Multimodal Sarcasm Detection Based on MUStARD Dataset

Yufei Wang **Zhen Huang** **Mona Gandhi** **Haoxin Chen**

Abstract

In the domain of natural langauge processing, sentiment analysis is often used to comprehend people's subjective view. However, such analysis often does not incorporate sarcasm detection, making the comprehension task difficult. As such, the capacity to recognize sarcasm is essential to appropriately interpret people's genuine purpose. According to the published study (Castro et al., 2019), several sarcasm detection models have been created in the past. In this term project, we intend to increase the accuracy of binary classification of sarcasm detection via attention based LSTM model on the MUStARD dataset, which captures the textual, audio, and video attributes around a sarcastic scenario. Our model is able to surpass the performance of model presented in Castro et al (2019), achieving 0.7027 on the F1-score.

1 Introduction

Sarcasm is a frequently used expression that is often employed in the context of making jokes, showing scorn, and delivering critiques. Detecting sarcasm might be a crucial problem for NLP applications in the real world, such as marketing research, opinion mining, and AI customer service. From a linguistic standpoint, however, sarcasm languages are drastically distinct from typical jokes and criticism in that they convey pleasant emotions in a negative context. Without verbal or visual clues, it is difficult for computers to identify sarcasm or not.

In this project, we use both verbal cues and nonverbal cues, such as tones of speech and facial expression, in order to perform a binary classification task on determining whether a given sentence and its corresponding non-verbal cues are sarcasm or not. In particular, we make use of the MUStARD dataset that was developed by Castro et al (2019). The data were obtained from YouTube and the MELD dataset, both of which include several episodes of famous television shows, such as Sheldon from "The Big Bang Theory" and Chandler from "Friends" are responsible for most of the caustic comments.

The picture in Figure 1 is a great example of sarcastic utterance from Friends, which consists of text, audio, and video frames. The background of this conversation was that someone stole Rachel's credit card and used it for online shopping for multiple times. Joey was asking if she call the police after Rachel figured out who the stealer was. However, Rachel's reaction was that she took the stealer to lunch. The sarcastic utterance was from Chandler said Rachel has her own brand of vigilante justice which satirize how Rachel treat someone who steal her credit card is totally insane and ridiculous in reality.

For this term project, the inputs of our model are the video frame, the audio of the sentences, and the text of the phrases, and the output is a classification result indicating whether sarcasm was present or not. The overall tasks of this project not only combinate the knowledge, homework, and quiz that we learned throughout the semester, such as sentiment analysis, semantic role labelling, etc., but also extend some unknown field for us to explore. This project is challenged but valuable; our group believe this project would be the best practice to applied what we learned into the real-world problems.

2 Literature Review

2.1 Previous work on MUStARD Dataset

The MUStARD Dataset we utilized for this term project was proposed in Santiago Castro's paper(Castro et al., 2019) on multimodal sarcasm detection and the paper also performed feature ex-

Figure 1: Sample sarcastic utterance in the dataset along with its context and transcript

tractions and built a couple of standard machine learning baselines for the classification task. For text features, they used the average of the last four transformer layers of the first token of the utterance to create a 768-dimensional vector. For audio features, they removed background noise and extracted local features using Librosa for each non-overlapping window of this utterance and then took the average over the windows. As for video features, they first resized, center-cropped, and normalized each frame in the video clip and then ran it through a pretrained ResNet model to get a 2048-dimensional vector representation and then took the average across the frames of the entire clip. They then performed a 5-fold cross validation evaluation on three baseline models: Majority, Random, and SVM. They discovered that the SVM with only the text features and the audio features have the highest F1-score of 0.631. We utilized the extracted features provided by this paper when building models for this project.

2.2 Neural Network usage in sarcasm detection

Aside from traditional machine learning models used in the original paper of the MUStARD Dataset, we decided to explore the utilization of deep learning for sarcasm detection. Two novel deep neural network models for sarcasm detection, namely ACE 1 and ACE 2, in the paper(Babanejad et al., 2020). Their models extend the architecture of BERT by incorporating both affective and contextual features. They directly alter the BERT architecture and train it from scratch to build a sarcasm classifier. They design and evaluate alternatives that materialize each of the two components (affective feature embedding and contextual feature embedding) of the proposed deep neural network architecture model. They evaluate the proposed models on datasets like Onion, Reddit and others (with only textual data) to find that they significantly outperform current state-of-the-art models for sarcasm detection.

