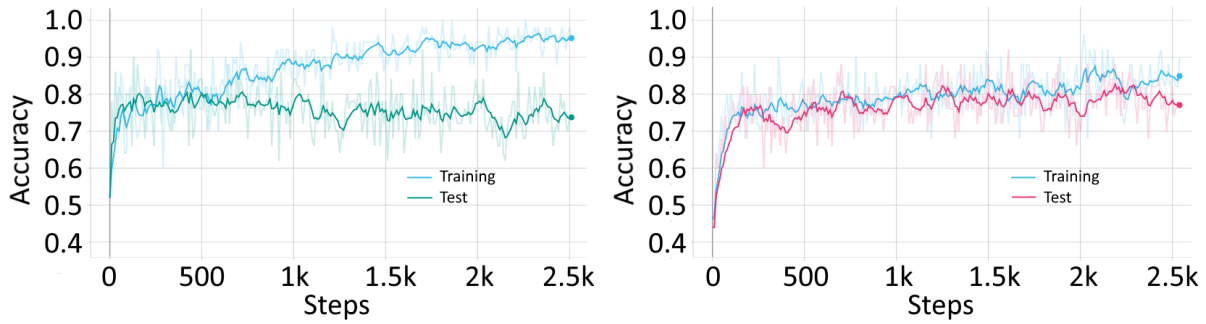


Figure 4: Word Vector Embedding

(i.e. words), we place a distribution over the word embeddings. Dropout in the word-based model then corresponds to randomly dropping word types in the sentence and might be interpreted as forcing the model not to rely on single words for classification.

With a dropout value of 30% as tested in the paper, Figure 5 shows the effect of adding variational dropout to the RNN.



(a) No Dropout

(b) Variational Dropout

Figure 5: RNN Dropout Comparison

There is a clear reduction in overfitting and a slight improvement in test performance, as predicted due to the reasoning described above.

2.1.2 Image Classification

A concept that has never been explored in the literature for sentiment analysis is the combination of both textual and visual features to make a prediction. This exploits an otherwise unused dimension of data, the images associated with each article. To exploit this, another architecture designed specifically for image classification will be implemented with the two classifiers then combined together to form an ensemble classifier that can yield a prediction of higher accuracy.

2.1.2.1 AlexNet CNN Architecture

A simplified version of the AlexNet CNN architecture developed by the University of Toronto[15] was adopted for this project for the following reasons:

1. Its simple construction allows for intuitive and mathematical insights into layer behaviour.
2. It consists of numerous layers that can model the hierarchical nature of image features. (Deeper than it is wide).
3. It yields good performance on ImageNet and Cifar10 classification challenges[15] relative to its size.

The simplified AlexNet architecture consists of a pattern of (convolution, pooling, normalisation) repeated three times, followed by a fully connected layer which passes into a softmax function to produce the output. This is depicted below in Figure 6 where ReLU signifies the passing through of a rectified linear unit and LRN represents a Local Response Normalisation Layer.

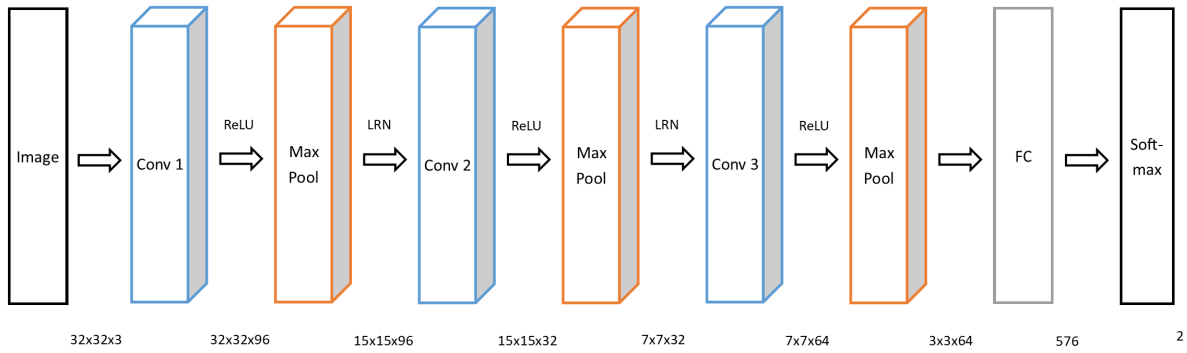


Figure 6: Simplified AlexNet architecture

Looking at each stage in more depth, it is an important point that the input image is first standardised before passing it through the network. This involves subtracting the

mean and dividing by the standard deviation of an image; this step ensures that each image updates the weights by the same amount for the same misclassification error. This is due to the backpropogation terms involving a multiplication of the reversed image pixel values. The derivation for this can be seen in Section A.2 of the Appendix.

Convolutional layers - Motivated by the translation invariance of images that they model and the need to reduce the number of model parameters to improve generalisation.

$$a_{x,y}^{(l)} = \sum_{u=-m}^{m} \sum_{v=-m}^{m} w_{u,v}^{(l)} Z_{x-u,y-v}^{(l)} \tag{6}$$

ReLU - (Pointwise) Non-linearities are added to make the neural network a non-linear system, this is necessary for the CNN to approximate any arbitrary function f. ReLUs were chosen over other activation functions such as tanh since they have shown to yield faster convergence rates[15].

$$y_{x,y}^{(l)} = f(a_{x,y}^{(l)}) = \begin{cases} a_{x,y}^{(l)} & \text{if } a_{x,y}^{(l)} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

LRN - Local Response Normalisation models lateral inhibition that is observed in neurobiology, where an excited neuron can subdue its neighbours. This is especially important when using ReLUs since they have an unbounded output, thus we want to dampen uniform outputs and increase sensitivity to relatively excited (high value output) neurons. Note k and α are hyper-parameters and i/j identify the channel index.[15]

$$y_{x,y,i}^{(l)} = a_{x,y,i}^{(l)} / \left(k + \alpha \sum_{j=\max\{0,i-\frac{n}{2}\}}^{\min\{N-1,i+\frac{n}{2}\}} (a_{x,y,j}^{(l)})^2 \right)^{\beta} \tag{8}$$

Max Pooling - Motivated by the fact that high-level image features are coarser than low-level image features. Through subsampling it also reduces the number of parameters to be learned, and also adds in further translation invariance.

$$t_{x,y}^{(l)} = \max_{u \in \{-m,\ldots,m\}, v \in \{-m,\ldots,m\}} y_{x-u,y-v}^{(l)} \tag{9}$$

FC - The Fully Connected layer is necessary to perform linear projection of the feature vector (output from final maxpool) to a K dimensional datapoint where K is the number of classes. Note when $K = 2$, the feature vector can equivalently be projected to a single

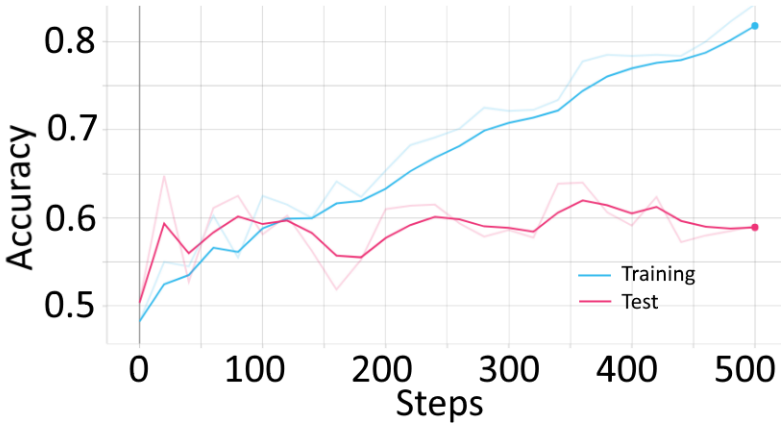
scalar value.

$$\mathbf{y}^{(l+1)} = \mathbf{V}^{(l)^T}\mathbf{t}^{(l)} \tag{10}$$

Softmax - The final layer scales the output(s) to the range $[0, 1]$ and normalises to form a valid probability mass function, albeit a pseudo-probability since the network can produce a biased estimate. Classification is then performed by selecting the class corresponding to the largest output.

$$x_j^{(L)} = \sigma(\mathbf{y}^{(L)})_j = \frac{e^{y_j^{(L)}}}{\sum_{k=1}^{K} e^{y_k^{(L)}}} \tag{11}$$

This network has 145103 model parameters, and training it on this project's relatively small data dataset yields the training curves shown below in Figure 7. This unsurprisingly doesn't generalise to the unseen test data well, and so strategies to deal with overfitting will need to be employed.



(a) Accuracy

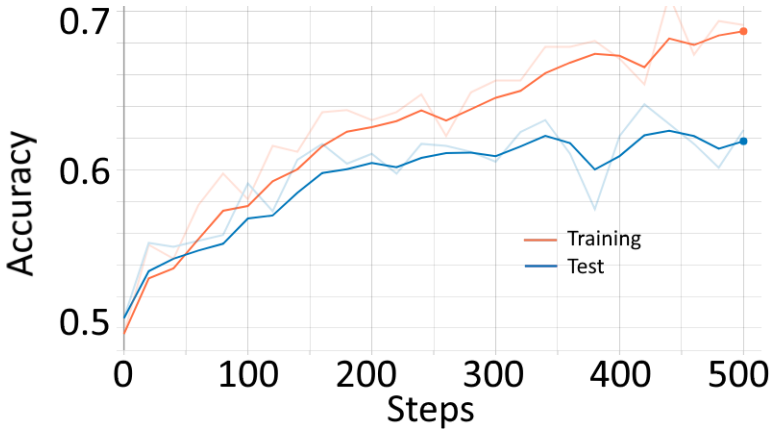
(b) Four Conv 1 Filters

Figure 7: AlexNet with no Overfitting Strategies

2.1.2.2 Transfer Learning

The best way to vastly reduce the number of model parameters is through transfer learning. There are two main approaches, both require first using a larger dataset to train the network and then using those trained weights as the initialisation for the desired classification task. The first method fixes the weights of all but the final layer, reducing the classification task to one of just identifying which high level image features correspond to which class. The second method allows for all the network weights to be changed from their initialisation, noting that in both cases the final fully connected layer is usually of a different dimension and randomly initialised.

