Sarcasm Detection in News Headlines

Stephen Obadinma
Department of Electrical and Computer Engineering
Queen's University at Kingston
16sco@queensu.ca

Gavin McClelland[*]
Department of Mechanical and Materials Engineering
Queen's University at Kingston
gavin.mcclelland@queensu.ca

Marshall Cunningham
Department of Mechanical and Materials Engineering
Queen's University at Kingston
20msc@queensu.ca

Abstract

In Natural Language Processing (NLP), sentiment analysis is used to assess the underlying meaning of text. However, this meaning can be skewed due to the presence of figurative language. One such mode of figurative language is sarcasm, where sarcasm detection is a sub-task of sentiment analysis that warrants further attention to understand the true meaning and intent of a piece of text. Existing deep learning approaches to sarcasm detection apply sequence modelling standards such as LSTMs to samples considered in isolation, or with added context (i.e. dialogue in online forums). In this paper, sarcasm detection is further explored by considering a dataset of news headlines consisting of satirical and non-satirical works. This dataset was chosen due to the explicit class labels and minimal pre-processing required. A previous paper using this dataset applied a hybrid neural network model consisting of convolutional neural networks and LSTMs. The performance of transformer models and convolutional neural networks are investigated, where the best observed performance surpassed the previous results. Sensible extensions of this work to support additional datasets and context are also discussed.

1 Introduction

In an era where text-based communication has become prevalent, there is a need to detect the presence of figurative language as it can skew the perception of information. This need is amplified in the case of digital media, where the integrity of information produced with the intent of mass consumption is paramount. As such, work in the Natural Language Processing (NLP) domain regarding sentiment analysis has increased in popularity in recent years (1). One mode of figurative language, sarcasm, is defined as a "sharp and often satirical or ironic utterance designed to cut or give pain" (2).

In text-based communication, it is difficult to detect language that deviates from the literal meaning due to the absence of tone, expression, and context. Moreover, sarcasm is particularly difficult to detect due to its ability to flip or shift the sentiment polarity of a sentence (3). This concept is highlighted by Figure 1. Early sentiment analysis approaches failed to detect sarcasm as the wrong

[*]Code (work in progress): https://github.com/gavinmcclelland/sarcasm-detection

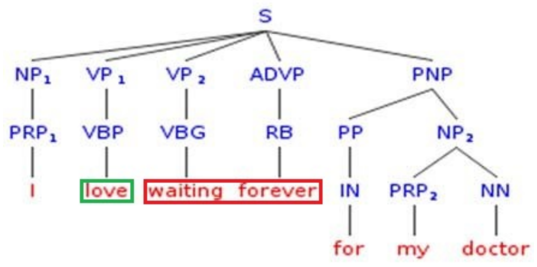

Figure 1: Sample parse tree of a sarcastic sentence, where the sentiment shifts from positive (green) to negative (red). This was adapted from (4).

sentiment polarity was inferred. It is therefore useful to detect the presence of metaphoric or satirical language in a piece of news to discern the true facts about a given event. Additionally, any progress made with respect to this sub-task of figurative language detection would push the boundaries of NLP.

To target this specific task, this work exposes two deep learning approaches to the main dataset entitled "News Headlines Dataset for Sarcasm Detection" from Kaggle (5). This dataset contains headlines from the popular satirical publication The Onion (sarcastic), and a legitimate publication, The Huffington Post (non-sarcastic). The first approach is a transformer-based model which applies a more sophisticated architecture to this dataset than the hybrid neural network approach in the original paper (5). This architecture was re-purposed from (6), and was optimized to assess the performance gap between the two approaches. The second approach uses convolutional neural network (CNN) modules used as part of a larger framework. This approach allows for different characteristics of language to be taken into account, and to fine-tune the baseline CNN model trained only on the sarcasm dataset. This approach was modified from (3) to experiment with this dataset.

The primary contributions of this work are as follows:

1. The best model–RoBERTa-RCNN–achieves a 94.4% test accuracy, which is superior to the original study using this dataset (5).
2. Using pre-trained CNN models (transfer learning) to fine-tune a baseline is an approach not commonly used in the NLP domain, as discussed in (3).
3. The frameworks detailed in this paper suggest sensible steps to be taken in future work, see Section six.

The rest of this paper is organized as follows: Section two provides a brief survey of literature specifically concerned with the task of sarcasm detection; Section three outlines the models considered; Section four details the experiments performed and supplementary datasets used; Section five analyzes the results obtained from the study; Section six details next steps for improving the approaches in this paper, along with other opportunities worthy of exploration; Section seven concludes the paper.