2.3 The correlation between textual and visual features

The motivation of utilizing the cross-information between textual and visual features in the cross-model attention we built in our final neural network started from Yen-Chun Chen's paper(Chen et al., 2019) on joint image-text embedding. The paper introduces a large-scale pretrained model called UNiversal Image-TExt Representation (UNITER) for joint multimodal embedding. With the model core being a transformer, the model (transformer) was pre-trained on "masked Language Modeling (MLM) conditioned on image; [three variants of] Masked Region Modeling (MRM) conditioned on text; Image-Text Matching (ITM); and Word-Region Alignment (WRA)." MLM and MRM are used to recover words or image regions. ITM is used to align images and texts, and WRA is to provide more fine grained alignment on top of it. Specifically, the model takes a pair of image and sentence, and it will first embed the image regions with R-CNN and text tokens with BERT to a common embedding space; such embeddings will then be applied to the transformer mentioned before to obtain contextualized embeddings. The model was trained on four Vision and Language datasets and has achieved state of the art performance (that outperformed many multimodal pre-training methods at the time) when being evaluated on nine V+L datasets.

3 Method

3.1 Dataset

3.1.1 Data overview

The MUStARD Dataset was collected from YouTube and the MELD dataset that feature many popular TV show clips by the authors of the Multimodal sarcasm detection paper (Castro et al., 2019). Most of the sarcastic entries are from Sheldon from TBBT and Chandler from Friends. It consists of two major parts: a json file that contains all the text file with the format shown in Figure 2 and 690 videos of the specific sentence (the utterance) that we are classifying and the corresponding 690 slightly longer videos that contains the context of the utterance.


```
{
  "1_60": {
    "utterance": "It's just a privilege to watch your mind at work.",
    "speaker": "SHELDON",
    "context": [
      "I never would have identified the fingerprints of string theory in the aftermath of th
      "My apologies. What's your plan?"
    ],
    "context_speakers": [
      "LEONARD",
      "SHELDON"
    ],
    "sarcasm": true
  }
}
```


Figure 2: Example of one entry in the json file

From Figure 2 we can see that the textual part of the input data contains information about the character who are engaged in the conversation context and the speaker of the utterance specifically in addition to the utterance itself and the context sentences. The label of the utterance is under the key "sarcasm". The Dataset is very balanced as it contains a total of 345 utterances that are labeled as Sarcastic (True), and the other 345 utterances are labeled as Non-sarcastic (False). We split the data into training 80% (552), development 10% (69), and testing 10% (69) randomly.

3.1.2 Feature Extraction

To extract features that can be processed by models from the raw texts and videos, we decided to follow the methods that were utilized in the MUStARD Dataset paper(Castro et al., 2019). However, due to constraints of computing resources we currently have, we ended up directly using the pre-extracted features provided along with the Dataset.

- **Text**:
  As stated in the previous literature review section, for each utterance and each sentence in the context, a [CLS] token was put in front of the sentence and then the sentence is passed into a pre-trained BERT model. The average of the outputs of the last four transformer layers of the [CLS] token is calculated as the embeddings of the corresponding sentence. Thus, for each utterance sentence we have a 1-D vector of size 768 and for each context we have a 2-D vector of size [number of sentences in the contex] * 768. In this term project we essentially utilized only the utterance BERT embeddings for building our models.

- **Audio**:
  Audio features are processed by Librosa to extract local features across non-overlapping window after removing background noise. Each audio feature for the utterance is of different size due to the length of the sentence. We then pad the audio features with 0 to the maximum length in the certain batch of data input.

- **Visual**:
  For both the utterance videos and the context videos, the frames were first resized with a new height of 256 while keeping the ratio of width and height unchanged, center-cropped, and normalized. Then they were passed into a pre-trained ResNet model which output a 2048-size vector.

3.1.3 Problem Formation

The problem we want to address in this term project is essentially a binary classification of features extracted from three different types of input data.