Since low level image features (edges, blobs, corners) are common across all classification tasks, the first approach to transfer learning was adopted in the project because it is the only method that truly reduces the number of model parameters and so will help improve generalisation to a greater extent. The results of this are shown below in Figure 8, with the cifar-10 dataset being used for initial pre-training.



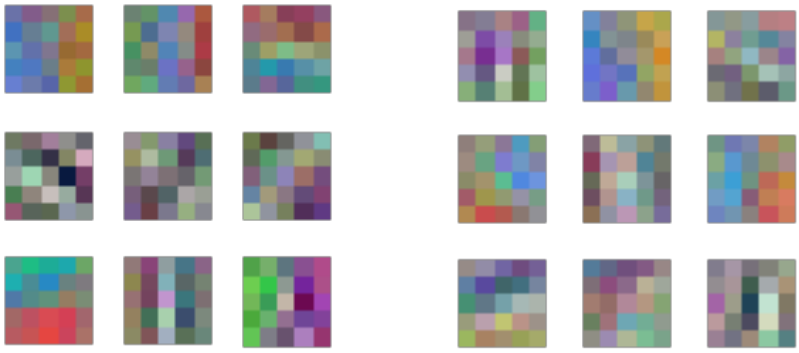
(a) Accuracy

(b) Four Conv 1 Filters

Figure 8: AlexNet with Transfer Learning

2.1.2.3 Dropout

Dropout, as described for text classification in Section 2.1.1.3, is another way to improve generalisation where random nodes are turned off (output set to zero) during each training step. This helps to reduce the overspecialisation of neurons, and encourage the discovery of multiple identifying features for each class. Following the method outlined in G. Hinton's paper[10], 50% of the nodes were dropped after every fully connected layer. Figure 9 shows a sample of nine filters from the first convolutional layer with and without dropout used during training. Just from this sample, it can be observed that there is an increase in the diversity of the filters learned, supporting the dropout heuristic. The performance curves also demonstrate this through a reduction of the gap between the training and test curves.



(a) 50% Dropout (b) No Dropout

Figure 9: AlexNet with Transfer Learning nine Conv 1 Filters

2.1.2.4 Deepening the Network

An advantage about using the transfer learning approach is that the size of the pre-trained network used for feature extraction does change the number of model parameters. Thus, a much deeper CNN consisting of 48 layers called InceptionV3[16] was used. Pre-training on the much larger imagenet dataset, before then learning the final layer that maps to sentiment 'show'/'hide' classifications, led to a significant performance increase shown below in Figure 10. This architecture was then taken forward and used by the ensemble classifier.

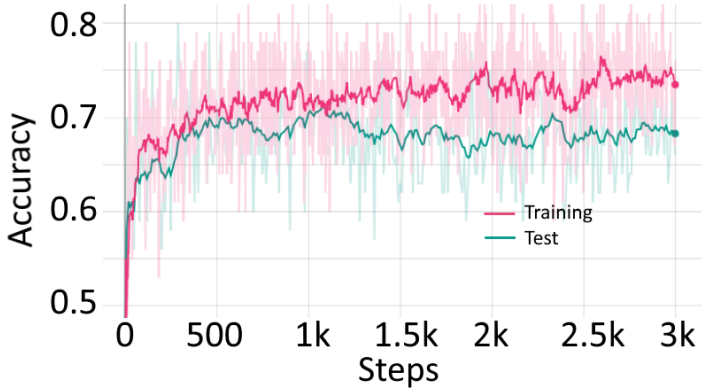


Figure 10: InceptionV3 with Transfer Learning

2.1.3 Ensemble Classifier

The ensemble classifier is the architecture that ties together the RNN, CNN, and additional features describing a datapoint to make an improved prediction.

The beauty about using an ensemble approach to make a prediction is that for appropriately trained weights, the performance can only be as low as the best weak learner (an

input classifier). This is the worst case scenario and occurs when either all of the inputs to the network are completely dependent or, non-dependent inputs simply give random values uncorrelated to the output.

Ideally, all inputs to the ensemble classifier should be independent; if this were the case then for a majority vote ensemble the overall accuracy is given by a poisson binomial distribution shown in (12).

$$\begin{aligned} accuracy &= P(z > \left\lceil \frac{N}{2} \right\rceil) \\ &= 1 - \sum_{l=0}^{\left\lceil \frac{N}{2} \right\rceil} \sum_{A \in F_l} \prod_{i \in A} p_i \prod_{j \in A^c} (1 - p_j) \end{aligned} \tag{12}$$

$N -$ number of weak learners

$p_i -$ probability weak learner i correct

However, since the ensemble has only two classifiers at its input, majority voting cannot be used. Instead, a weight based prediction will be implemented which has the added benefits of allowing additional features describing a datapoint and hidden layers to be included. Thus, an integer representing the length of an article's text was added as another input due to the information content it could contain with regards to the importance of an article's text with respect to its image.

The results of training the ensemble classifier with these inputs and a single hidden layer are shown below in Figure 11.

Figure 11: Accuracy of Ensemble Classifier

This achieves an accuracy of 85%, which is an improvement on the text and image classifier which had accuracies of 72% and 81% respectively. This shows there exists some independence between text and image features in the probabilistic sense and that combining them together has a positive effect on sentiment classification of news articles.

2.1.4 Methodology Conclusion

This section of the report aimed to develop a methodology of identifying negative click bait. It investigated novel ways of solving this problem including the bringing together of both image and text features through an ensemble classifier, proving that there is independence between these features and combining them yields an increase in performance.

It showed that adding dropout regulation increases the diversity of the learned filters which in turn improved the model generalisation for both RNNs and CNNs.

The section highlighted the importance of transfer learning for small datasets, with both the RNN and CNN displaying sub-par accuracies of less then 60% without its implementation. For the RNN, transfer learning was utilised in the embedding and for the CNN, it was used in a deep neural network to extract high level image features.

These strategies, as well others discussed in the section, led to an overall classification accuracy of 85%.

2.2 The Dataset

With all supervised learning methodologies, their performance is heavily determined by the quality of the dataset used for training. Usually it is this, and not the complexity of the model or algorithm that limit the accuracy and so careful consideration needs to be given with regards to what datapoints will be gathered, how it will gathered, and how it can be augmented.

Due to the labour intensive nature of constructing sentiment datasets, there exists very few of high quality and none in the news domain which are publicly available. Pseudo sentiment datasets, generated automatically from reviews have been constructed from amazon reviews[4][5] and movie reviews[1]. These make the assumption that a review of a high rating will contain a phrase positive in sentiment, with the converse being true for low ratings. A sentiment dataset also exists for twitter[6], where tweets are classified based on if they are positive or negative about a company of interest.

Since it is not clear if any of these datasets are related to the sentiment of news articles, it is necessary to construct a new bespoke dataset. A mapping can be constructed by training a network on this newly gathered data, then predicting the sentiment of datapoints from these existing datasets. If there is a strong correlation then the sentiment of news articles dataset can be expanded by including appropriate datapoints from the existing datasets; for example 5 star amazon reviews in the 'show' class. This is discussed in more depth in 2.2.3 - Extension through Existing Datasets.

2.2.1 Datapoint Gathering

The methodology for acquiring datapoints will be to construct a website that allows for many users to easily classify articles on desktop and mobile. This has advantages over a paper method or group sessions due to the convenience and increased robustness to individual biases. A snapshot of the created tool can be seen below in Figure 12.

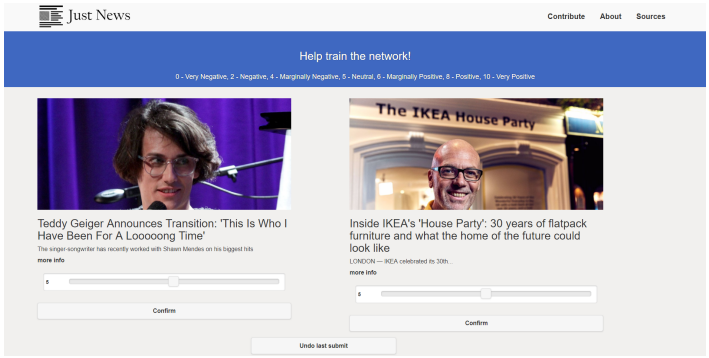


Figure 12: Classification Page on Website (justnews.io/supporters)

Data is gathered on an 11 class basis stretching from 'Very Negative' to 'Very Positive'. Although this is not needed for the binary 'show'/'hide' classification, it provides greater flexibility to handle incorrect assumptions made about user perceptions. This enables performance increases by providing greater insight into where the trained model is most erroneous.

Methods to enlist support in the classification of articles included: emails, Facebook posts, adding article share functionality, and native website notifications. This resulted in a modest dataset size of 1636 articles. To ensure an even class size ratio, a pre-classification technique was employed half way through the dataset construction stage. This involved training the RNN on the small number of datapoints already classified, with the website then showing unseen articles predicted in the less frequent class with a greater probability.