2 Related Works

The proliferation of NLP research over the past decade has yielded a strong foundation of reference material for this study. Early work in this field applied traditional machine learning models to a feature space engineered from a set of n-grams, lexical features, and more pragmatic features such as emoticons. Recent advancements in NLP such as pre-trained embeddings (GloVe, word2vec) and transformers have become popular choices for tasks such as Neural Machine Translation (NMT) and text-based classification, yielding state-of-the-art results for several tasks (7).

For the task of sarcasm detection, three papers served as primary resources. Firstly, this project was greatly inspired by Misra and Arora, who also compiled the headlines dataset (5). This paper explores this dataset using a hybrid neural network approach consisting of a bidirectional LSTM with an attention mechanism. Soon after this paper was published, Potamias et al. (6) applied a transformer-based approach to irony and sarcasm detection using another dataset. This approach consists of a pre-trained RoBERTa transformer and a recurrent convolutional neural network (RCNN). They were able to achieve state of the art results on datasets like SemEval-2018 Task 3. Prior to the popularization of transformers, Poria et al. (3) applied a CNN framework consisting of several modules pre-trained on different datasets to fine-tune the baseline sarcasm model. The intention for this project was to use the results obtained by Misra and Arora as a baseline, and to use the two approaches by Potamias et. al and Poria et. al as templates with a few modifications to observe how frameworks that were successful in other domains performed against this dataset. Lastly, the RoBERTa paper was referenced to offer insight into its performance benefits in comparison to the original BERT implementation (8).

Many different approaches are taken with regards to enhancing sarcasm detection in literature beyond classification of individual sentences. Context-based approaches are popular and have achieved great performance recently. This approach seeks to use additional context from earlier in a discussion to enhance the judgement as to whether a given sentence is sarcastic. Ghosh et al. (9) use LSTM networks with sentence level attention on the conversation context and response to improve performance compared to the model that only looks at a response without context. Similarly, Hazarika et al. (10) adopt a hybrid approach of both content and context-driven modeling for sarcasm detection. They extract contextual information from prior discourse in a discussion thread in addition to extracting user embeddings that encode the style and personality features of users to know their tendency to produce sarcastic remarks. They see a large performance boost over using only content-based features for classification. Similar improvements have been seen using prior conversation context in transformer-based models (11; 12; 13).

3 Models

Two architectures were explored and are outlined in the following sections. These were adopted from previous works and modified for this study (3; 6). Implementations were written in the Pytorch Deep Learning framework. This was made easier with references to these papers, as well as through the `RoBERTa` python package from "huggingface"[2].

3.1 RoBERTa-RCNN

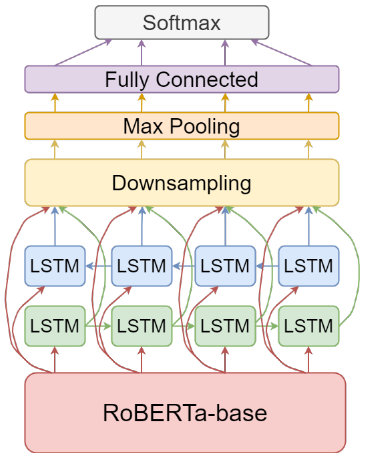

Figure 2: RoBERTa-RCNN Implementation (adopted from (6)).

This approach uses an end-to-end methodology with limited pre-processing. The architecture builds on the effectiveness of transformer-based models on downstream tasks by further enhancing their output through the use of an RCNN (14). Transformer-based models have achieved state of the art performance on many NLP tasks, particularly difficult tasks like natural language understanding and natural language generation (15). These high-capacity models are pre-trained on a massive number

[2]https://huggingface.co/transformers/model_doc/roberta.html

of general data and can easily be adapted to a downstream task of choice by pairing them with an alternate head. In this case, the head is an RCNN .

The overall architecture can be seen in figure 2. The key idea is to first use the transformer-based model, in this case RoBERTa (8), to generate rich word embeddings. A sentence **s** is passed through a RoBERTa tokenizer which converts the sentence into a series of *L* number WordPiece tokens. These tokens are passed into RoBERTa. RoBERTa outputs in its final hidden state a vector of size *L*, which consists of a series of individual word vectors of dimensionality 768 for RoBERTa base. These word embeddings are passed into an RCNN, which first contains a bidirectional LSTM (biLSTM) layer. This layer is used to capture unbiased temporally reliant dependencies within the word embeddings from both the left and right context (6).

