An Analysis of Machine Learning and Deep Learning Techniques for Detecting Sarcasm in Text

Student Name: Molly Hayward

Supervisor Name: Dr Noura Al-Moubayed

Submitted as part of the degree of BSc Computer Science to the

Board of Examiners in the Department of Computer Sciences, Durham University

Abstract —

Context / Background – In recent years, sarcasm detection has gained interest due to the increased use of sentiment analysis and social media analytics. Sarcasm is a unique form of language that cannot be interpreted at surface-level, therefore detecting sarcasm proves a significant challenge for traditional sentiment analysers and humans alike. This highlights the scope for innovative machine learning and deep learning solutions to this ill-structured problem.

Aims – The aim of this project is to contrast the performance of machine learning and deep learning classifiers in order to develop an effective sarcasm detection solution. Furthermore, it aims to extend the superior binary classification solution in order to highlight tokens that correlate more to sarcastic and non-sarcastic labels. Lastly, this project aims to investigate to what extent manually-extracted sentiment features enhance machine learning classification, drawing a comparison to conventional lexical features.

Method – Two datasets containing Twitter data and news headlines are transformed into meaningful numerical vectors using several approaches, including a state-of-the-art contextualised embedding approach. These features are used to train machine learning and deep learning classifiers, and their precision, recall and F_1 scores are compared. An evaluation corpus of Amazon reviews is gathered to assess the cross-dataset performance of candidate solutions, and an attention mechanism is incorporated into the superior binary classification solution to localise the presence of sarcasm.

Results – The solution utilises a Bidirectional Long Short-Term Memory model trained on the Twitter dataset, as this achieves the highest cross-dataset performance. Integrating an attention layer into this model improves the previous F_1 score by 1.5% to 0.862. An F_1 score of 0.899 is achieved on the news headlines dataset using a Bidirectional Gated Recurrent Unit, exceeding the state-of-the-art on this dataset.

Conclusions – Deep learning models consistently outperform machine learning models in the sarcasm detection task. Furthermore, models trained on ElMo vectors were consistently superior to those trained on all other vector types. While sentiment features were informative, models trained on these features were unable to rival their deep learning opposition.

Keywords — Sarcasm Detection, Natural Language Processing, Binary Classification, Sentiment, Attention, Machine Learning, Deep Learning

INTRODUCTION

Sarcasm is defined as speech or writing which actually means the opposite of what it seems to say, usually intended to mock or insult[1]. It is a complex linguistic phenomenon that is prevalent in online user-generated content, and is typically used as a form of mockery or humour when expressing an opinion; hence, to interpret sarcastic text literally would be to overlook its implied (usually opposite) meaning. Sarcasm poses a significant challenge within the field of natural

language processing, particularly within sentiment analysis. This is the task of deducing the sentiment polarity of text - typically whether the author is in favour of, or against, a specific subject. When sarcasm is present in text, it can invert the sentiment polarity of a positive or negative utterance, as positive sounding phrases can have an implicit negative meaning. It is increasingly common for organisations to use sentiment analysis to gauge public opinion on their products and services; however, classic sentiment analysers cannot deduce the implicit meaning of sarcastic text and will wrongly classify the author's opinion. Advancements in automatic sarcasm detection research have the potential to vastly improve the task of sentiment analysis; hence, any tool that strives to accurately determine the meaning of user-generated text must be capable of detecting sarcasm.

A Problem Background

Identifying sarcasm in text is a unique and interesting challenge. Although sarcasm is a wholly human construct, even humans struggle to consistently recognise it; due in part to the lack of an all-encompassing, universal definition. In a 2011 study, González-Ibánez et al. [8] found low agreement rates between human annotators when classifying statements as either sarcastic or non-sarcastic, and in their second study, three annotators unanimously agreed on a label less than 72% of the time. Furthermore, developmental differences such as autism, as well as cultural and societal nuances cause variations in the way different people define and perceive sarcasm. Consequently, there is no set formula for producing (and therefore identifying) sarcastic text.

Prior work in the field has primarily focused on overt instances of sarcasm; capturing constrasting positive and negative sentiment phrases [23], such as *"isn't it great to feel tired?"*, or *"working again tomorrow, yay!"*. However, identifying implicit forms of sarcasm may require knowledge of additional context; as these contextual cues can be more powerful sarcastic indicators than the words themselves. Consider the scenario where a person is congratulated on their hard work, despite the obviousness of them having not worked hard at all. Or perhaps a customer thanking a waiter for the delicious food, despite sending it back to the kitchen. In isolation, speech may not enough to convey the sarcastic intent; therefore in transcribing this interaction, we lose necessary context and we lose the sarcastic undertones. In the absence of context, we look to more sophisticated deep learning techniques in order to learn the underlying patterns in implicit forms of sarcasm.

B Research Questions and Objectives

The **research questions** guiding this project are as follows –

- *Can advances in machine learning and deep learning address the challenges in sarcasm detection?*
- *Can a custom model identify which linguistic cues correlate more to sarcastic labels?*
- *Can handcrafted sentiment features improve sarcasm predictions?*

The following objectives are designed to address these specific research questions, and are divided into three categories depending on their priority level and difficulty.

The **minimum** objectives of this project are to evaluate and clean high-quality datasets, and to employ machine learning models in order to evaluate their performance on our chosen datasets.

[1] https://www.collinsdictionary.com/dictionary/english/sarcasm

To fulfil these objectives, we gather two large balanced datasets containing Twitter data and news headlines; as well a small unbalanced corpus of product reviews. Upon implementing five machine learning classifiers, we conclude that logistic regression is our superior machine learning model on both datasets, achieving F_1 scores of 0.851 and 0.790. The purpose of this objective is to establish a baseline against which to compare our deep learning classifiers.

The **intermediate** objectives of this project are to experiment with and evaluate deep learning models on our chosen datasets, as well as to determine the best performing solution by evaluating the model against unseen data. We achieve these objectives by implementing shallow and deep Convolutional Neural Networks; as well as advanced Recurrent Neural Networks and their bidirectional counterparts. We evaluate our candidate solutions on an evaluation corpus of Amazon reviews, where we discover that our bidirectional LSTM model, trained on ElMo vectors from the Twitter corpus, is our most adaptable solution - achieving an F_1 score of 0.611.

The **advanced** objectives of this project are to implement an attention-based neural network in order to identify words that correlate more to classification decisions. A visualisation of the attention weights associated with specific tokens would be produced in testing. We fulfil these objectives by integrating a custom attention layer into our superior binary classification solution, observing an improvement in the F_1 score of our solution. We visualise attention weights (returned by our attention layer) by projecting them into a colourspace and highlighting the corresponding tokens in their respective colours.

RELATED WORK

In this section, we report on the architectural design of solutions used specifically in sarcasm detection research, as well as techniques that have seen success in other natural language processing (NLP) classification tasks. Oftentimes, sarcasm detection is treated as a binary classification problem, in which text is grouped into two categories - sarcastic and non-sarcastic. It is otherwise treated as a multi-class problem where the extent to which a statement is sarcastic is ranked on a discrete scale, e.g. 1 to 5. In similar studies, the terms irony and sarcasm are used synonymously [26], although their definitions differ slightly (*table 1*). Both definitions capture the humourous intent in both sarcasm and irony, however sarcasm includes the potential for mockery and criticism - often present in opinion-based content such as product reviews.