A T-Distributed Stochastic Neighbour Embedding (t-SNE) visualisation of the data can be seen below in Figure 13, where each colour represents a different class.

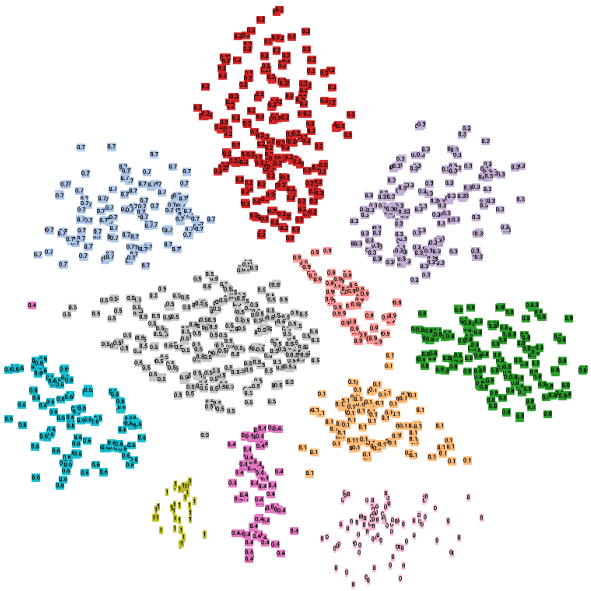


Figure 13: 2D t-SNE with Supervision

2.2.2 Data Augmentation

2.2.2.1 Similar Articles

Augmenting text is hard due to the high interdependence of words within a sentence. This project devised a novel method of augmenting the number of datapoints by finding similar articles on the internet and giving them the same class. The methodology is analogous

to an automated process of googling an article and taking the top n matches as valid augmentations.

The results of doing this were poor, with a reduction in performance of approximately 10%. Instead, it was decided to use this algorithm to improve the speed of the data gathering process by suggesting similar articles on the classification page. The user then quickly identifies which articles (if any) have the same sentiment. A visualisation of this tool can been seen below in Figure 14.

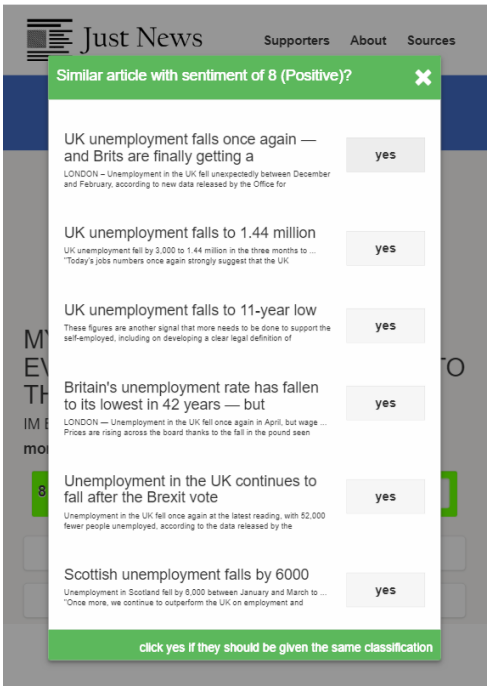


Figure 14: Website Similar Articles Popup

2.2.2.2 Thesaurus Word Swapping

Another novel method this project devised to augment text data is to swap certain words, for example nouns and adjectives with other highly relevant words of the same type. This works because a thesaurus is based on semantics, so careful substitution of words should not change the overall sentiment of an article. However, thought should be given to how these changes affect the vectors of embedded words described in Section 2.1.1.2, since it will only have effect if semantically similar words are not close to one another in the embedding. The embedding used in the project is a contextual one, for example the closest word to 'good' is 'bad'. Thus, thesaurus word swapping should be a viable method of augmenting the dataset.

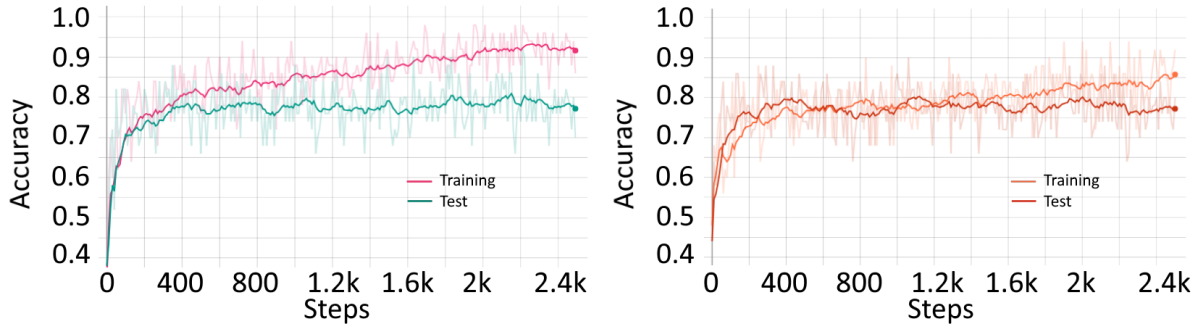
An example output of the algorithm is shown below in Figure 15.

Word Type	['noun'	'verb'	'verb'	'verb'	'adj'	'as in'	'unknown'	'noun'	'prep'	'as in'	'unknown'	'adj'	'unknown'	'noun'	'noun'	'adj'	'noun'	'verb'	'as in'	'noun'	'adv'	'adj'	'noun'	'adj'	'as in']
Original Scentence	[['Science'	['may']	['have'	['cured'	['biased']	['AI']	-	['Researchers'	['at']	['Columbia']	['Lehigh']	['developed'	['IT_NPfIT']	['software'	['tool'	['capable'	['error'	['checking'	['neural'	['networks'	['potentially'	['curing']	['bias'	['in']	['AI']]
Possible Augmentations	'discipline' 'information' 'art' 'technique' 'system' 'learning' 'skill' 'education']		'enjoy' 'include' 'get' 'receive' 'acquire' 'bear' 'accept' 'keep' 'admit' 'own' 'take' 'carry' 'retain' 'possess' 'hold' 'obtain' 'gain']	'restore' 'repair' 'relieve' 'treat' 'correct' 'improve']				'scientist' 'analyst' 'investigator']				'advanced']		'program']	'engine' 'means' 'weapon' 'mechanism' 'machine' 'device']	'experienced' 'suited' 'talented' 'efficient' 'gifted' 'able' 'accomplished' 'good' 'qualified' 'adequate' 'intelligent']	'fault' 'failure' 'offense' 'sin']	'monitor' 'confirm' 'review' 'correct' 'try' 'investigate' 'test' 'analyze' 'study']	'visual']	'web' 'chain' 'system' 'organization' 'net' 'structure']	'possibly' 'likely' 'probably']		'preference' 'leaning' 'tendency']		

Figure 15: Example Augmented Sentence

During mini-batch generation in the network training phase, a set of sentences are first randomly drawn from the dataset. Each sentence is then augmented by randomly choosing words from the possible augmentations, such as the one shown in Figure 15.

After many experiments it was found that nouns, verbs and adjectives are the only word types that give an increase in performance when augmented. Augmenting just these word types significantly reducing overfitting, which is shown by the accuracy plots in Figure 16.



(a) No Augmentation (b) Nouns, Verbs, & Adjectives Augmentation

Figure 16: RNN Thesaurus Augmentation Accuracy Comparison

2.2.2.3 Images

Images, unlike text, have the desirable property of being easily augmented. Thus, all of the standard augmentation practices were implemented including: rotations, flips, crops, changes in brightness and changes in contrast.

2.2.3 Extension through Existing Datasets

As mentioned in the preface to this section, another way to increase the number of datapoints is by calculating mappings from existing datasets. A mapping can be constructed by training a network on the this project's bespokely built dataset, then predicting the sentiment of datapoints from the existing datasets. If there is a strong correlation then this project's dataset can be expanded by including appropriate datapoints from the existing datasets; for example 5 star amazon reviews in the 'show' class.

Mappings were calculated for both the pseudo-sentiment datasets of Amazon and movie reviews, while the twitter dataset was discarded upon further inspection due to a strong disparity in sentiment definition, leading to clearly contradicting results (tweets were classified based on their sentiment towards a particular company, not about the underlying matter involved). A mapping for a further dataset, acquired from a local Cambridge

startup, which classified the sentiment of political articles was also calculated. The results of these mappings for textual data is shown below in Figure 17.

Stars Rating	Predicted: hide	Predicted: show
0	0.63	0.37
1	0.63	0.37
2	0.66	0.34
3	0.69	0.31
4		
5		
6		
7	0.62	0.38
8	0.60	0.40
9	0.55	0.45
10	0.47	0.53

(a) Movie Reviews

Star Rating	Predicted: hide	Predicted: show
1	0.16	0.84
2	0.14	0.86
3	0.12	0.88
4	0.10	0.90
5	0.08	0.92

(b) Amazon Reviews

Given	Predicted: hide	Predicted: show
vneg	0.92	0.08
neg	0.84	0.16
neu	0.62	0.38
pos	0.44	0.56
vpos	0.21	0.79

(c) Politically Classified

Figure 17: Existing Dataset Mappings

From these results it can be seen that their only exists a mapping for the political dataset. Thus, the amazon and movie review datapoints were discarded and all datapoints from the political dataset's 'vneg' and 'vpos' classes were added to the 'hide' and 'show' classes respectively. This increased the number of text datapoints from 1636 to 2761 before any other augmentations were applied.