The biLSTM consists of two different LSTM layers which both take in the same input at time step t. Each LSTM layer processes the sequence from a different direction which allows information from the left context and right context of each word to be utilized. Thus, the biLSTM produces two hidden states for each input. These two hidden states are d-dimensional vectors corresponding to a set hidden state size. For each word, these two LSTM vectors and the original RoBERTa embedding are concatenated into one vector and then go through a fully connected layer to downsample each word to have size d. Then a 1D max pooling operation is conducted over the entire sequence vector. The fully connected layer along with the max-pooling operation simulates a 1D convolution with a large kernel the length of the sentence. Essentially, for each dimension of the word vectors, the maximum element is chosen out of all the words in the sequence. This operation captures both spatial and temporal dependencies in RoBERTa's latent embedding space and can select the most discriminative features in the contextual information in the sequence (6). As a final step, the pooled outputs are fed into a linear layer and then the softmax function is used to get probabilities for each class for classification.

The advantage of using an RCNN rather than a plain CNN is that a standard CNN convolves over a fixed window of words to extract contextual information, while the RCNN takes advantage of its recurrent structure to capture a wider range of contextual information (14). It also allows more dependencies to be captured than simply using a single fully connected layer over the transformer's embeddings. RoBERTa is specifically chosen because Potamias et al. (6) empirically show how it performs the best out of all of the other transformer-based models they test like BERT, ELMo, XLNet etc. RoBERTa is trained primarily with a masked language modelling approach where it manages to learn bidirectional context by predicting randomly masked tokens (8). By learning to use the bidirectional context, RoBERTa can generate word representations that have a deeper understanding of contextual relations between words. This is in contrast with pre-trained word embeddings like GloVe (16) that manage to capture semantic similarity between words but are inflexible to the multiple different meanings and contexts of use of different words. Context is highly important for sarcasm detection as words have a far different meaning in a sarcastic context relative to their literal meaning (8).

3.2 Modular CNN

The motivation for this architecture was drawn from work using a basic CNN model for sentence-level classification tasks, which performed well on multiple benchmarks with little hyperparameter tuning (17). In this study, this basic CNN model was used in conjunction with the modular approach by Poria et. al to create the architecture shown in Figure 3 (3).

Each CNN module in this implementation is a slight variant from the model architecture specified by Kim for text classification (17). Here, pre-trained word embeddings from sentence s of variable size n are passed through one layer of convolution. Each convolution operation applies a filter to a windowed set of word embeddings to produce a new feature using

$$c_i = f(w * x_{i:i+h-1} + b) \tag{1}$$

In this case, w is a filter applied to a window of word embeddings size $i + h - 1$, where f is a `ReLU` activation, and b is a bias. This is repeated for each window of word embeddings in the sentence, where each output c_i is then used to create a feature vector $\mathbf{c} = [c_1, c_2, ..., c_{n-h+1}]$. Max pooling is applied to extract the maximum feature from **c**. This is common practice in sentiment analysis approaches as the intent is to capture the most important feature in the vector. In this implementation, multiple filter sizes are used to obtain multiple features from the same input. These features are then passed through a fully connected layer. The features from each module are concatenated together to create a feature vector from several pre-trained models. This concatenation is then passed through another fully connected layer with dropout, and finally through a softmax function to yield the probability distribution over class labels.