Table 1: Definitions of Irony and Sarcasm

Term	Definition
Sarcasm	Speech or writing which actually means the opposite of what it seems to say, usually intended to mock or insult.[1]
Irony	A subtle form of humour which involves saying things that you do not mean.[2]

A Rule-based Classifiers

A number of naïve approaches have been used in previous research, where text is classified based on a set of linguistic rules. Bharti et al. (2015) [2] proposed two rule-based approaches - one approach (IWS) classifies tweets that begin with injerjections, such as *"wow"* and *"yay"*, as sarcastic, and the other uses a parsing based lexical generation algorithm (PBLGA) to recognise sentiment-bearing situation phrases juxtaposed with contrasting sentiment phrases. When tested

[2]https://www.collinsdictionary.com/dictionary/english/irony

on 1500 tweets containing #sarcasm, the IWS and PBLGA approaches yielded F_1 scores of 0.90 and 0.84, respectively. An earlier example of a rule-based approach is displayed in Maynard and Greenwood (2014) [15]. They proposed that the sentiment contained in hashtags can indicate the presence of sarcasm in tweets, such as #notreally in the example *"I love doing the washing-up #notreally"*. They used a rule-based approach where tweets are labelled as sarcastic if their sentiment contradicts the sentiment of the hashtag, reporting an F_1 score of 0.91.

Rule-based classifiers can perform well if their rule-sets are tailored to specific datasets - however, they cannot generalise effectively to data that is structurally different, hence they are considered primitive when compared to their modern, deep learning counterparts.

B Machine Learning Classifiers

Machine learning algorithms do not rely upon a static linguistic rule-set and can instead learn the underlying patterns associated with each class, allowing them to accurately classify unseen data. Previous research predominantly focuses on the use of tree-based classifiers, support-vector machines (SVM) and logistic regression models. Reyes et al. (2013) [22] used a naïve bayes and decision trees algorithm in order to detect irony which is closely related to sarcasm. They experimented with balanced and unbalanced distributions (25% ironic tweets and 75% other), achieving an F-score of 0.72 on the balanced distribution, dropping to 0.53 for the inbalanced distribution. In a similar vein, Barbieri et al. (2014) [1] used random forest and decision tree classifiers, also for the detection of irony. They used six types of features in order to represent tweets, and recorded results over three categories of training data - education, humor and politics.

As one of the first studies to consider sentiment features, Riloff et al. (2013) [23] describes sarcasm as a contrast between positive sentiment (e.g. *"i love"*), in reference to a negative situation (e.g. *"being ignored"*). They use a bootstrapping algorithm to extract positive verb phrases and negative situation phrases from a Twitter corpus and used these to train a hybrid model consisting of a SVM and a contrast method which classifies text as sarcastic if a postive sentiment phrase precedes a negative situation. This yielded an F_1 score of 51%. González-Ibánez et al. (2011) [8] also considered unique miscellaneous features, including lexical (e.g. unigrams) and pragmatic (e.g. emoticons) features. They trained an SVM with sequential minimal optimisation and a logistic regression model, achieving 71% accuracy using an SVM trained on unigrams. Ptáček et al. (2014) [21] also used a support-vector machine, performing classification on balanced and inbalanced datasets.

C Deep Learning Classifiers

Deep neural networks are increasingly being used in text classification - models that leverage time-series data, such as Recurrent Neural Networks (RNN), are well-suited to this task, as the order of the input sequences bares significance to its meaning. *Zhang et al. (2016)* [29] used a bidirectional gated RNN to capture local information in tweets, as well as a pooling neural network to extract context from historic tweets, achieving 78.55% accuracy.

Convolutional neural networks, that are typically associated with image-related tasks, have more recently been employed in text classification. *Zhang et al. (2015)* [30] explored the use of character-level convolutional neural networks for text classification. This network consists of 6 convolutional layers and 3 fully-connected layers. They found that it showed better peformance on raw texts such as a corpus of Amazon product reviews. A similar technique was applied in *Ghosh et al. 2018)* [7] where they concluded that emoticons such as ':)' and interjections e.g. "ah", "hmm" correspond with highly weighted sentences. The first deep learning approach to

sarcasm detection was used in *Poria et al. (2017)* [20], producing state-of-the-art results. They created three CNN pre-trained on personality, emotion and sentiment features, combining these to form a deep CNN. Deep neural networks often require a lot more training data than machine learning algorithms, however, they have been shown to produce state-of-the-art results in several domains.

D Incorporation of Attention

One particularly interesting technique is the *Hierarchical Attention Network (HAN)* defined in Yang et al. (2016) [28] - an attention-based LSTM. The intuition is that certain words and sentences in a document contribute more to its overall meaning, and this is highly context dependent. They included one attention mechanism at word-level and another at sentence-level, and this allowed them to determine which words and phrases correlate more to certain labels. Using a similar approach in the domain of sarcasm may allow me to highlight the attention-words that are strongly influence the level of sarcasm.

SOLUTION

This project addresses a binary classification problem, aiming to determine if an unseen snippet of text is sarcastic or non-sarcastic. Our sarcasm detection tool employs trained classification models which form predictions based on knowledge gained from features encountered during training. Developing highly accurate classification models remains the greatest challenge in this project, therefore it is the central focus of our solution. Model peformance is strongly dependent on the data pre-processing and feature extraction techniques employed, as well as the choice of model architecture and training procedures.

In our solution, high quality datasets are first collected and pre-processed. During feature extraction, the textual training data is transformed into numerical vectors and these features are used to train a classification model. The trained classifier is then carefully evaluated in order to account for potential overfitting. The following figure outlines the model pipeline, and each of these stages are further addressed in the remaining sections.

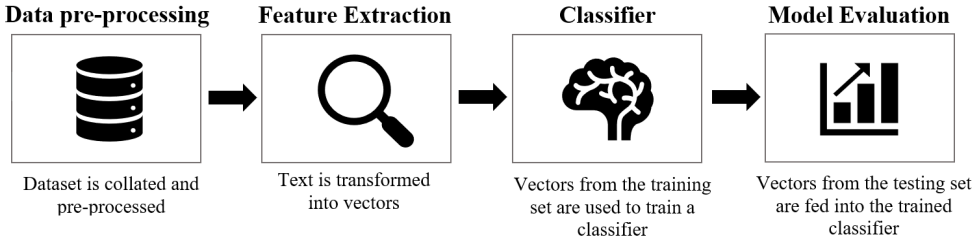


Figure 1: Overview of the solution

A Implementation Tools

This project is majoritively written in Python, as it has an abundance of open-source libraries well-suited to machine learning and deep learning tasks. Furthermore, Python is very readable, making it easy for others to test and reproduce results. We use pandas for manipulating our datasets, as it provides functionality for applying functions to all instances at the same time, and harness the built-in functionality from SpaCy[3] to clean and tokenise our data. In terms of *classification*, we implement most of the machine learning architectures using scikit-learn [27] -

an open-source machine learning library. For constructing deep learning models, we use keras as its modular structure allows for extensive experimentation and tweaking of model architectures.