2.2.4 Test Set Construction

An important part of constructing a dataset is reserving some datapoints to use for model testing. This section aims to quantify how many datapoints should be reserved for a given confidence interval on the test result. It is especially important in this project due to the modest dataset size used, forcing the test set to work close to the boundary where statistical fluctuations may have an effect.

For supervised learning it is important to maintain even class sizes to avoid bias in the predictions and misleading test results. This was done dynamically when generating random mini-batches which avoids having to discard datapoints from the class with a larger number.

Let X_j be a binary random variable representing a correct or incorrect prediction of the j^{th} datapoint in the test set, where $X_j = 1$ and $X_j = 0$ represent a correct and incorrect prediction respectively. Assuming the underlying probability of being correct is p, this

can be compactly written as a Bernoulli random variable given in (13).

$$X \sim Ber(p) \tag{13}$$

Thus, the observed accuracy of the test set can be now represented as another random variable, Y, given below in (14).

$$\begin{aligned} Y =& \frac{1}{N}\sum_{j=0}^{1} X_j \\ & N = \text{Number of Datapoints in Test Set} \\ NY \sim& Binomial(p, N) \end{aligned} \tag{14}$$

A Matlab script, documented in Figure 29 Section A.3 of the Appendix, was written to calculate the 95% confidence interval for a given value of N and p. Using 20% of all the datapoints for testing and assuming a likely value of $p = 0.75$, gives a confidence interval of $\pm 4.2\%$.

Note this requires guessing a p value, or assuming the most conservative estimate that $p = 0.5$. A better estimate can be obtained via a method devised by Cornell University[7], which in this case yields a similar answer to assuming $p = 0.75$.

2.3 Further Results and Discussion

To demonstrate the effectiveness of this project, a live experiment was devised to test how accurate the filtering algorithm is with respect to user perception. The algorithm was run on live news articles from 12 different sources over the course of a month. 100 articles from this set were then randomly chosen and classified based on their sentiment, with very negative sentiment values indicating the 'hide' class and all others the 'show' class.

The confusion matrix in Table 2 shows the results, with the overall accuracy at 84%.

Actual \ Predicted	Hide	Show
Hide	0.786	0.214
Show	0.128	0.872

Table 2: Live Experiment Confusion Matrix

Further insight can be gained by looking into the false positives and false negatives, where a positive result is defined as classifying an article in the 'show' class.

Two random examples of false positives (an article classified as hide but it's shown by the network) are given below.

"James Stannard assault: Briton pleads not guilty in Sydney - The UK man will contest charges that he punched Australia's rugby sevens captain James Stannard"

"21 Of The Most Powerful Photos Of This Week - Here are the most incredible and breathtaking pictures from the past week"

For the first statement, it could be argued that this in fact a neutral article since it is just informing about the state of a defendant with regards to a trail. But to remain consistent with our definition of sentiment as described in Section 1.2, since this is about a very negative matter, assault, it should be hidden by the network.

On first sight the second statement, when reading only the text, looks positive and one the network should show. However, it is the image associated with the article (and all subsequent ones contained within in it, although these are not seen by the network) that contain very negative, graphic scenes which is why it should be hidden. The image associated with this article can be seen below in Figure 18.

Figure 18: Example False Positive

To gain more insight into exact network behaviour during this classification, outputs from both the text classifier and image classifier were analysed independently and are shown below, where an output of 1 indicates complete certainty in showing the article and 0 complete certainty in hiding it.

$$\begin{aligned} text &: 0.995 \\ image &: 0.339 \end{aligned} \tag{15}$$

This clearly shows correct behaviour, with the image classifier indicating a very negative image and the text classifier indicating near certainty in it being something we want to see. This case highlights the issue of combining multiple classifiers together via their direct network output, when no constraints are placed on their relative values during training. Thus for the three stage architecture adopted in this project, it is an understandable misclassification for the algorithm to make.

Looking now at two examples of false negatives (an article classified as 'show' but it is hidden by the network).

> *"Belgium May Have New Appeal for Millennials: Join the Army and Sleep at Home - A proposal to allow recruits to the country's armed forces to avoid staying in barracks during basic training drew scorn from veterans and military experts."*

> *"No 10 refuses Caribbean request to discuss children of Windrush - British citizens threatened with deportation to countries they left as children and have not returned since "I'm here legally, but they're asking me to prove I'm British" Downing Street has rejected a formal diplomatic request to discuss the immigration ..."*

The first article, although it is about a negative topic *"training drew scorn from..."*, it is not very negative and therefore still one we want to see. This borderline case fooled the text classifier but interestingly, the image was accurately predicted as positive even though it contained a uniformed man holding a gun, see Figure 19. To further test the robustness of the image classifier, the positive looking civilians were edited out as shown

in Figure 20.

Figure 19: Example False Negative

Figure 20: False Negative Edited

This resulted in the network output dropping from 0.643 to 0.602, indicating a slight reduction in the certainty of showing the article but still greater than the threshold of 0.5. This shows a high level of robustness in the network through the redundancy of features, with the image classifier still being able to detect that this type of picture of a soldier is likely not to be about an article of very negative sentiment.

The second example of a false negative is a similar story of correct image prediction but incorrect text prediction. This time, the certainty of the image classifier is higher at 0.867 but because the ensemble classifier gives a lower weighting to image predictions, borderline text cases can sometimes cause an overall prediction error as in this case.

All of the false positive and false negative prediction errors are justifiable and highlight the difficultly, even for humans, for distinguishing between ambiguous cases. From scanning all the prediction errors from the 100 articles in this test there were no obvious mistakes, showcasing the good performance this algorithm achieves in solving the problem.

Finally, analysis of the correct predictions was undertaken to gain insight into difficult predictions the algorithm got right.

Two examples of true positives are shown below.

"Balkan dam projects could result in loss of one in 10 European fish species - Exclusive: Plans for a network of hydropower plants in three countries would cause 'chain reaction' for endangered species. Report warns nearly one in 10 of Europe's fish species will be pushed to the brink of extinction by a constellation of hydropower plant"

Trump Blasts Comey in Barrage of Tweets, Calling Him 'Slippery' - President Trump called James Comey, the former F.B.I. director, a slime ball for the second time this week. Ist was one of numerous disparaging comments Mr. Trump made about Mr. Comey on Twitter on Sunday morning.

Both of these articles have a clear negative aspect to them but are not negative enough to warrant filtering out.

The first article about endangered fish is particularly difficult because if you take the last sentence alone *"Report warns nearly one in 10 of Europe's fish species* ***will be*** *pushed to the brink of* ***extinction*** *by a constellation of hydropower plant"*, that can be considered of very negative sentiment. Thus, the text classifier must ensure the earlier features are not lost since they provide the less negative context *"Balkan dam projects* ***could result*** *in* ***loss*** *of one in 10 European fish species"*. Passing only the aforementioned last sentence through the text classifier results in a certainty of showing the article of 0.447, inline with the human expectations above. The fact this is close 0.5 suggests some uncertainty in the network prediction which is understandable since the last sentence is arguably borderline on being very negative. Passing the entire article through the text classifier results in a certainty of showing the article of 0.981. This indicates the text classifier is able to retain key information from earlier on and is therefore robust to contradicting statements within an article.

Looking finally at an example of a true negative:

"First monsoon rain exposes risks for Rohingya refugees in Bangladesh - Projected rainfall at Cox's Bazar could put refugees in danger of contracting diseases and infections. The first much-feared rain fell on Bangladeshi border town of Cox's Bazar on Wednesday, bringing with it apprehension and the first signs of flooding in the..."

This example discuses natural disasters and disease and so can be considered to be of the very negative hide class, something the network predicts accurately with a pseudo certainty of showing the article of 0.000254. To test the sensitivity of this estimate, specific key words were replaced with their positive antonyms. The changes are shown below in Table 3 along with the network outputs, noting that these specific changes do not affect the overall sentiment of the article.

Words swapped (from ⇒ to)	Network Output
refugees ⇒ people	0.000254
disease/infection ⇒ happiness/affection	0.000242
risks/danger/feared ⇒ opportunities/opportunity/loved	0.000253
rain/rainfall/flooding ⇒ sun/sun/sun	0.000348
monsoon ⇒ light	0.000351
Bangladesh ⇒ England	0.000263

Table 3: Robustness to Word Antonym Swapping

These results clearly show the text classifier is not sensitive to individual word changes and has learnt a complex model with redundancy to enable this strong robustness. However, combining all of the above word antonym swaps yields a very different article, given below.

> *"First light sun exposes opportunities for Rohingya people in England - Projected rainfall at Cox's Bazar could put people in opportunity of developing happiness and affection. The first much-loved sun fell on Bangladeshi border town of Cox's Bazar on Wednesday, bringing with it delight and the first signs of sun in the..."*

This article has a far from 'very negative' sentiment and is clearly one the network should show. Running the text classifier on this yields an output of 0.731, showcasing once again how the deep dependencies enable the network to achieve good performance on this challenging problem.

2.4 Summary of Results

To summarise, this project investigated and implemented a number of key strategies that aimed to improve the overall classification accuracy. Most notably, it devised the novel approach of combining both the text and image features of sentiment, leading to a performance increase of 4% compared to just using a text classifier. Appropriate dropout was added with the results showing an increased diversity of learned filters for the CNN and increased robustness to individual word swaps for the RNN, both resulting in a reduction of overfitting shown by Figures 9 and 5 respectively. Transfer learning was added with the aim of reducing the number of model parameters to less than the number of datapoints, enabling better generalisation by restricting the amount each model can overfit to the training data. This dramatically increased performance by over 10% in each case. As supervised learning is heavily reliant on the dataset, in addition to the standard image augmentations three further novel strategies were implemented. These were, using a thesaurus to augment only nouns, verbs and adjectives, finding similar news articles on the internet and extending the number of datapoints by calculating mappings from

existing datasets in different domains. As was discussed in the relevant sections, thesaurus word swapping significantly reduced overfitting, similar articles successfully increased the speed at which datapoints were gathered and dataset mapping increased the number of datapoints by 69%.