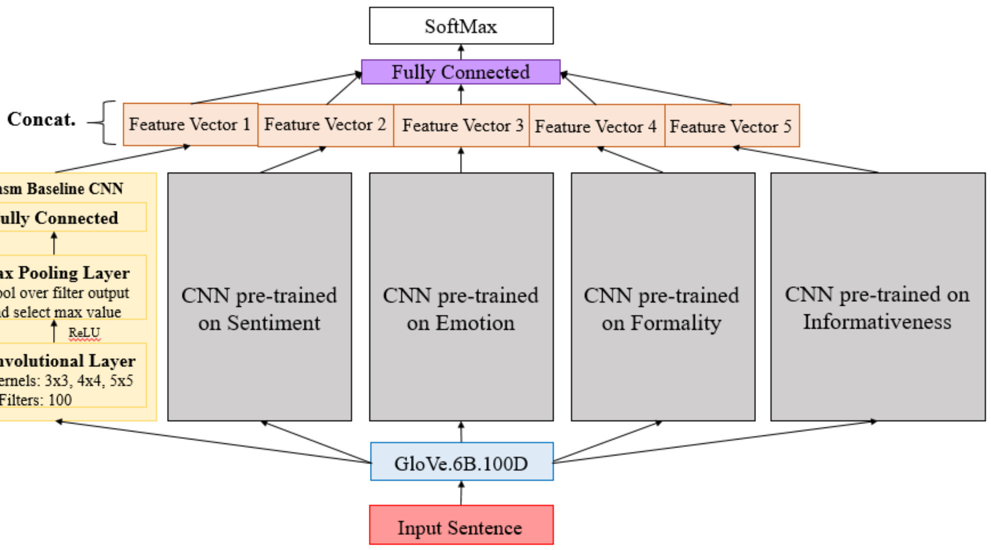

Figure 3: Complete architecture for the modular CNN model consisting of pre-trained modules.

The complete modular CNN architecture consists of several modules pre-trained on different classification tasks pertinent to detecting the presence of sarcasm. The purpose of these modules is to incorporate different aspects of language that can influence sarcasm detection. As such, a CNN–as described above–is trained on a benchmark dataset for classifying Sentiment, Emotion, Formality, and Informativeness to address these aspects independently. Then, a CNN was trained on the main sarcasm dataset used for this study to create baseline features, which are then supplemented by passing the embeddings for the same input sample through each pre-trained module. Enriching the feature vector in this manner allows for the overall model to capture different elements of language and local context when making a prediction.

4 Experiments

4.1 Datasets

4.1.1 Sarcasm Dataset

Sarcasm can be hard for humans to detect and as such, obtaining quality labeled data is challenging. Many datasets use distant supervision to amass large quantities of labeled data with no manual effort (18). Labels are created on the basis on inclusion criteria, usually the presence of sarcasm tags. This can lead to widely inaccurate labeling, as demonstrated by Riloff (2013) where only 45% of tweets collected with the "#sarcasm" tag were judged to be sarcastic by three high-quality annotators (19). One reason for the disagreement was the reliance on external content, such as other tweets or linked media, for the sarcasm of the tweet to be apparent. While manual annotation can filter out examples relying on external content, there exist two vulnerabilities. Firstly, the annotators' knowledge of named entities, everyday situations, and socio-political situations influence their ability to detect sarcasm in text (20). Secondly, perceived sarcasm and intended sarcasm have been shown to vary by 30-45% when comparing labels obtained from third-party annotators compared to labels obtained from the authors (18).

The "News Headlines Dataset for Sarcasm Detection" was selected because it avoids many of the common shortcomings of sarcasm datasets (5). It contains 26,709 records, 11,725 of which are sarcastic news headlines obtained from The Onion[3] and the remaining 14,984 are non-sarcastic

[3]https://www.theonion.com/

headlines obtained from Huffington Post[4]. The Onion publishes sarcastic versions of current events, while articles from the Huffpost "Cover a diverse range of topics and different perspectives in a sincere, relatable voice"(21). Therefore the labels of each record in the dataset accurately reflect its author's intent. Unlike social media posts, all news headlines are self contained, they do not refer to external content. Furthermore, there are no spelling mistakes or informal language usage because all news headlines are written by professionals in a formal setting. This drastically reduces the size of the vocabulary and increases the chance of finding pre-trained word embeddings.

4.1.2 Sentiment Dataset

The sentiment dataset (22) consists of 25,000 positive and 25,000 negative movie reviews posted on IMDb[5], with no more than 30 reviews per movie. The dataset contains only polarizing reviews and no neutral reviews.

4.1.3 Emotion Dataset

The Emotion dataset (23) consists of 11,327 records distributed among five classes: neutral, sadness, fear, anger, and joy. It was created by combining the DailyDialog (24), ISEAR (International Survey on Emotion Antecedents and Reactions) (25), and emotion-stimulus (26) datasets. The texts mainly consist of short messages and dialog utterances. Table 1 shows the class distribution of the emotion dataset.