B Datasets

We utilise three publically available datasets and provide a statistical analysis of their properties in *table 2*. This includes the number of instances in the dataset, the average number of tokens per instance and a benchmark score where the dataset was used in the original paper to train one or more classifiers. This data is collected from multiple domains and online platforms, including Twitter, Amazon, and online news outlets; providing broad coverage of sarcasm usage in both formal and informal media. We use our News Headlines and Twitter datasets for training, and reserve our Amazon review corpus for cross-dataset evaluation.

1. **News Headlines** - *Misra et al. (2019)* [17] collected thousands of news headlines from two sources - *The Onion* [4] (renowned for posting satiricial news) and *The Huffington Post* [5]. They utilised a hybrid neural network architecture and achieved a classification *accuracy* of 0.897.
2. **Twitter Data** - Collated in *Ptáček et al. (2014)* [21], we use the balanced dataset of 100000 English tweets. Of this original corpus, 17.28% of tweets remain available to access on Twitter, therefore all others are omitted from the compiled dataset. Sarcastic instances are tweets that include *#sarcasm*, and non-sarcastic instances are selected arbitrarily from the set of all other tweets. Ptáček et al. (2014) [21] achieved an F_1 score of 0.947.
3. **Amazon Reviews** - *Filatova (2012)* [6] used crowdsourcing to collect Amazon reviews, human-labelled by 5 annotators. This dataset was not used by its compilers for classification, hence there is no score available for comparison.

Table 2: Statistical breakdown of the datasets used in this project

Dataset	Usage	Dataset Size	Composition*	Avg. tokens	Benchmark Score
News Headlines	Training	28619	pos: 47.6% neg: 52.4%	11.2	0.897
Twitter Data	Training	17280	pos: 50.4% neg: 49.6%	19.8	0.947
Amazon Reviews	Testing	1254	pos: 34.9% neg: 65.1%	276.4	N/A

**pos* indicates positive instances (sarcastic) and *neg* indicates negative instances (non-sarcastic)

The training datasets used in this project are large enough to provide a representative view of their respective data sources, without exceeding our computational resources. Furthermore, they are approximately balanced; despite the fact that sarcasm is used infrequently in real-life. On an unbalanced dataset, a classifier can achieve high accuracy simply by annotating each data instance with the majority class label. As most machine learning algorithms are designed to optimise overall accuracy, it is important to provide them with approximately equal amounts of both sarcastic and non-sarcastic data, to prevent them from overfitting to either class.

[3] https://spacy.io/
[4] www.theonion.com
[5] www.huffpost.com

C Data Pre-Processing

Datasets of user-generated content are inherently noisy and ill-structured, as users are free to include spelling errors and inconsistent capitalisation. Pre-processing is necessary for removing as much of this noise as possible to reduce sparsity in the feature space; although, we must be careful not to remove useful features.

On Twitter, users can include hyperlinks and references to other users in their tweets; however, this can be a source of noise in the dataset therefore we remove them using regular expressions. In *Ptáček et al. (2014)* [21], hashtags are removed in pre-processing. However, *Liebrecht et al. (2013)* [13] showed that the sentiment contained in hashtags can be used to indicate sarcasm, such as the use of #not to suggest insincerity. In the simplest case, single-word hashtags can be included simply by removing the # symbol, however multi-word hashtags must first be parsed to separate the tokens. As this is beyond the scope of this project, we remove all hashtags with the exception of *#not* (which we substitute with *not*).

Punctuation is removed and text is transformed to lowercase in order to reduce sparsity induced by the optional inclusion of punctuation and variations in letter capitalisation. This ensures that are fewer representations of the same word, as tokens such as *'CAN'T'*, *'can't'*, and *'Cant'* are all mapped to the same single token - 'cant'. However, punctuation and capitalisation can be used for emphasis, which may in turn be indicative of sarcasm. Hence, we re-introduce this meaning in the form of additional lexical features using a procedure outlined in the feature extraction section. Finally, we remove numbers and stopwords, such as *'the'*, *'in'* and *'an'*, as they provide little additional meaning to a sentence despite the number of times they occur.

Table 3: Proportion of tokens represented in the GloVe dictionary

Dataset	**Original data**	**Processed data**
News Headlines	97.71%	97.71%
Twitter Data	89.62%	96.43%
Amazon Reviews	83.17%	98.22%

Table 3 shows the proportion of GloVe-representable tokens before and after processing, where the same processing techniques are applied to each dataset. GloVe is a word-embedding technique, detailed in the feature extraction section, where tokens are encoded as pre-computed numerical vectors. We use the number of tokens represented in the GloVe dictionary (containing 27 billion tokens) as a measure of quality, as uncommon or ill-formed tokens are ommitted from this dictionary. The quality of the Amazon and Twitter data vastly improved after processing, however the quality of the news headlines corpus remained consistent as the original data is written formally and in lowercase.

D Feature Extraction

Extracting meaningful data from large corpora is vital for enabling classifiers to make high-quality predictions. As such, it follows that in the context of natural language processing, feature extraction techniques should encode the underlying semantics of text into the numeric vectors they produce. We produce our dominant features using established vectorisation techniques such as GloVe [18] and ElMo [19]. However, we also experiment with a number of custom features in combination with our main features, generated using the procedures outlined in *section D.2*.

D.1 Main Features

Traditional Approaches – As our most standard techniques, *Bag of Words (BOW)* and *Term Frequency-Inverse document frequency (TF-IDF)* [24] are used as a baseline against which to compare more advanced approaches. In BOW, text is encoded as a single vector containing a count of the number of occurrences of each word. TF-IDF extends the BOW model by providing each word with a score that reflects its importance based upon its frequency within the document. These approaches create sparse inefficient representations, where each vector is the size of the vocabulary. Additionally, vectors produced in this way do not preserve the context within which words are found, although sarcasm is a highly contextual phenomenon.

Word Embedding – Introduced in Mikolov et al. (2013a) [16], *Word2Vec* describes a group of related models that were the first to produce high-quality vector representations of words, known as word embeddings. Word2Vec consists of a shallow (two-layer) neural network that takes a large corpus and produces lower-dimensional vectors than the sparse representations generated by BOW and TF-IDF; however, intrinsic statistical properties of the training corpus, which were key to earlier techniques, are neglected, therefore global patterns may be overlooked. To mitigate this, we use the *GloVe* [18] approach, which generates word embeddings by constructing an explicit word co-occurrence matrix for the entire corpus. Word embeddings appear in a high-dimensional vector space, whereby words that share a similar context are clustered together.

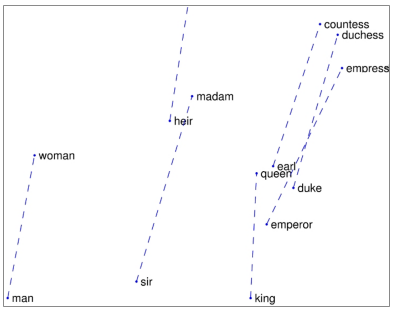


Figure 2: GloVe substructure[6]

GloVe is able to preserve the semantic relationships between words, even constructing analogies by composing vectors e.g. $king - man + woman \approx queen$. Likewise, it captures syntactic regularities such as the singular to plural relationship e.g. $cars - car \approx apples - apple$. GloVe is used in this project to produce word embeddings for each corpus, however it does not accommodate polysemy, which describes the co-existence of alternative meanings for the same word, and cannot generalise to words that were not specifically included in the training set.