All of the above led to overall accuracy of 85% with the further results and discussion section showcasing that this is a good result for what can be a tricky problem even for humans.

3 Just News Platform

3.1 Implementation

The website, justnews.io, is the platform that displays live news articles post filtering to help reduce the exposure people have to negative click bait. At its core, the website simply displays articles that the algorithm deems not of the negative, click bait type. Since the final stage of the ensemble classifier is a sigmoid function, where outputs above 0.5 signify a 'show' prediction for the network, it is possible to adjust the severity of the filtering simply by changing the minimum network output required for an article to be shown on the website. A threshold of zero will perform no filtering and all articles will be seen, whereas a high threshold will result in the website only showing articles for which it has a high certainty of not being from the negative, click bait class. A slider adjustment that varies this threshold from $0 - 0.9$ can be found on the website by clicking the down arrow shown in Figure 21.

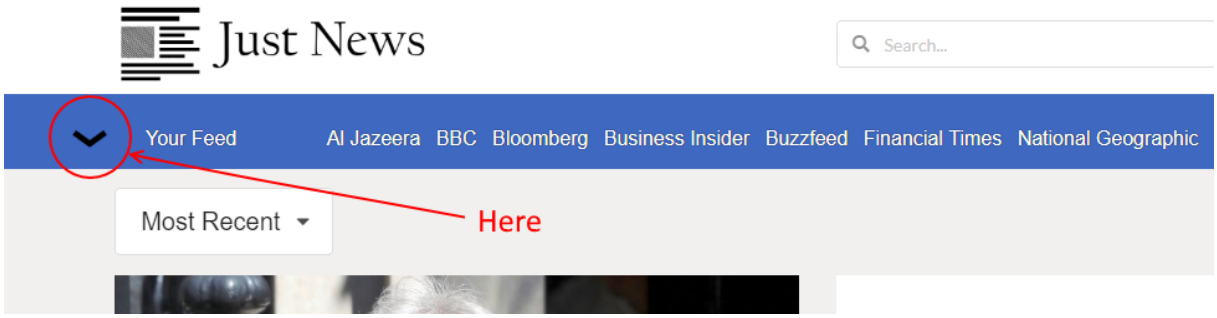


Figure 21: Website Button to Show Threshold Slider

3.2 Features

The website contains numerous features to help fulfil the aim of reducing peoples exposure to negative click bait.

The first functionality is to filter articles via news providers using the tabs. This is essential for any news aggregation site due to the large variance in user preference, but especially important for this project because it helps the user to identify how much negative click bait each provider is publishing. Thus, more informed choices can be made by the user when selecting news sources in the future which will further help to reduce exposure.

The second key functionality is notifications, which can be subscribed to by clicking the icon in the bottom right hand corner of the display, shown in Figure 22.

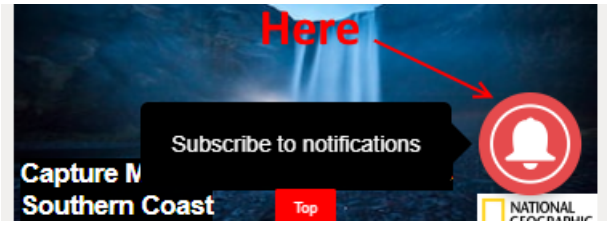


Figure 22: Website Subscribe to Notifications Button

The notifications are used to send articles of interest and provide users with important updates about negative click bait. The purpose of this is to ensure Just News is not forgotten about after the first couple of uses, since reducing exposure to negative news requires long term adoption of the technology.

An 'About' section on the website has been implemented, which can be navigated to by clicking the button labelled 'About' in the top right hand corner of the website. The page displays a short abstract on the purpose of this project, along with what the website does. This not only aims to inform users about the issue and how they can solve it, but also aims to persuade them to become advocates for the issue such that their influential circle will also use the website and reduce their exposure.

For similar reasoning, a share button has been added shown below in Figure 23. This enables an article to be shared via a private message or social media, with the link taking the recipient to the Just News website version of that article.

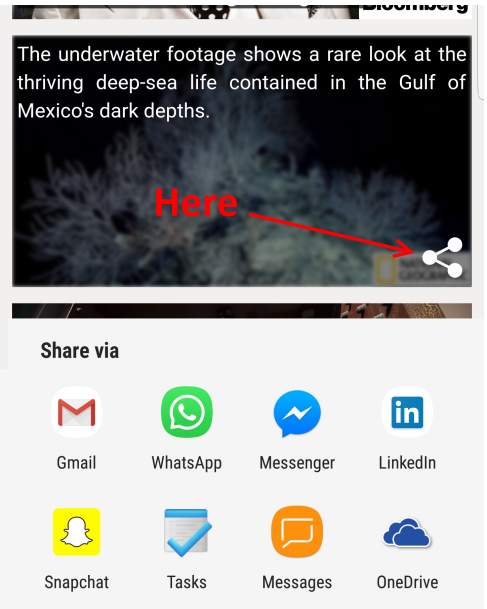


Figure 23: Website Share Functionality

3.3 Usability Tests

A series of usability tests were undertaken with 8 different students, where they were asked to use Just News as their main source of news for a week. Feedback was gathered during and after the process which was used to guide the design of the website. It lead to a number of key features being added; most notably the search bar, trending filter, personalised feed and topic filtering.

The search bar (located at the top of the screen on desktop), as the name suggests, allows for custom terms to be entered resulting in relevant articles being returned. This functionality allows for a very good live demonstration of the negative click bait filtering algorithm. Searching terms such as 'death' or 'murder', while changing the filtering threshold highlights the good performance this algorithm achieves and the positive affects of using it.

Prior to user testing, articles were simply ordered by the most recent ones at the top. The purpose of this project is to help reduce people's exposure to negative click bait which was stated in Section 1.1 to have the benefit of uncovering more relevant articles. However, that benefit does not exist if there are many unpopular/trivial stories above the interesting ones. Thus, an option to view trending articles was added to correct this problem, shown in Figure 24.

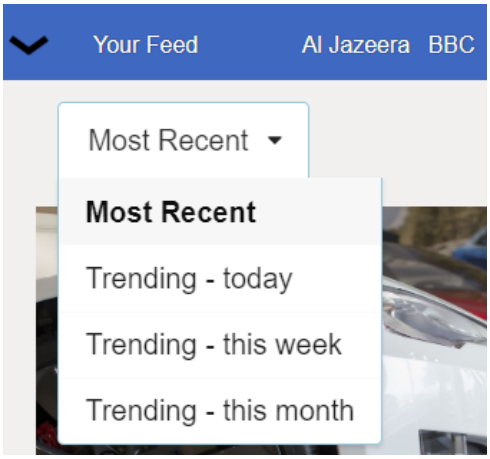


Figure 24: Website Trending Filter

A popular request was the functionality of a personalised feed, where you can set your preferences on news providers and topics with the website remembering them for subsequent visits. The was implemented through the addition of the 'Your Feed' tab shown by Figure 25, and helps users to select a custom range of news that minimises their exposure to negative click bait.

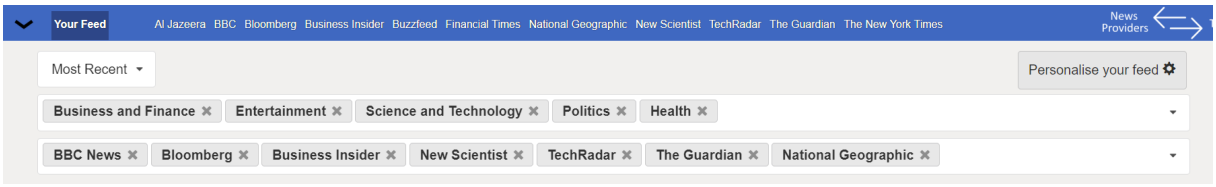


Figure 25: Website Personalised Feed

3.4 Topic Filtering

As mentioned in the usability tests in the previous section of this report, an important feature for this project to fulfil the latter part of its aim of providing a platform that reduces peoples exposure to negative news, is an algorithm that can identify the topic of a news article. Topic classification, unlike the classification of negative click bait, is a much easier problem to solve and there are many readily available datasets that can be used.

The reason topic filtering is a much easier problem to solve is because a topic can typically be inferred from a high density of specific words or phrases. Thus, a deep learning approach is not required and more classical methods can be used.

3.4.1 Vector Representation of Words

The first stage is to convert an article into a vector representation. This project uses the term frequency inverse document frequency method (TFIDF) due to its ability to extract n-gram features as devised by K. Spärck Jones from Cambridge University[17].