Table 1: Class distribution of the Emotion dataset

Emotion Label	Count	%
neutral	2254	19.90
sadness	2317	20.46
fear	2171	19.17
anger	2259	19.94
joy	2326	20.54

4.1.4 Formality and Informativeness Datasets

The Formality dataset consists of a 7,032 sentences, rated on a Likert scale between 0 - 7 on the basis of both formality and informativeness. The sentences were rated by two groups of human annotators from Amazon Mechanical Turks[6]. The sentences were obtained from three main genres:

1. Posts from the top 100 blogs listed by Technorati[7] on October 31, 2009. (2110 sentences)
2. News articles from 20 news sites (five from each). The most common categories included “Breaking News”, “Recent News”, and “Local News” but no preference was used in the selection of the articles. (3009 sentences)
3. Forum documents taken from the Ubuntu Forums[8] and the TripAdvisor New York forum[9]. (2569 sentences)

For simplicity, the formality and informativeness datasets were then dichotomized where scores greater than 3.5 were assigned a label of "formal" and "informative" respectively, and scores less than or equal 3.5 were assigned labels of "informal" and "uninformative" respectively. While simplified, this results in heavily imbalanced datasets, where the formality dataset has 2518 formal sentences and 257 informal sentences. Similarly, there are 2601 informative sentences and 174 uninformative ones.

[4]https://www.huffpost.com/
[5]https://www.imdb.com/
[6]https://www.mturk.com/
[7]https://technorati.com/
[8]http://ubuntuforums.org/
[9]http://www.tripadvisor.com/ShowForum-g60763-i5-New_York_City_New_York.html

4.2 Evaluation Metrics

Metrics considered for quantitative evaluation were accuracy on the test split, F1 score, and expected calibration error (ECE). Formulae for accuracy and F1 score are included below:

$$Accuracy = \frac{TP + TF}{TP + TF + FP + FN} \tag{2}$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN} \tag{3}$$

where,

$$Precision = \frac{TP}{TP + FP} \quad (4) \qquad Recall = \frac{TP}{TP + FN} \quad (5)$$

Additionally, we assess the models in terms of calibration. For any classification task, the model should produce confidence scores on its predictions that match it's ability to correctly classify the data at that confidence level. For example if for 100 samples it predicts a confidence score of 0.7 on each sample, then it should classify 70 of those samples correctly. Any deviation shows miscalibration is present, which makes it difficult to trust the predictions of a classifier as it becomes impossible to know when the model is unsure or dealing with data it is unfamiliar with. Deep neural networks are known to be overconfident compared to shallower networks (27). This is particularly a concern for large transformer-based models like RoBERTa. When classifying sarcasm it is crucial that model does not output overconfident predictions, especially on sentences that it gets incorrect a lot. There are often a great number of sarcastic sentences that are hard to gauge as being sarcastic even for a human, and we want the model's probabilities to be able to reflect this difficulty. Expected Calibration Error (ECE) is a widely used metric for assessing the level of calibration error in a model's predictions, and we use it here to gain an idea as to how our uncalibrated architectures fare. The formula for ECE is included below:

$$ECE = \sum_{m=1}^{M} \frac{|B_m|}{n} |acc(B_m) - conf(B_m)| \tag{6}$$

ECE sorts data points by their predicted confidence score into M equally spaced bins and calculates the calibration error in each bin. Each bin's error is weighted by the amount of data points they contain. For our experiments, we use 15 bins to calculate the error.

4.3 RoBERTa-RCNN Training Details

For training the RoBERTa-RCNN architecture, we use a train/validation/test split of 80/10/10. We use AdamW (28) as the optimizer for the network's weights.

4.3.1 Hyperparameter Tuning

We conduct hyperparameter tuning using random search rather than grid search, because Bergstra and Bengio empircally show that random search over the same domain can find models that are as good as grid search or better in a fraction of the computational time (29). This is especially true when the number of hyperparameters that can vary is large, like in this case. Here we randomly choose a combination of hyperparameters than vary within a range. The hyperparameters we optimize include the number of epochs (1 to 4), the learning rate (2e-5 to 5e-5), batch size (16, 32, or 48), dropout (0.1 to 0.4), the number of biLSTM layers (1 to 3), and the LSTM hidden state size (64, 128, 256, or 768). The best hyperparameters we obtained after random search are given below:

- epochs: 4
- learning rate: 2e-5
- LSTM hidden units: 64
- batch size: 48
- dropout: 0.2
- layers: 1

We set the max sequence length of the inputs to be 64.

4.4 Modular CNN Training Details

For the CNN architecture we use the same train/validation/test split of 80/10/10 as before. Each individual pre-trained CNN module is trained using the same hyperparameters to save time as it is computationally expensive to tune each one individually to optimize the performance of the final model. The same structure is used for each module, with only the final layer differing. For the binary modules, the output layer is a single neuron that goes through a sigmoid function. For the emotion module the output layer has 5 neurons for each class which are fed into a softmax function. When the modules are combined for the final sarcasm model, we freeze the weights of all of the pre-trained ones, so that only the sarcasm module gets updated. For the main model, we only use each pre-trained module to generate its max pooled feature maps over the input sentence, and we ignore its classification outputs. For the each of the modules and the overall architecture we use Adam (30) optimizer to optimize parameters. For word embeddings we use pre-trained GloVe.6B.100D vectors since it is a common practice to use GloVe vectors for CNN text classification models. We limited the size of the vocabulary to 25,000 different words.