Contextualised Word Embedding – Peters et al (2018) [19] introduced the *ELMo* model, and showed that ELMo markedly improved the state-of-the-art in a number of NLP tasks.

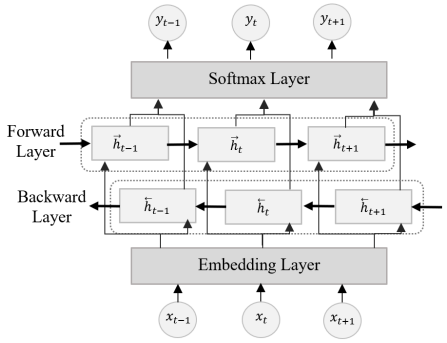


Figure 3: ElMo model pipeline

ELMo utilises a bi-directional LSTM *(long short-term memory)* model, concatenating the left-to-right and right-to-left LSTM in order to produce deep contextualised word representations. They are character based, therefore robust representations can be constructed for out-of-vocabulary tokens. However, rather than 'looking-up' pre-computed embeddings, ELMo generates them dynamically. Hence, it can disambiguate the sense of a word given the context in which it was found. These state-of-the-art contextualised embeddings are used in this project as the most advanced method of feature extraction.

[6]Figure obtained from https://nlp.stanford.edu/projects/glove/

D.2 Miscellaneous Features

We consider miscellaneous features as well as those produced by the aforementioned techniques - including lexical (punctuation) features, topic features and sentiment features. We focus specifically on the effects of incorporating our *novel sentiment features* in comparison to incorporating more conventional punctuation and topic features. We produce low-dimensional feature vectors for each of our datasets, concatenating those that are shown to improve classifcation scores with existing vectors.

1. Punctuation features – We extract punctuation features following the procedure outlined in Tsur et al. (2010) [26]. Five lexical features are combined to produce a 1×5 dimensional vector for each text snippet, including the length of the snippet in tokens, as well as the number of: "!" characters, "?" characters, quotations and tokens that include at least one capitalised character. These features are normalised by dividing them by the (maximal observed value · average maximal value of remaining feature groups).

2. Topic features – We use Latent Dirichlet Allocation (LDA) to perform topic modelling - this is an unsupervised machine learning technique where a fixed number of abstract "topics" are identified in a corpus. Each instance in a dataset is represented as the distribution of its topics, relative to the topics in the global dataset. The intuition is that positive and negative instances may each contain their own distinct assortment of topics, thereby informing future predictions on unseen sarcastic and non-sarcastic data.

3. Sentiment features – We produce sentiment features using the state-of-the-art *SentimentAnnotator* [25] by Stanford CoreNLP [14] which assigns a discrete sentiment label to text, where 0 is very negative and 4 is very positive.

Example: "i love when my train is late"

Tokens	**i**	**love**	**when**	**my**	**train**	**is**	**late**
Sentiment	2	4	2	2	2	2	1

Sentiment	**0**	**1**	**2**	**3**	**4**
Frequency	0	1	6	0	1
Proportion	0	0.125	0.750	0	0.125

Overall sentiment: 3.0
Feature vector: [0.0, 0.125, 0.750, 0.0, 0.125, 3.0]

For each instance in the dataset, we collect the sentiment values for each of the individual tokens, as well as the overall sentiment of the snippet. We contruct 1×6 dimensional feature vectors, where values 0 - 4 show the distribution of sentiment classes for tokens in the snippet, and value 5 shows the overall sentiment label.

E Binary Classification

In this section, we describe the machine learning and deep learning classifiers used in this project. These algorithms are trained on a set of rich features. For example, our machine learning classifiers are trained on Bag of Words, TF-IDF, GloVe and ElMo vectors, and we experiment with concatenating miscellaneous features where they have been proven to be informative in our binary classification task. We train each deep learning model on GloVe and ElMo vectors only, as it is computationally intractable to train neural networks on high-dimensional sparse features.

E.1 Machine Learning Models

We experiment with five candidate machine learning classifiers, evaluating each of them when trained on four types of features. Each of these supervised models performs binary classification, where unseen samples are predicted to be sarcastic or non-sarcastic. Brief descriptions of the models are provided below in the context of our specific binary classification problem.

1. **Naïve Bayes Classifier –** A group of probabilistic models that use Bayes theorem to make predictions; variations include Multinomial, Bernoulli and Gaussian Naïve Bayes. The naïve assumption that all features are independent of one another allows an estimate of $P(Sarcastic \mid x)$ (i.e. probability that a statement is sarcastic, given its feature vector, x) to be made using Bayes theorem.
2. **K-Nearest Neighbours (KNN) –** A non-parametric model which uses lazy learning at the time of prediction. An unseen sample is assigned the majority class label amongst its k nearest neighbours, where k is a hyperparameter to be finetuned. We found that k=5 achieved the optimal balance between the computational cost of a large number of neighbours and increased noise and bias associated with small values of k.
3. **Support-Vector Machine (SVM) –** An SVM [5] uses a kernel function to draw a hyperplane which divides a feature space into two subspaces. Features at the shortest euclidean distance (*margin*) to the hyperplane are known as support-vectors, and their distance from the hyperplane is maximised to create a decisive boundary between the sarcastic and non-sarcastic data. Our SVM uses a linear kernel as this model performed better than the SVM with radial basis function (RBF) kernel.
4. **Logistic Regression –** Models a binary dependent variable by finding a relationship between features and the likelihood that they belong to either class. It is a more advanced approach than linear regression, which aims to model data using a linear equation.
5. **Random Forest Classifier –** Random forest classifers [3] are an ensemble learning method which fit a number of decision trees on subsamples of the training set. The decision trees produce class predictions which are condensed into a single prediction by means of voting. As they are largely independent of one another, this low correlation allows them to produce collective predictions that outperform their individual predictions.