TFIDF is the product of two statistics, term frequency and inverse document frequency. There are various ways for determining the exact values of both statistics. In the case of the term frequency, $tf(t, d)$, the simplest choice is to use the raw count of a term in a document, i.e. the number of times that term t occurs in document d. If we denote the raw count by $f_{t,d}$, then the tf scheme is:

$$tf(t, d) = f_{t,d} \tag{16}$$

The inverse document frequency is a measure of how much information the word provides, i.e. whether the term is common or rare across all documents. It is the logarithmically scaled inverse fraction of the documents that contain the word, obtained by dividing the total number of documents by the number of documents containing the term, then taking

the logarithm of that quotient.

$$
\begin{aligned}
idf(t, D) &= \log \frac{N}{|\{d \in D : t \in d\}|} \\
D &- \text{set of all documents} \\
N &= |D|
\end{aligned}
\tag{17}
$$

TFIDF is then calculated as:

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D) \tag{18}$$

A high weight in TFIDF is reached by a high term frequency (in the given document) and a low document frequency of the term in the whole collection of documents; hence the weights tend to filter out common terms. Since the ratio inside the idf's log function is always greater than or equal to 1, the value of idf (and TFIDF) is greater than or equal to 0. As a term appears in more documents, the ratio inside the logarithm approaches 1, bringing the idf and TFIDF closer to 0.

For this project, a term is defined as any uni-gram or bi-gram contained within at least five documents. Thus, uni-grams or bi-grams that occur in very few documents will not be used as features due to the tenancy for an algorithm to overfit to the training data if the specifics of each document can be learnt.

The vector representation of an article is therefore the concatenation of scalar TFIDF features for all valid terms. For a quick sanity check, summing all TFIDF feature vectors for every article within a topic enables us to view the most common terms. This is shown below in Figure 26, which upon first inspection shows promise for finding a viable topic classifier.

```
# 'business':
  . Most correlated unigrams:
      . economy
      . bank
      . oil
  . Most correlated bigrams:
      . analysts said
      . oil prices
      . economic growth
```

```
# 'politics':
  . Most correlated unigrams:
      . blair
      . election
      . labour
  . Most correlated bigrams:
      . tony blair
      . prime minister
      . mr blair
```

```
# 'science and technology':
  . Most correlated unigrams:
      . microsoft
      . apple
      . google
  . Most correlated bigrams:
      . wi fi
      . galaxy s5
      . samsung galaxy
```

```
# 'entertainment':
  . Most correlated unigrams:
      . kardashian
      . singer
      . film
  . Most correlated bigrams:
      . game thrones
      . kim kardashian
      . box office
```

```
# 'health':
  . Most correlated unigrams:
      . fda
      . mers
      . ebola
  . Most correlated bigrams:
      . study finds
      . medical marijuana
      . breast cancer
```

```
# 'sport':
  . Most correlated unigrams:
      . cup
      . coach
      . match
  . Most correlated bigrams:
      . australian open
      . year old
      . grand slam
```

Figure 26: Largest TFIDF features within each topic

3.4.2 Classifier

Three different approaches for the classification of news topics using article TFIDF features were trained, tested and compared. These were a Random Forest Classifier, Logistic Regression, and Multinomial Naive Bayes' Classifier.

3.4.2.1 Random Forest

Random forest classifiers work by constructing many different decision trees and averaging their outputs to get a better prediction. A decision tree is defined by its nodes which govern binary splits of the data stemming from a root node. Nodes can split either nominal or naturally ordered data, with topic classification containing only the naturally ordered data of TFIDF features. Ideally for a datapoint $\mathbf{x}$, the probability of class label ω_i could be calculated as in (19).

$$\begin{aligned} P(\omega_i|\mathbf{x},\mathcal{D}) &= \sum_{\mathcal{T}\in\boldsymbol{T}} P(\omega_i|\mathbf{x},\mathcal{T})P(\mathcal{T}|\mathcal{D}) \\ \boldsymbol{T} &- \text{set of all decision trees} \\ \mathcal{D} &- \text{training data} \end{aligned} \tag{19}$$

However, since it is not feasible to compute all possible trees, a monte carlo approximation will be used as shown in (20).

$$\begin{aligned} P(\omega_i|\mathbf{x},\mathcal{D}) &\approx \frac{1}{N}\sum_{j=1}^{N} P(\omega_i|\mathbf{x},\mathcal{T}^{(j)}) \\ &where\ \mathcal{T}^{(j)} \sim P(\mathcal{T}|\mathcal{D}) \end{aligned} \tag{20}$$

Sampling from $P(\mathcal{T}|\mathcal{D})$ is approximated by the following algorithm used during tree construction. For each node $\mathcal{N}$:

1. sort the questions by purity gain (impurity loss)
2. randomly select a question from the top Q questions
3. apply the question to node $\mathcal{N}$

The impurity of a node can be defined in many different ways as long as it is a symmetric function for all classes, maximum when $P(\omega_i) = \frac{1}{K}$, and minimum when $P(\omega_i) = 1$ and $P(\omega_j) = 0\ \forall j \neq i$. This project uses the entropy measure which is defined below in

(21).

$$\mathcal{I}(\mathcal{N}) = -\sum_{i=1}^{K} P(\omega_i|\mathcal{N}) log(P(\omega_i|\mathcal{N})) \tag{21}$$

Thus, the impurity loss for a node $\mathcal{N}$ splitting into its left and right descendants, $\mathcal{N}_L$ and $\mathcal{N}_R$, is defined as:

$$\begin{aligned} \Delta\mathcal{I}(\mathcal{N}) = \mathcal{I}(\mathcal{N}) - \frac{n_L}{n_L + n_R}\mathcal{I}(\mathcal{N}_\mathcal{L}) - \frac{n_R}{n_L + n_R}\mathcal{I}(\mathcal{N}_\mathcal{R}) \\ n_L - \text{number of samples in the left descendant} \\ n_R - \text{number of samples in the right descendant} \end{aligned} \tag{22}$$

Another important consideration when constructing random forests is the stopping criteria. This is needed to improve generalisation by stopping pure nodes being reached. This project used an easy to implement, but effective method of doing this by limiting the maximum tree depth to three. Other parameters this project used were $N = 200$ trees in the forest and Q, the number of questions to sample over, was limited to the square root of the number of TFIDF features, which by empirical observation often leads to good classification results[18].

3.4.2.2 Logistic Regression

Logistic regression is a classification algorithm that models the probability describing a possible binary outcome using a logistic function. Logistic regression can handle multiclass problems by using the one vs rest technique[19], where a single binary classifier is trained per class, with all datapoints not contained in that class grouped under a separate label (rest). The prediction of the entire model is the class corresponding to the largest output across all binary classifiers. However, do note that there are no constraints placed between the binary classifier outputs, so there is no mathematical certainty that picking the largest output will indeed give the most likely class; this method is just a heuristic that has good empirical results.

To train each binary classifier, the cost function given in (23) is minimised where $y^{(i)}$ represents the binary class label $\in \{-1, 1\}$ for the i^{th} datapoint $\mathbf{x}^{(i)}$.

$$\min_{\mathbf{w},c} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N} \log(\exp(-y^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} + c)) + 1) \tag{23}$$

This project minimises the cost function using an extended version of the conjugate gra-

dient decent method as described and implemented by LIBLINEAR[20].

3.4.2.3 Multinomial Naive Bayes' Classifier

A Naive Bayes' Classifier is one which applies Bayes' theorem with the 'naive' assumption of independence between every pair of features shown by (24), where $\mathbf{x}$ and y represent a feature vector and class variable respectively.

$$P(x_i|y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_d) = P(x_i|y) \tag{24}$$

Thus, the decoupling of the class posterior feature distributions means that each of them can be independently estimated as a one dimensional distribution shown by (25). This helps to alleviate problems from large dimensionality[21].

$$\begin{aligned} P(y \mid x_1, \ldots, x_d) &\propto P(y) \prod_{i=1}^{d} P(x_i \mid y) \\ \hat{y} &= \arg\max_y P(y) \prod_{i=1}^{d} P(x_i \mid y), \end{aligned} \tag{25}$$

The Multinomial Naive Bayes' Classifier is the variant of this for multinomially distributed data. The input data is typically the word vector counts, although TFIDF vectors are also known to work well in practice[22]. The distribution is parametrized by vectors $\theta_y = (\theta_{y1}, \ldots, \theta_{yd})$ for each class y, where d is the number of features and θ_{yi} is the probability $P(x_i \mid y)$ of feature i appearing in a sample belonging to class y.

The parameters θ_y are estimated by a smoothed version of maximum likelihood, i.e. relative frequency counting of TFIDF terms:

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n} \tag{26}$$

where $N_{yi} = \sum_{\mathbf{x} \in \mathcal{D}_y} x_i$ is the sum of TFIDF feature i over all samples in the training set belonging to class y, $(\mathcal{D}_y)$. $N_y = \sum_{i=1}^{|\mathcal{D}_y|} N_{yi}$ is the total sum of all TFIDF features for class y.

The smoothing priors, $\alpha \geq 0$, account for features not present in the learning samples and prevent zero probabilities in further computations.

3.4.2.4 Comparison

All three classification algorithms were trained on the six aforementioned topics and evaluated using 10-Fold cross validation shown below in Figure 27. K-Fold cross validation is where the dataset is split into K equally sizes partitions, trained on $K-1$ partitions and evaluated on the other. Thus, testing results in K different accuracy values.