4.4.1 Hyperparameter Tuning

In a similar manner to the RoBERTa-RCNN, we use random search to find the ideal hyperparameters. The hyperparameters we optimize include the number of epochs (1 to 8), the learning rate (1e-4 to 1e-2), batch size (32 or 64), dropout (0.1 to 0.5), the number of filters for each filter size (100, 200, 300, 400, or 500), and the ideal combination of kernel sizes (2,3,4,5,7). Through the search, we found that the hyperparameters shown below gave the best results on the validation and test split of the sarcasm dataset:

- epochs: 8
- learning rate: 1e-3
- filters: 300 (100 for each filter size)
- batch size: 64
- kernel sizes: 3*100, 4*100, 5*100
- dropout: 0.5

5 Results

After hyperparameter tuning, the final results for the two implementations explored in this study are outlined in Table 2. Note that while the baselines achieved in the original work with the sarcastic headlines dataset are included for reference, they are independent from our results since different pre-trained word embeddings were used (word2vec instead of GloVe.6B.100D in this study).

Table 2: Comparison of implementations in this study and published results.

Model	Test Accuracy	Macro F1	ECE
CNN Baseline from (5)	84.88%	N/A	N/A
Attention Implementation (5)	89.7%	N/A	N/A
CNN Module (Headlines only)	87.69%	0.876	0.007
Complete CNN Implementation	89.4%	0.893	0.004
RoBERTa for Sequence Classification	93.3%	0.933	0.0030
RoBERTa-RCNN	**94.4%**	**0.944**	**0.0028**

In reflecting on these results, the power of transformer-based models is apparent. With a test accuracy of 94.4%, the RoBERTa-RCNN model achieves a 4.7% improvement over the original work with this dataset (5). While the original authors did use an attention mechanism to achieve their best results, transformers were in the early stages of development (2017). Additionally, the improved transformer-based used in this work–BERT (2018) and RoBERTa (2019)–were introduced at least one year after the time of the original publication by Misra and Arora (5). It is also clear that the RoBERTa-RCNN outperforms the RoBERTa for Sequence Classification model, which uses just a single layer fully connected layer to do classification, showing how it is an advantageous upgrade to use the RCNN head.

In terms of calibration, both of our models are surprisingly well calibrated having errors well under 1%, with the RoBERTa-RCNN having the lowest calibration error despite how we theorized it to be poorly calibrated. This is an interesting result and further analysis should be conducted as to why the deeper transformer-based models are not more overconfident than the shallower CNN models.

5.1 Modular CNN Ablation Studies

Prior to optimizing the modular CNN framework, individual CNN modules needed to be trained on the datasets outlined in section 4.1. The results from training each of these individual modules are included in Table 3. Note that here, the CNN module trained on the sarcastic headlines dataset yields baseline results for this work, meaning that the objective is to find a combination of modules surpassing 87.69% test accuracy.

Table 3: Results for training a CNN module on each dataset.

Model	Test Accuracy	Macro F1	ECE
Sarcasm Baseline	87.69%	0.876	0.007
Sentiment	84.21%	0.842	0.007
Emotion	73.00%	0.726	0.106
Formality	91.93%	0.673	0.000
Informativeness	94.43%	0.637	0.003

Ablation studies were conducted to find the optimal combination of CNN modules. First, the sarcasm module was trained in isolation to establish a baseline. Then, all modules were used to observe a slight increase in performance. Next, a non-exhaustive set of combinations were tested to find the optimal combination. The reason an exhaustive set of combinations was not used is because once a given module appeared to have a negative impact on the baseline, it was no longer considered. The optimal scenario was found to be using the weights from modules pre-trained on sentiment and emotion to fine tune the baseline sarcasm module. The results of these studies are shown in Table 4.

Table 4: Results from ablation studies on CNN module combinations.

Sarc.	Sent.	Emo.	Inf.	Form.	Test Accuracy	Macro F1	ECE
●					87.69%	0.876	0.007
●	●				82.47%	0.824	0.015
●		●			85.63%	0.856	0.013
●	●	●			**89.38%**	**0.893**	**0.004**
●	●	●	●	●	88.62%	0.885	0.007

One case that remains unclear is whether the formality and informativeness modules were detrimental to performance due to the small size and imbalanced nature of the dataset. Another remark of note is that the 89.4% test accuracy is just 0.3% less than the best result obtained in the original paper using this dataset (5).