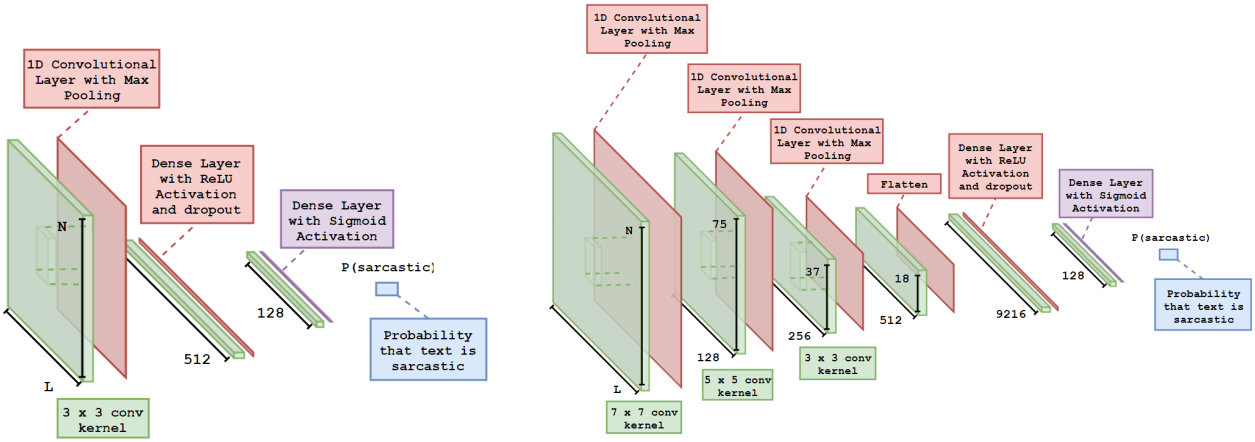
E.2 Deep Learning Models

Deep learning classifiers do not require the hand-crafted features used in supervised classifiers, and are instead capable of learning higher-level features from dense vectors such as ElMo and GloVe. We focus on two categories of neural networks - Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). While it is beyond the scope of this paper to describe in detail the concepts involved in CNNs and RNNs, we provide a brief explanation below. As we are performing binary classification, we use the Binary Cross-Entropy loss function.

Convolutional Neural Networks (CNN) [12] were popularised for their applications in computer vision and were designed to be used with image data [11]; however, they can be adapted for text-based classification tasks by replicating the two-dimensional structure of images. To create input features for the CNN, each token in a sentence is represented independently by a

fixed-size n-dimensional vector; as such, the sentence is represented by an $m \times n$ vector of feature weights, where m is the number of tokens in the sentence.
The characterising feature of a CNN is the convolutional layer, where filters are applied to feature vectors using the mathematical operation of convolution, and a max pooling operation is applied over each resulting feature map. Tokens in a sentence occur on a time axis, as the order that the tokens appear in gives rise to their meaning; hence, the filter slides downwards over the rows of the vector (*maximum pooling over time*) to produce a single output value. This is especially useful for classification tasks, where the desired output is a fixed-sized vector representing a class. Due to the high speed of the convolution operation, many convolutional layers can be used in a single network without excessivly slowing the training process.

We implement two convolutional neural networks and present their architectures in *figure 5*. We refer to one as a simple CNN as it contains only one convolutional layer, and to the other as a deep CNN as we incorporate three convolutional layers. With over 1.7 million trainable parameters, our deep CNN is more computationally complex and is slower to train than our simple CNN, which contains approximately 140 thousand trainable parameters. Both networks take an $N \times L$-dimensonal vector, where N is the length of each token vector, and L is the number of tokens in the snippet.



(a) Simple CNN Architecture

(b) Deep CNN Architecture

Figure 5: Illustration of Convolutional Neural Network Architectures

Recurrent Neural Networks (RNN) use feedback loops to process sequential data, creating a form of *memory* that allows information to persist in the network. RNNs maintain an additional hidden state, *h*, containing its own set of biases and weights. We can unroll this loop (*figure 6*) to reveal how outputs from previous time steps continue to propagate through the network. RNNs take an input sequence, denoted by *x*, and return an output sequence, denoted by *y*, which is a function of x and the hidden state. This hidden state is updated as inputs are fed through the network.

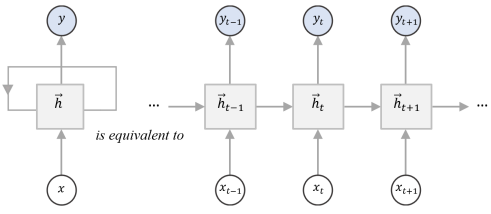


Figure 6: Illustration of RNN Cell

RNNs are limited to looking back only a few steps as they are susceptible to the *vanishing gradient problem*, more so when we continue to add complexity (in the form of additional layers) to our network. RNNs use backpropagation to calculate the gradient of the loss function with

respect to individual network weights, and these weights are updated in a manner that is proportional to the gradient. However, when the vanishing gradient problem occurs, the gradient with respect to the earliest layers becomes negligible; therefore the updates made to weights in these layers become insignificant. This can make the learning process slow and unstable, causing the network to have only a short-term memory.

Variations of the vanilla RNN such as *Long Short-Term Memory (LSTM)* [10] and *Gated Recurrent Units (GRU)* [4] use internal mechanisms, known as gates, which allow them to overcome the issue of short-term memory by controlling how much information is forgotten or allowed to pass through to the output. We implement an LSTM and GRU (as well as their bidirectional variants), comparing their performance against that of a standard RNN. Bidirectional models concatenate the output sequences of two recurrent layers, where one is trained on reversed input sequences and the other is trained on the original input sequences. This allows context to be captured from both directions, and can lead to improved predictions.

F Attention

When humans read text to answer a query or to inform a decision, we automatically focus more on specific regions that are more relevant to our query. This is known as *human visual attention* and has been well studied by psychologists; however, incorporating a similar form of attention in recurrent neural networks is a revolutionary new concept. The intuition behind this is such that not all components of text are equally relevant for classifying it as sarcastic or non-sarcastic; hence, this provides insight into which tokens contribute more or less in classification decisions. Additionally, incorporating an attention mechnism can lead to better classification performance. Extending our binary classification solution in this way enables us to answer the research question: *Can a custom model identify which linguistic cues correlate more to sarcastic labels?*

We employ the word-level attention layer outlined in Yang et al. (2016) [28] which uses context to identify relevant tokens, rather than filtering for key words irrespective of their context. This is an intermediate layer placed after the bi-directional RNN layer in our solution, as our attention mechanism utilises the forward and backward outputs of this layer. For a given word, w_{it}, the forward and backward hidden states of the bi-directional RNN are concatenated to obtain a sequence, h_{it}. This annotation harnesses the contextual information of the tokens that come before and after w_{it}; crucial for informing classification decisions and producing meaningful attention visualisations, as words can have different meanings depending on the context in which they are found. Figure 7 shows the word-level attention mechanism used in this project. Three trainable weights, W_w, b_w and u_w, are intialised from a random normal distribution.

Equation 1
$$u_{it} = tanh(W_w h_{it} + b_w)$$

Equation 2
$$\alpha_{it} = \frac{exp(u_{it}^T u_w)}{\sum_t exp(u_{it}^T u_w)}$$

Equation 3
$$s_i = \sum_t \alpha_{it} h_{it}$$

Each word annotation, h_{it} is fed through a fully-connected layer with tanh activation to produce u_{it} - this is the hidden representation of h_{it} (*Equation 1*). We apply the softmax function to the similarity between a word-level context vector (u_w), learned during training, and u_{it} (*Equation 2*). This produces an attention weight, α_{it}, which represents the relative importance of the token w_{it} in the sentence, s. Finally, the sentence-level context vector, s_i, is computed as the sum of $h_{it} \cdot a_{it}$ over all timesteps, t (*Equation 3*). Our attention layer returns the attention weight, a_{it}

for each token, as well as the sentence-level content vector, s_i, which is propagated through the remainder of our classifier. We visualise our attention weights by projecting them into a colour-space and displaying the tokens highlighted in their respective colours, where darker colours indicate a higher-level of attention.