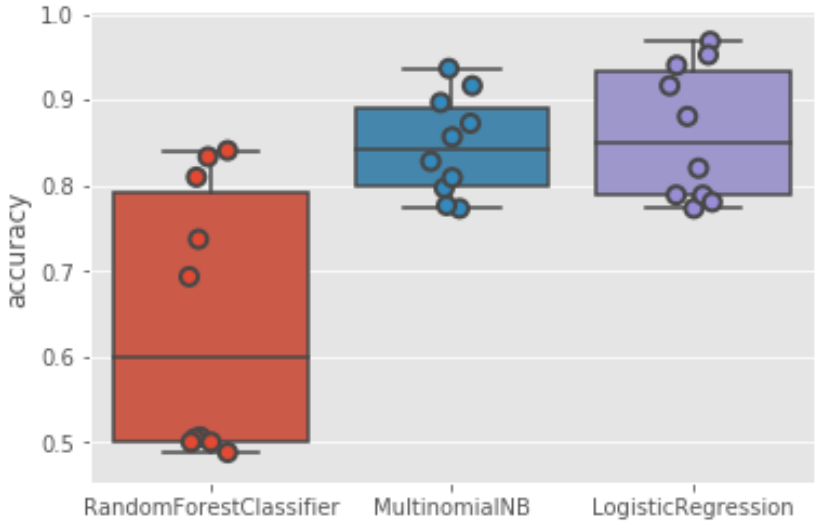


Figure 27: Classifier Comparison

The results show logistic regression, on average, performs slightly better than multinomial naive Bayes' but with higher variance. However, both methods significantly outperform the random forest classifier. Visualising the topics for a couple of TFIDF feature dimensions show that they are not piece-wise linearly separable. This is an assumption only made by the random forest, explaining its poor performance.

Logistic regression makes a crude assumption of being able to compare binary classifier outputs using one vs rest, where as multinomial naive Bayes' assumes the likelihood of each feature dimension are independent. Figure 27 shows these two assumptions have similar affects, therefore due to the slight increase in performance on average, logistic regression was chosen to classify the TFIDF features.

3.4.3 Demonstration

Filtering by topic can be performed through either 'Your Feed', or in its own write via clicking on the switch button identified in Figure 28.

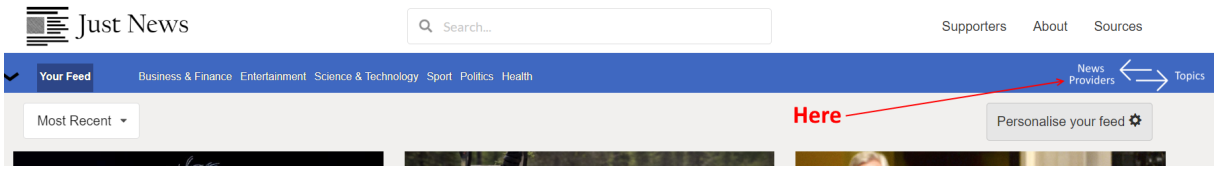


Figure 28: Website Topic Filtering

4 Conclusion

To conclude, this project aimed to solve the growing problem of negative click bait in the news. This consisted of a two part strategy; firstly, building a classifier capable of detecting said articles and secondly, developing a platform to help reduce people's exposure.

To address the first part, the project investigated novel ways of solving this problem including bringing together both image and text features through an ensemble classifier, proving that there is independence between these features and combining them yields an increase in performance. It developed a method for augmenting textual data, where for this problem augmenting nouns, verbs and adjectives with highly relevant words of the same type improves performance. The investigation into challenging datapoints that caused erroneous predictions demonstrated appropriate behaviour for both individual text and image classifiers, but highlighted the difficulty in combining just two classifiers as the strength of their outputs are difficult to compare when no constraints are placed on them during training. An overall accuracy of 85% was achieved for this algorithm, a good result for a challenging task with many debatable borderline cases.

The second part of the problem was addressed through a website named Just News. The platform successfully enabled the efficient construction of a high quality dataset consisting of 1636 datapoints. The homepage, shows a constant stream of live news articles from 12 popular providers with realtime filtering using the developed algorithm. It boasts a range of features to help reduce peoples exposure to negative click bait including: topic filtering, search bar, trending items, news provider filtering, share buttons and notifications. A series of user tests were undertaken to identify these features and therefore demonstrate a certain level of effectiveness in addressing a solution to the problem.

Taken together, this project fulfils its aim of developing a comprehensive solution to negative click bait in the news, reducing peoples exposure by 85%.

5 Future Work

Various aspects of this project can be investigated and developed further. Most notably for the ensemble classifier where more advance strategies have been developed recently such as AdaBoost and XGBoost. To improve performance when there is contracting information from both image and text features, training a combined system with a more complex loss function should enable a more valid comparison; this system however would be more difficult to train due to the increase in model parameters. Further investigations into the word embedding can be made, with the current pre-trained embedding containing examples such as 'good' being closest to 'bad', which seems suboptimal for sentiment classification.

References

[1] Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C.D., Ng, A. and Potts, C., 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In Proceedings of the 2013 conference on empirical methods in natural language processing (pp. 1631-1642).

[2] Go, A., Bhayani, R. and Huang, L., 2009. Twitter sentiment classification using distant supervision. CS224N Project Report, Stanford, 1(12).

[3] Harris, C., 2018. Searching for Diverse Perspectives in News Articles: Using an LSTM Network to Classify Sentiment.

[4] R. He, J. McAuley. Modeling the visual evolution of fashion trends with one-class collaborative filtering. WWW, 2016

[5] J. McAuley, C. Targett, J. Shi, A. van den Hengel. Image-based recommendations on styles and substitutes. SIGIR, 2015

[6] Kaggle.com. (2018). Twitter sentiment analysis — Kaggle. [online] Available at: https://www.kaggle.com/c/twitter-sentiment-analysis2.

[7] Chen, X., 2007. Exact computation of minimum sample size for estimation of binomial parameters. arXiv preprint arXiv:0707.2113.

[8] Gers, F., 2001. Long short-term memory in recurrent neural networks. Unpublished PhD dissertation, Ecole Polytechnique Fdrale de Lausanne, Lausanne, Switzerland.

[9] Colah.github.io. (2018). Understanding LSTM Networks – colah's blog. [online] Available at: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[10] Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R.R., 2012. Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580.

[11] Mikolov, T., Chen, K., Corrado, G. and Dean, J., 2013. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

[12] McCormick, C. (2016). Google's trained Word2Vec model in Python. [online] Mccormickml.com. Available at: http://mccormickml.com/2016/04/12/googles-pretrained-word2vec-model-in-python/

[13] Pennington, J., Socher, R. and Manning, C., 2014. Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

[14] Gal, Y. and Ghahramani, Z., 2016. A theoretically grounded application of dropout in recurrent neural networks. In Advances in neural information processing systems (pp. 1019-1027).

[15] Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

[16] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z., 2016. Rethinking the inception architecture for computer vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 2818-2826).

[17] Sparck Jones, K., 1972. A statistical interpretation of term specificity and its application in retrieval. Journal of documentation, 28(1), pp.11-21.

[18] Scikit-learn.org. (2018). 1.11. Ensemble methods scikit-learn 0.19.1 documentation. [online] Available at: http://scikit-learn.org/stable/modules/ensemble.html#id6

[19] Nasrabadi, N.M., 2007. Pattern recognition and machine learning. Journal of electronic imaging, 16(4), p.049901.

[20] Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R. and Lin, C.J., 2008. LIBLINEAR: A library for large linear classification. Journal of machine learning research, 9(Aug), pp.1871-1874.

[21] Friedman, J.H., 1997. On bias, variance, 0/1loss, and the curse-of-dimensionality. Data mining and knowledge discovery, 1(1), pp.55-77.

[22] Scikit-learn.org. (2018). 1.9. Naive Bayes' scikit-learn 0.19.1 documentation. [online] Available at: http://scikit-learn.org/stable/modules/naive_bayes.html

A Appendix

A.1 RNN Backpropogation

The chosen objective function is the binary entropy function, where $t^{(n)} \in \{0, 1\}$ and $y^{(n)}$ are the class label and network output for n^{th} datapoint $\mathbf{x}^{(n)}$ respectively.

$$\mathcal{L} = -\sum_{n=1}^{N} t^{(n)} ln y^{(n)} + (1 - t^{(n)}) ln(1 - y^{(n)}) \tag{27}$$

Assuming an input sequence of length T, $\{\mathbf{x}_t\}_{t=1}^{T}$, the RNN can be unrolled to form a feedforward network shown in Figure 2. Thus, the network output y can be represented as shown below in (28), where $g(\mathbf{x}_t, \tilde{\mathbf{h}}_{t-1})$ is the function that describes the LSTM unit given in (2). This calculation is called the forward pass, storing the computed values of $\{\mathbf{h}_t\}_{t=1}^{T}$ and $\{\mathbf{c}_t\}_{t=1}^{T}$ for later computations. $\mathbf{h}_0$ and $\mathbf{c}_0$ are usually initialised, and are for this project, to zero.

$$\begin{aligned} y &= \sigma(\mathbf{v}^T \mathbf{h}_T + b) \\ &= \sigma([\mathbf{v}^T, \mathbf{0}^T]\tilde{\mathbf{h}}_T + b) \\ &= \sigma(\tilde{\mathbf{v}}^T g(\mathbf{x}_T, g(\mathbf{x}_{T-1}, ..., g(\mathbf{x}_1, \tilde{\mathbf{h}}_0)...)) + b) \end{aligned} \tag{28}$$

Derivatives of the loss function w.r.t network weights can now be calculated, this is called the backwards pass. For a general weight matrix W

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial W_{i,j}} &= \sum_{n=1}^{N} \frac{\partial \mathcal{L}}{\partial y^{(n)}} \frac{\partial y^{(n)}}{\partial z^{(n)}} \frac{\partial z^{(n)}}{\partial W_{i,j}} \\ &= \sum_{n=1}^{N} \frac{y^{(n)} - t^{(n)}}{y^{(n)}(1 - y^{(n)})} \cdot y^{(n)}(1 - y^{(n)}) \cdot \frac{\partial z^{(n)}}{\partial W_{i,j}} \\ &= \sum_{n=1}^{N} (y^{(n)} - t^{(n)}) \frac{\partial z^{(n)}}{\partial W_{i,j}} \end{aligned} \tag{29}$$