6 Future Work and Additional Considerations

Given the time constraints placed on the authors, several avenues of exploration were left for further work.

- Only the RoBERTa transformer was considered in the transformer-RCNN implementation. Other state-of-the-art transformers could be implemented within the same transformer-RCNN hybrid model architecture.
- The CNN implementation used multiple kernel sizes to extract varying levels of features from the input word embeddings. A deeper architecture may yield better results by building up higher level features through multiple convolutional blocks utilizing small kernel sizes.

- The 100 dimensional GloVe.6B.100D word embeddings were used in the CNN implementation. Higher dimensional GloVe embeddings, or other pre-trained embeddings like word2vec, may result in different model performance (31).
- Only one dataset was used to benchmark the models in this study. The models presented could be tested on well known sarcasm datasets such as iSarcasm (18) and SemEval-2018 Task 3 (32) to test its generality. Testing on iSarcasm is particularly interesting because it is a new dataset that is set up to be much harder than others. Additionally, the model architecture could be trained on multiple datasets to improve its overall generality and performance.
- This study focuses on the detection of sarcasm in news headlines. Sarcasm is only one form of figurative language, and the models proposed could be extended to detect other forms of figurative language such as irony, satire, metaphors, understatements, overstatements, and others.

7 Conclusion

In this paper, experiments were conducted using transformer models and convolutional neural networks on the task of sarcasm detection using a dataset of explicitly labeled news headlines compiled by Misra and Arora (5). Using previous work using this dataset as a baseline (89.7% test accuracy), a significant performance increase of 4.7% test accuracy was observed through the implementation of a RoBERTa-RCNN model. This architecture uses a RoBERTa transformer and bidirectional LSTM to capture both temporal and spatial context. In parallel, a modular CNN framework was modified from Poria et. al using several modules pre-trained on different datasets to capture multiple features of natural language as they pertain to the concept of sarcasm (3). This approach yielded similar results to the best reported by Misra and Arora of 89.4% test accuracy. In future efforts, exploring the benefit of a deeper CNN architecture could be used to enrich the features extracted from text. Additionally, more datasets should be explored to understand how this experiment generalizes to samples from other text-based domains (i.e. social media). Different types of figurative language could also be considered with this framework. Lastly, an interesting application worthy of exploration would be sarcastic text generation given a text sample and context.

Acknowledgements and Contributions

This work was completed to fulfill ELEC825 - Machine Learning and Deep Learning course requirements for the Winter 2021 term at Queen's University. Co-authors would like to thank Prof. Xiaodan Zhu for the learning opportunity and computing resources this term. The contributions of each co-author are outlined below:

Stephen Obadinma: In the implementation, was responsible for developing the RoBERTa-RCNN, assisting in the CNN implementation, and hyperparmeter tuning both architectures. In writing the report, was responsible for part of section 2, section 3.1, part of section 4.2, and sections 4.3 and 4.4.

Gavin McClelland: In the implementation, was responsible for developing the individual CNN modules, pre-processing supplementary datasets where applicable, and assisted with finalizing the complete CNN architecture. Also performed repository maintenance when applicable. In writing the report, was responsible for the abstract, section 1, part of section 2, 3.2, part of section 4.2, section 5, and the conclusion.

Marshall Cunningham: In the implementation, was responsible for writing modular and scalable CNN training code and exploring deep CNN architectures. In writing the report, was responsible for section 4.1 and section 6.

Everyone helped edit the report.

References

[1] E. Cambria, S. Poria, R. Bajpai, and B. Schuller, "Senticnet 4: A semantic resource for sentiment analysis based on conceptual primitives," in *Proceedings of COLING 2016, the 26th international conference on computational linguistics: Technical papers*, pp. 2666–2677, 2016.

[2] Merriam-Webster, “sarcasm,” in *Merriam-Webster.com dictionary*.

[3] S. Poria, E. Cambria, D. Hazarika, and P. Vij, “A deeper look into sarcastic tweets using deep convolutional neural networks,” 2016.

[4] S. K. Bharti, B. Vachha, R. Pradhan, K. S. Babu, and S. K. Jena, “Sarcastic sentiment detection in tweets streamed in real time: a big data approach,” *Digital Communications and Networks*, vol. 2, no. 3, pp. 108–121, 2016.