RESULTS

A Evaluation Method

This section discusses our approach to evaluating the performance of the trained models. Each trained model makes predictions on labelled unseen data, and these predictions are compared to the corresponding ground truth labels. From this, we extract the number of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN). We use four metrics that leverage this data differently to produce an evaluation of the trained models.

Equation 4
$$Precision = \frac{TP}{TP + FP}$$

Equation 5
$$Recall = \frac{TP}{TP + FN}$$

Equation 6
$$F_1\ Score = \frac{2 \times precision \times recall}{precision + recall}$$

Equation 7
$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

Precision refers to the proportion of classified-sarcastic data that is *truly sarcastic* (how many positives are true positives), and recall describes the proportion of the truly sarcastic data that is classified as such (how many true positives are labelled as positives). We also use the harmonic mean of precision and recall (F_1 score), however this metric has faced some criticism for giving equal weight to precision and recall [9], therefore we also consider these measures separately. The Matthews Correlation Co-efficient (MCC) is a balanced measure which shows high scores if the true positive and negative rates are high and if the false positive and negative rates are low.

B Results of Sentiment Features

To contrast the effectiveness of our novel sentiment features with our standard punctuation and topic features, we feed them to a logistic regression model and calculate the F_1 score using 5-fold cross-validation; logistic regression is used as these feature vectors are small. This allows us to explore whether these features alone are indicative of sarcasm in different media types. Models achieving F_1 scores greater than 0.5 are considered to have been trained on useful features, as this is the average score of a binary classification model predicting randomly on a balanced corpus.

All three feature types were most informative when extracted from Twitter data *(table 4)*. Punctuation features were the most effective feature extracted from Twitter data, while those extracted from News Headlines actually hindered classification, as they were used to train the only model achieving an F_1 score lower than 0.5.

Table 4: F_1 score of logistic regression models trained on hand-crafted features

	Dataset	
Features	News Headlines	Twitter Data
Sentiment	0.516	0.593
Punctuation	0.344	0.611
Topic	0.525	0.530

Our topic features were the most informative feature taken from our dataset of news headlines, outperforming both sentiment and punctuation features. This may be due to certain subjects being associated more with sarcastic or non-sarcastic news.

C Binary Classification

To compare the performance of our machine learning and deep learning classifiers, each model is trained independently on numerous features extracted from our datasets. For each dataset, we report on only a subset of our most informative results, including our highest-achieving models and those that produced unexpectedly good or bad results. We contrast the performance of our machine learning and deep learning classifiers, to assess whether deep learning techniques outperform machine learning approaches.

Experimental Settings – In our machine learning experimentation, we train each classifier on BOW, TF-IDF, GloVe and ElMo vectors, employing 5-fold cross validation to ensure reproducability and consistency of results; however, it is infeasible to apply the same approach to our deep learning experimentation as training is substantially slower. We train each deep learning model on GloVe and ElMo vectors only, as it is computationally intractable to train neural networks on high-dimensional sparse features. The upper-bound on the number of training epochs is 300, although this is seldom reached due to the use of the *Early Stopping* callback function.

1. **Early stopping –** Training is interrupted when the validation loss stops declining - this is used to determine the appropriate number of epochs.
2. **Model Checkpointing –** We track and save the weights of the best performing model during training i.e. the model with the lowest validation loss.

Model performance peaks early in the training process for all of our models, although there is some fluctuation in the optimal number of epochs for different model types, hence early stopping is incorporated to stop training once model performance begins to plateau. The number of epochs for which no improvement in validation loss is seen (*patience*) is set to 10. Setting the patience too low can hamper a model that plateaus for a while before improving again. We allocate 90% of data to the training set, and the remaining 10% to the testing set.

C.1 News Headlines Dataset

Table 5: Binary classification results on the *News Headlines* dataset

Machine Learning Model	**Feature(s)***	**Precision**	**Recall**	**F_1 Score**
Random Forest Classifier	GloVe	0.774	0.735	0.754
Support-Vector Machine	TF-IDF	0.688	0.751	0.718
Random Forest Classifier	ElMo	0.801	0.793	0.797
Logistic Regression	ElMo	0.840	0.852	0.846
Logistic Regression	ElMo $\cdot$ S $\cdot$ T	**0.848**	**0.854**	**0.851**
Deep Learning Model	**Feature(s)**	**Precision**	**Recall**	**F_1 Score**
CNN	GloVe	0.828	0.816	0.822
Long Short-Term Memory	ElMo	0.853	0.888	0.870
Bidirectional LSTM	GloVe	0.855	0.873	0.864
Deep CNN	ElMo	0.883	0.879	0.881
Bidirectional GRU	ElMo	**0.892**	**0.906**	**0.899**

**S* refers to sentiment features and *T* refers to topic features

Our deep learning classifiers generally outperformed our machine learning classifiers *(table 5)*. In comparing our best-performing classifiers in both categories; we observe a 5.6% increase

from machine learning to deep learning. We achieve an F_1 score of 0.899 on this dataset using a bidirectional GRU, thereby exceeding the score achieved in Misra et al. (2019) by [17] 0.2%.

Trained on ElMo vectors, our advanced RNN models exceeded the F_1 score of our vanilla RNN (0.818) by an average of 7.7%, and our deep CNN achieved an increase of 2.2% compared to our simple CNN (0.862). Logistic regression with ElMo achieved the highest values for all three metrics of our machine learning classifiers; hence, we concatenate sentiment and topic features with our ElMo vectors and observe a further increase in F_1 score of 0.6%. The random forest classifier was the next best machine learning model on this dataset, although its F_1 score was 5.9% lower.

C.2 Twitter Dataset

Table 6: Binary classification results on the *Twitter* dataset

Machine Learning Model	**Feature(s)***	**Precision**	**Recall**	**F_1 Score**
Gaussian Naïve Bayes	Bag of Words	0.677	0.685	0.681
Random Forest Classifier	GloVe	0.669	0.623	0.645
Support-Vector Machine	ElMo	0.722	**0.789**	0.754
Logistic Regression	ElMo	0.758	0.766	0.762
Logistic Regression	ElMo · S · T · P	**0.794**	0.786	**0.790**
Deep Learning Model	**Feature(s)**	**Precision**	**Recall**	**F_1 Score**
Vanilla RNN	ElMo	0.844	0.811	0.827
Bidirectional LSTM	GloVe	0.839	0.825	0.832
CNN	ElMo	**0.861**	0.824	0.842
Bidirectional LSTM	ElMo	0.857	**0.861**	**0.859**
Long Short-Term Memory	GloVe	0.847	0.855	0.851

**S* refers to sentiment features, *T* refers to topic features and *P* refers to punctuation features

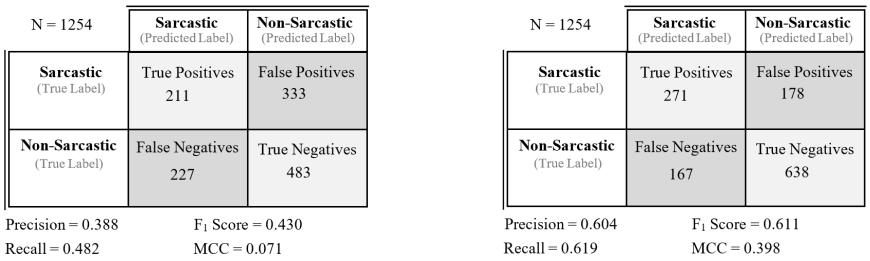
The performance of our deep learning classifiers exceed that of our machine learning classifiers *(table 6)*. In comparing our best-performing classifiers in both categories; we observe an increase of 8.7% from machine learning to deep learning. Our logistic regression model trained on ElMo vectors achieved an F_1 score of 0.762. In concatenating sentiment, topic, and punctuation features with our ElMo vectors, we observe an increase of 3.7% - leading to our highest F_1 score of our machine learning classifiers. Our bidirectional LSTM model trained on ElMo vectors attained the highest overall F_1 score of 0.859, as well as the greatest recall. While this model did not surpass the F_1 score of 0.947 attained in *Ptáček et al. (2014)* [21].