For the fully connected layer's weight vector ($\mathbf{v}$) and bias (b) derivatives have been calculated in (30) and (31) respectively.

$$\frac{\partial \mathcal{L}}{\partial \mathbf{v}} = \sum_{n=1}^{N} (y^{(n)} - t^{(n)}) \mathbf{h}_T \tag{30}$$

$$\frac{\partial \mathcal{L}}{\partial b} = \sum_{n=1}^{N} (y^{(n)} - t^{(n)}) \tag{31}$$

For the parameters governing the LSTM cell behaviour, (32) shows how their derivative can be calculated.

$$\frac{\partial \mathcal{L}}{\partial W_{i,j}} = \sum_{n=1}^{N} (y^{(n)} - t^{(n)}) \mathbf{v}^T \frac{\partial \mathbf{h}_T}{\partial W_{i,j}} \tag{32}$$

There are 11 unique weight and 4 bias vector derivatives for the equations governing the LSTM cell defined in (3) and (4). The derivatives for one of these weight matrices and one bias vector derivative are calculated below in expressions (33) and (34) respectively, with the others following a similar form.

$$\frac{\partial \mathbf{h}_T}{\partial (W_f^f)_{i,j}} = \boldsymbol{i}_o \odot sech^2(\mathbf{c}_T) \odot \frac{\partial \mathbf{c}_T}{\partial (W_f^f)_{i,j}}$$

$$\frac{\partial \mathbf{c}_t}{\partial (W_f^f)_{i,j}} = \boldsymbol{i}_f \odot \frac{\partial \mathbf{c}_{t-1}}{\partial (W_f^f)_{i,j}} + \begin{bmatrix} 0 \\ \vdots \\ \sigma'(a_{f_i}(t)) x_{t_j} c_{t-1_i} \\ \vdots \\ 0 \end{bmatrix}$$

$$(\frac{\partial \mathbf{h}_T}{\partial (W_f^f)_{i,j}})_k = \begin{cases} i_{o_i} sech^2(c_{T_i})(i_{f_i}^T c_{0_i} + \sum_{l=1}^{T} i_{f_i}^{T-l} \sigma'(a_{f_i}(l)) x_{l_j} c_{l-1_i}) & \text{if } k = i \\ 0 & \text{otherwise} \end{cases}$$

$$\frac{\partial \mathcal{L}}{\partial (W_f^f) i, j} = \sum_{n=1}^{N} (y^{(n)} - t^{(n)}) v_i i_{o_i} sech^2(c_{T_i})(i_{f_i}^T c_{0_i} + \sum_{l=1}^{T} i_{f_i}^{T-l} \sigma'(a_{f_i}(l)) x_{l_j} c_{l-1_i}) \tag{33}$$

$$
\begin{aligned}
\frac{\partial \mathbf{h}_T}{\partial b_{f_i}} &= \boldsymbol{i}_o \odot sech^2(\mathbf{c}_T) \odot \frac{\partial \mathbf{c}_T}{\partial b_{f_i}} \\
\frac{\partial \mathbf{c}_t}{\partial b_{f_i}} &= \boldsymbol{i}_f \odot \frac{\partial \mathbf{c}_{t-1}}{\partial b_{f_i}} + \begin{bmatrix} 0 \\ \vdots \\ \sigma'(a_{f_i}(t))c_{t-1_i} \\ \vdots \\ 0 \end{bmatrix} \\
(\frac{\partial \mathbf{h}_T}{\partial b_{f_i}})_k &= \begin{cases} i_{o_i} sech^2(c_{T_i})(i_{f_i}^T c_{0_i} + \sum_{l=1}^{T} i_{f_i}^{T-l} \sigma'(a_{f_i}(l))c_{l-1_i}) & \text{if } k = i \\ 0 & \text{otherwise} \end{cases} \\
\frac{\partial \mathcal{L}}{\partial b_{f_i}} &= \sum_{n=1}^{N} (y^{(n)} - t^{(n)}) v_i i_{o_i} sech^2(c_{T_i})(i_{f_i}^T c_{0_i} + \sum_{l=1}^{T} i_{f_i}^{T-l} \sigma'(a_{f_i}(l))c_{l-1_i})
\end{aligned}
\tag{34}
$$

where $a_{f_i}(t)$ is i^{th} row of $W_f^f \mathbf{x}_t + W_f^r \mathbf{h}_{t-1} + W_f^m \mathbf{c}_{t-1} + \mathbf{b}_f$, and σ' is the derivative of the sigmoid function. The expanded column vectors are zero in all expect the i^th dimension, and for a zero initialisation of $\mathbf{h}_0$ and $\mathbf{c}_0$ the $i_{f_i}^T c_{0_i}$ term disappears.

A.2 CNN Backpropogation

Assuming a neural network consisting of 4 stages; a convolutional layer, point-wise non-linearity, fully connected layer, and normalising logistic function. The equations governing these four stages are given by (6), (7), (10) and (35) respectively, noting that the logistic function is softmax equivalent for 1 dimension.

$$\begin{aligned} x^{(n)} &= \sigma(u^{(n)}) \\ &= \frac{1}{1 + e^{-\sum_{i,j} V_{i,j} y_{i,j}^{(n)}}} \end{aligned} \tag{35}$$

The chosen objective function is the binary entropy function, where $t^{(n)} \in \{0, 1\}$ and $x^{(n)}$ are the class label and network output for n^{th} datapoint $\mathbf{z}^{(n)}$ respectively. This is combined with regularisations terms on the weights (equivalent to a Gaussian prior), that softens the decision boundary which helps to reduce overfitting.

$$\mathcal{L} = -\sum_{n=1}^{N} t^{(n)} ln x^{(n)} + (1 - t^{(n)}) ln(1 - x^{(n)}) + \frac{\alpha}{2} \sum_{i,j} V_{i,j}^2 + \frac{\beta}{2} \sum_{i,j} W_{i,j}^2 \tag{36}$$

Derivatives of the loss function w.r.t network weights are then taken.

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial W_{a,b}} &= \sum_{n=1}^{N} \frac{\partial \mathcal{L}}{\partial x^{(n)}} \frac{\partial x^{(n)}}{\partial u^{(n)}} \sum_{i,j} \frac{\partial u_{i,j}^{(n)}}{\partial y_{i,j}^{(n)}} \frac{\partial y_{i,j}^{(n)}}{\partial a_{i,j}^{(n)}} \frac{\partial a_{i,j}^{(n)}}{\partial W_{a,b}} + \beta W_{a,b} \\ \frac{\partial \mathcal{L}}{\partial x^{(n)}} &= \frac{x^{(n)} - t^{(n)}}{x^{(n)}(1 - x^{(n)})} \\ \frac{\partial x^{(n)}}{\partial u^{(n)}} &= \sigma(u^{(n)})(1 - \sigma(u^{(n)})) = x^{(n)}(1 - x^{(n)}) \\ \frac{\partial u_{i,j}^{(n)}}{\partial y_{i,j}^{(n)}} &= V_{i,j} \\ \frac{\partial y_{i,j}^{(n)}}{\partial a_{i,j}^{(n)}} &= f'(a_{i,j}^{(n)}) = \begin{cases} 1 & \text{if } a_{i,j}^{(n)} > 0 \\ 0 & \text{otherwise} \end{cases} \\ \frac{\partial a_{i,j}^{(n)}}{\partial W_{a,b}} &= Z_{i-a,j-b}^{(n)} \\ \frac{\partial \mathcal{L}}{\partial W_{a,b}} &= -\sum_{n=1}^{N} (t^{(n)} - x^{(n)}) \sum_{i,j} V_{i,j} f'(a_{i,j}^{(n)}) Z_{i-a,j-b}^{(n)} + \beta W_{a,b} \end{aligned} \tag{37}$$

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial V_{a,b}} &= \sum_{n=1}^{N} \frac{\partial \mathcal{L}}{\partial x^{(n)}} \frac{\partial x^{(n)}}{\partial u^{(n)}} \sum_{i,j} \frac{\partial u_{i,j}^{(n)}}{\partial V_{a,b}} + \alpha V_{a,b} \\ &= -\sum_{n=1}^{N} (t^{(n)} - x^{(n)}) V_{a,b} y_{a,b}^{(n)} + \beta V_{a,b}\end{aligned} \tag{38}$$

A.3 Code

```
N  = 220            % number of datapoints
p = 0.75            % binomial p parameter
x = 0:N;            % (note: ensure p*N is integer for the function below)

% plot binomial pdf
y = binopdf(x,N,p);
plot(x./N,y,'+')

% get confidence interval range
for i = 1:N
    prob = sum(y(p*N-i+1:p*N+i+1));
    inc = x(p*N-i+1:p*N+i+1)./N;
    ran = inc(end) - inc(1);
    if prob >= 0.94 % confidence interval percentage (approximately)
        break
    end
end

disp(prob)          % actual confidence interval percentag
disp(i)             % number of possible outcomes in this confidence interval
disp(ran)           % confidence interval range
```

Figure 29: Test Dataset Size Code

A.4 Risk Assessment

Before work on this project commenced, a risk assessment form that identified hazards and appropriate mitigation strategies was completed. The main risks were primarily related to injuries associated with computer use as no physical experiments were required for this project. This included injuries to the back and RSI however, none were sustained due to effective mitigation strategies.