[5] R. Misra and P. Arora, “Sarcasm detection using hybrid neural network,” *arXiv preprint arXiv:1908.07414*, 2019.

[6] R. A. Potamias, G. Siolas, and A. G. Stafylopatis, “A transformer-based approach to irony and sarcasm detection,” 2019.

[7] J. Howard and S. Ruder, “Universal language model fine-tuning for text classification,” *arXiv preprint arXiv:1801.06146*, 2018.

[8] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, “Roberta: A robustly optimized bert pretraining approach,” 2019.

[9] D. Ghosh, A. R. Fabbri, and S. Muresan, “The role of conversation context for sarcasm detection in online interactions,” 2017.

[10] D. Hazarika, S. Poria, S. Gorantla, E. Cambria, R. Zimmermann, and R. Mihalcea, “CASCADE: Contextual sarcasm detection in online discussion forums,” in *Proceedings of the 27th International Conference on Computational Linguistics*, (Santa Fe, New Mexico, USA), pp. 1837–1848, Association for Computational Linguistics, Aug. 2018.

[11] N. Jaiswal, “Neural sarcasm detection using conversation context,” in *Proceedings of the Second Workshop on Figurative Language Processing*, (Online), pp. 77–82, Association for Computational Linguistics, July 2020.

[12] H. Srivastava, V. Varshney, S. Kumari, and S. Srivastava, “A novel hierarchical BERT architecture for sarcasm detection,” in *Proceedings of the Second Workshop on Figurative Language Processing*, (Online), pp. 93–97, Association for Computational Linguistics, July 2020.

[13] K. Pant and T. Dadu, “Sarcasm detection using context separators in online discourse,” 2020.

[14] S. Lai, L. Xu, K. Liu, and J. Zhao, “Recurrent convolutional neural networks for text classification,” in *AAAI*, 2015.

[15] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. Rush, “Transformers: State-of-the-art natural language processing,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, (Online), pp. 38–45, Association for Computational Linguistics, Oct. 2020.

[16] J. Pennington, R. Socher, and C. Manning, “GloVe: Global vectors for word representation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Doha, Qatar), pp. 1532–1543, Association for Computational Linguistics, Oct. 2014.

[17] Y. Kim, “Convolutional neural networks for sentence classification,” 2014.

[18] S. Oprea and W. Magdy, “isarcasm: A dataset of intended sarcasm,” 2020.

[19] E. Riloff, A. Qadir, P. Surve, L. De Silva, N. Gilbert, and R. Huang, “Sarcasm as contrast between a positive sentiment and negative situation,” in *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 704–714, 2013.

[20] A. Joshi, P. Bhattacharyya, M. Carman, J. Saraswati, and R. Shukla, "How do cultural differences impact the quality of sarcasm annotation?: A case study of indian annotators and american text," in *Proceedings of the 10th SIGHUM Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, pp. 95–99, 2016.

[21] The Huffington Post, "Huffpost | about us," 2021. [Online]. Available: https://www.huffpost.com/static/about-us. [Accessed 21 April 2021].

[22] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, (Portland, Oregon, USA), pp. 142–150, Association for Computational Linguistics, June 2011.

[23] "Github - lukasgarbas/nlp-text-emotion: Multi-class sentiment analysis lstm, finetuned bert," 2021.

[24] Y. Li, H. Su, X. Shen, W. Li, Z. Cao, and S. Niu, "Dailydialog: A manually labelled multi-turn dialogue dataset," in *Proceedings of The 8th International Joint Conference on Natural Language Processing (IJCNLP 2017)*, 2017.

[25] K. R. Scherer and H. G. Wallbott, "Evidence for universality and cultural variation of differential emotion response patterning.," *Journal of personality and social psychology*, vol. 66, no. 2, p. 310, 1994.

[26] D. Ghazi, D. Inkpen, and S. Szpakowicz, "Detecting emotion stimuli in emotion-bearing sentences," in *International Conference on Intelligent Text Processing and Computational Linguistics*, pp. 152–165, Springer, 2015.

[27] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proceedings of the 34th International Conference on Machine Learning* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, pp. 1321–1330, PMLR, 06–11 Aug 2017.

[28] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2019.

[29] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. 10, pp. 281–305, 2012.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[31] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013.

[32] C. Van Hee, E. Lefever, and V. Hoste, "Semeval-2018 task 3: Irony detection in english tweets," in *Proceedings of The 12th International Workshop on Semantic Evaluation*, pp. 39–50, 2018.