Our simple CNN achieved the highest value for precision; however, recall was signicantly lower than precision, indicating that this model is likely to have overfit to the non-sarcastic class. Nonetheless, it achieved an F_1 score only 1.2% lower than our deep CNN model (also trained on ElMo vectors). Our vanilla RNN performed better than expected on this particular dataset; hence, the average increase in F_1 score between our vanilla RNN and our more advanced RNN models is 3.3%, lower than our news headlines dataset.

Models trained on GloVe vectors showed poor performance on this particular dataset. For example, our gaussian naïve bayes model achieved a higher F_1 score when trained on basic BOW vectors than on GloVe. This could be due to the fact that this dataset has the lowest percentage of tokens present in the GloVe dictionary; hence, we lose their meaning in our embeddings.

C.3 Cross-Dataset Evaluation

Given that the Bidirectional LSTM and Bidirectional GRU achieved the highest F_1 scores on the Twitter and News Headlines dataset, respectively. It is crucial to the success of our solution that these models are able to generalise to other forms of media, hence we assess our candidate solutions on an evaluation dataset of 1254 Amazon reviews. This gives us an indication of which model is more likely to generalise to unseen data across various media types, as this is the most suitable model to employ in our tool.



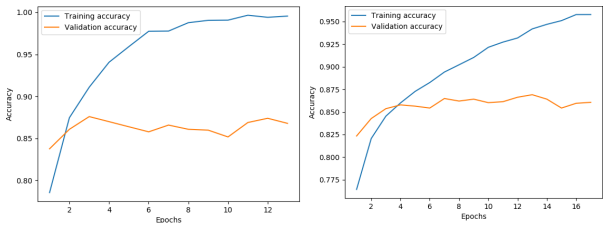
(a) Trained on News Headlines (b) Trained on Twitter Data

Figure 9: Confusion Matrices of models evaluated on Amazon Reviews

False positives and negatives make up 44.7% of our results for the News Headlines model, compared with 27.5% on the Twitter model. This is reflected in the summary statistics produced for each model, as all metrics demonstrated better performance for our Twitter model. Additionally, the Matthews Correlation Coefficient is very weak for our news headlines model (0.071), indicating that these predictions are only marginally better than random (where MCC = 0).

D Attention

The superior model in the binary classification task is shown to be the bidirectional LSTM model, trained on ElMo vectors from the Twitter dataset. We incorporate our attention mechanism after our bidirectional LSTM layer to identify the tokens that lead to positive and negative classification decisions. We present a number of attention visualisations, and examine whether incorporating attention can boost model performance as hypothesised.



(a) Without Attention **(b)** With Attention

Figure 10: Fluctuation of training and validation accuracy of models during training

Table 7: Comparison of bidirectional LSTM classification results

	Bidirectional LSTM	
Metrics	Without Attention	With Attention
Precision	0.847	0.859
Recall	0.851	0.865
F_1 Score	0.849	0.862

Incorporating our attention mechanism led to an increase in training time as the number of training epochs increased from 12 to 16 (*figure 10*). Furthermore, the validation accuracy increased at a steadier rate and peaked at a slightly lower value than in our non-attention model. However, the performance of our bidirectional LSTM with attention mechanism exceeds our foremost solution in the binary classification task, improving upon precision, recall and F_1 score by an average of 1.51% *(table 7)*.

EVALUATION

In the following section, we analyse the effectiveness of the solution, discussing its strengths and weaknesses in the context of our research questions: *Can handcrafted sentiment features improve sarcasm predictions? Do deep learning techniques perform better than machine learning approaches? Can a custom model identify which linguistic cues correlate more to sarcastic labels?*

A Evaluation of Sentiment Features

Figure 8 illustrates the distributions of the overall sentiment labels (value 5 in our sentiment feature vectors) in each dataset. The distribution of sarcastic and non-sarcastic overall sentiment scores are very similar in our news headlines dataset, however we see more variation in our Twitter dataset, indicating that these instances are easier to classify as their sentiment compositions are more distinct.

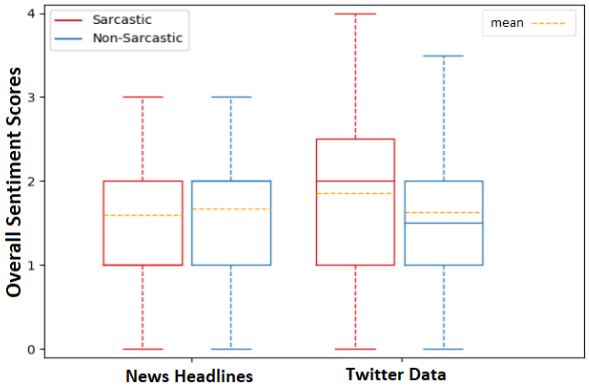


Figure 8: Visualisation of overall sentiment

While sentiment features were not the most or least effective feature type on either dataset, they were consistent in that they exceeded our 0.5 baseline on both datasets.

We found that sentiment features are objectively useful predictors of sarcasm in twitter data and news headlines, as logistic regression models trained on these features achieved F_1 scores of 0.593 and 0.516, respectively. We found that sentiment features were most informative when extracted from Twitter data as opposed to News Headlines, and we hypothesise that this is due to the use of more overtly sarcastic expressions that utilise strong sentiment, such as "*i love working weekends! #not*". As both models achieved F_1 scores exceeding 0.5, this indicates that their performance bettered that of a trivial classifier which predicts class labels with apparent randomness. Hence, we conclude that sentiment features are sufficiently informative so as to guide a model to make non-trivial predictions.

B Solution Strengths

We demonstrated the powerful capabilities of deep-learning architectures, as well as the potential for some machine learning models to compete with less sophisticated deep neural architectures when trained for a fraction of the time. For example, we achieved an F_1 score of 0.851 using a logistic regression model, trained with state-of-the-art contextualised word embeddings and hand-crafted sentiment and topic features on our news headlines dataset. We also exceed the state-of-the-art by 0.2% using a bidirectional GRU model on the news headlines corpus [17].

On an evaluation dataset of Amazon reviews, our solution achieved an F_1 score of 0.611 - despite being applied to an unseen dataset from a different form of social media. This demonstrates its ability to generalise to unseen data, suggesting that it successfully learned how to identify sarcasm, irrespective of the domain in which it is used. Additionally, our solution uses padding to allow variable-length input sequences of up to 150 tokens. Incorporating this flexibility allows

for inputs up to 7.6 times larger than the average tweet in our Twitter dataset *(table 2)*, hence we can continue to apply it to other forms of media where a larger input limit may be necessary.

We employed a novel approach to sarcasm detection by incorporating an attention layer into our bidirectional LSTM and extending our binary classification solution. Incorporating an attention mechanism allows us visualise the tokens correlating more to sarcastic labels. Furthermore, intergrating this intermediate layer improved our binary classification solution, leading to an increase in precision, recall and F_1 score by 1.4%, 1.6% and 1.5%, respectively.

Visualising the output of our attention layer shows that the model places more emphasis on qualitatively informative tokens.

C Solution Limitations

In general, our model achieved significantly lower scores when trained on Twitter data as opposed to news headlines. We did not exceed the state-of-the-art on this particular datastet. News headlines are written formally, hence there is little noise in this dataset due to the absence of erroneous spelling. These headlines are self-contained, unlike tweets which may be in response to another tweet thereby excluding necessary context. Additionally, due to the collection method used to collect tweets, some of the sarcastic instances may not be sarcastic, and some non-sarcastic instances may be sarcastic. The author's spelling and use of punctuation is also preserved, therefore there are a lot of spelling mistakes, and some tweets that are not written in English. This indicates that in order to achieve higher performance, alternative data cleaning methods could be applied to preserve necessary information used to inform predictions. Rectifying this limitation would have enabled our model to make enhanced predictions. It may have been beneficial to employ a dedicated Twitter tokeniser in order to harness nuanced features present in tweets, such as hashtags or emoticons. It was only feasible to evaluate a small range of online media types, however surveying a broader range of media types could give us a wider perspective of sarcasm in online media.

While there is evidence to suggest that sentiment features are useful, our results indicate that they are no more informative than conventional lexical features - two key findings were noted. Firstly, punctuation features extracted from our Twitter dataset (which was used to train our foremost model) were more informative than sentiment features *(table 4)*. Secondly, extraction of sentiment features was a reasonably slow process in comparison to our topic and punctuation features; hence, we would require our sentiment features to be significantly more informative than our other hand-crafted features in order for this to be worthwhile.

D Project organisation and approach

We began by evaluating machine learning classifiers, which are by their nature, likely to perform at a lower level than our deep learning classifiers. This enabled us to develop an intial solution whilst also giving us a baseline against which to compare the performance of our deep learning models. We followed an agile development model by iteratively expanding our collection of machine learning and deep learning models. This allowed us to gauge which types of models. For example, we discovered that tree-based classifiers rarely attained good scores, therefore we were able to look for alternatives early in the process. Similarly, our RNN models typically outperformed our convolutional architectures, potentially as they utilise time-series data more

effectively than CNN models. Hence, we focused on improving our RNN models, going on to incorporate their bidirectional variants, as they had already showed potential.

CONCLUSIONS

In terms of satisfying the objectives, this project was an overall success. We develop a sarcasm detection solution using a bidirectional LSTM with attention, trained on deep state-of-the-art contextualised embeddings. This model achieves an F_1 score of 0.862 on the Twitter dataset [21]. In order to achieve this, we constrasted the performance of numerous machine learning and deep learning classifiers; demonstrating that deep learning models are more effective at detecting sarcasm in news headlines and tweets. We also demonstrated that our solution can successfully generalise to other forms of media, despite being trained on tweets. Additionally, we incorporated our solution into a user-friendly tool with an interactive console, which may be useful to individuals who struggle to detect sarcasm in their daily-life, or for the industrial applications of enhanced opinion mining. In future, we may experiment with the use of this tool as a preprocessing stage for sentiment analysis, enabling us to filter out sarcastic instances that may skew sentiment predicitions. Our results show that sentiment features can be useful when extracted from ill-structured, user-generated data. Hence, as suitable direction for future work may be to explicitly incorporate sentiment features into our deep learning architectures to explore their potential in more sophisticated deep neural networks. Furthermore, it would be interesting to explore the use of sarcasm in a wider variety of media types, and to extend support to additional non-English languages.

References

[1] Francesco Barbieri and Horacio Saggion. Modelling irony in twitter. In *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 56–64, 2014.

[2] Santosh Kumar Bharti, Korra Sathya Babu, and Sanjay Kumar Jena. Parsing-based sarcasm sentiment recognition in twitter data. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, pages 1373–1380. ACM, 2015.

[3] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[4] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[5] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[6] Elena Filatova. Irony and sarcasm: Corpus generation and analysis using crowdsourcing. In *Lrec*, pages 392–398. Citeseer, 2012.

[7] Debanjan Ghosh, Alexander R Fabbri, and Smaranda Muresan. Sarcasm analysis using conversation context. *Computational Linguistics*, 44(4):755–792, 2018.

[8] Roberto González-Ibánez, Smaranda Muresan, and Nina Wacholder. Identifying sarcasm in twitter: a closer look. In *Proceedings of the 49th Annual Meeting of the Association for*

Computational Linguistics: Human Language Technologies: Short Papers-Volume 2, pages 581–586. Association for Computational Linguistics, 2011.

[9] David Hand and Peter Christen. A note on using the f-measure for evaluating record linkage algorithms. *Statistics and Computing*, 28(3):539–547, 2018.

[10] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[12] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[13] CC Liebrecht, FA Kunneman, and APJ van Den Bosch. The perfect solution for detecting sarcasm in tweets# not. 2013.

[14] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60, 2014.

[15] DG Maynard and Mark A Greenwood. Who cares about sarcastic tweets? investigating the impact of sarcasm on sentiment analysis. In *LREC 2014 Proceedings*. ELRA, 2014.

[16] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[17] Rishabh Misra and Prahal Arora. Sarcasm detection using hybrid neural network. *arXiv preprint arXiv:1908.07414*, 2019.

[18] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[19] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[20] Soujanya Poria, Erik Cambria, Devamanyu Hazarika, and Prateek Vij. A deeper look into sarcastic tweets using deep convolutional neural networks. *arXiv preprint arXiv:1610.08815*, 2016.

[21] Tomáš Ptáček, Ivan Habernal, and Jun Hong. Sarcasm detection on czech and english twitter. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 213–223, 2014.

[22] Antonio Reyes, Paolo Rosso, and Tony Veale. A multidimensional approach for detecting irony in twitter. *Language resources and evaluation*, 47(1):239–268, 2013.

[23] Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. Sarcasm as contrast between a positive sentiment and negative situation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 704–714, 2013.

[24] Stephen E Robertson and K Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information science*, 27(3):129–146, 1976.

[25] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality

over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[26] Oren Tsur, Dmitry Davidov, and Ari Rappoport. Icwsm—a great catchy name: Semi-supervised recognition of sarcastic sentences in online product reviews. In *Fourth International AAAI Conference on Weblogs and Social Media*, 2010.

[27] Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010.

[28] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489, 2016.

[29] Meishan Zhang, Yue Zhang, and Guohong Fu. Tweet sarcasm detection using deep neural network. In *Proceedings of COLING 2016, The 26th International Conference on Computational Linguistics: Technical Papers*, pages 2449–2460, 2016.

[30] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.