3.2 Performance Metrics

Since the problem is essentially a binary classification of *sarcasm* and *not sarcasm*, we chose accuracy, precision, recall, and f-1 score as our four major evaluation metrics. We employed the sklearn functions for binary classification: accuracy_score, precision_score, recall_score, f1_score, for which the input is the predicted value and the gold labels. Specifically, when measuring the performance of our models, because f-1 score is a harmonic mean of precision and recall, we decided to use f-1 as a metric to see our model's performance.

3.3 Simple Baseline

3.3.1 Majority Class Baseline

The majority class baseline takes the majority class label in the training data and predicts everything

	Accuracy	Precision	Recall	F1 score
Train	0.5036	1.0	0.5036	0.6699
Dev	0.4348	1.0	0.4348	0.6060
Test	0.4348	1.0	0.4348	0.6060

Table 1: Performance of Majority Class

	Accuracy	Precision	Recall	F1 score
Train	0.9674	0.9674	0.9674	0.9674
Dev	0.5429	0.5143	0.5454	0.5294
Test	0.6470	0.5588	0.6786	0.6129

Table 2: Performance of Logistic Regression

as that label. This baseline is meant to serve as an indication of what kind of evaluation metric results we can get by doing the minimum action more than randomly guessing. Because the data consists of equal number of entries labeled as *sarcasm* and *not sarcasm*, the majority is determined by the random seed that decides how the data are separated in *sklearn*'s train_test_split function. As a result, the accuracy of the majority baseline is close to 50%. Precision of 1 indicates that the majority label in the training data is sarcasm.

3.3.2 Logistic Regression

The original paper (Castro et al., 2019) utilized a SVM model on the MUStARD dataset to perform the classification using the extracted features, we implemented logistic regression, a non-deep learning classification model, as another simple baseline. However, the features for the available dataset are multidimensional and that of varying sizes. For simplicity, we picked the Bert embeddings of the utterance texts as input, since it is normally served as the input for monomodal sarcasm detection problems. For each sample, the embedding is a 768-dimension vector. Because the number of features vastly outnumbers the data instances, the model overfits the data, as indicated by f-1 score of 0.96 for the training set, but only 0.53 and 0.61 for development and test sets. Furthermore, we have experimented with creating an ensemble classifier that averages the probabilities of five logistic regressions using the utterance text feature, utterance video feature, context text feature, context video feature, and the audio feature, the model severely overfits, resulting in a f-1 score of 1 on the training data.

	Accuracy	Precision	Recall	F1 score
Train	0.8025	0.8849	0.7616	0.8186
Dev	0.7246	0.8108	0.7142	0.7595
Test	0.6232	0.8000	0.5455	0.6486

Table 3: Performance of LSTM

4 Experiments and Results

4.1 Strong baseline

We trained an LSTM classifier for each of our features for our strong baseline. We use three features – BERT embeddings for utterance, embedding for audio features and ResNet embeddings for utterance video. After tuning, the structure of the LSTM is as follows:

1. LSTM: A one layer, unidirectional LSTM with a hidden dimension size of 300. A separate LSTM of the same structure is applied to each of the three features.
2. Concatenation: The output of each LSTMs are concatenated together, creating a matrix with 900 columns.
3. Fully connected layer 1: A fully connected linear layer that maps the concatenated output to size 300.
4. Activation: ReLU is used here.
5. Fully connected layer 2: A fully connected linear layer that maps the input to size 1.
6. Sigmoid: A regular sigmoid function is used to compute the probabilities of falling into the *sarcasm* and *not sarcasm* categories.

Our task is binary classification, hence we use Binary Cross-Entropy Loss as the loss function and Adam as the optimizer. We tuned the learning rate to be 0.001 and the number of epochs to 15. We saved the best model based on the f-1 score for the development set from all the epochs and evaluated our test set only on the best model. We got our best model at epoch 11. We observed a significant improvement on the development set but a smaller improvement in the f-1 score on the test set. The LSTM model has good precisions for all three: train, development and test sets.

4.2 Extension

Since our main focus of this project is multimodal, we decided to implement different kinds of cross-model attention as our extensions. Cross-model

attention enables the LSTMs of different types of features to gain more information about each other and a weighted sum result based on the attention scores can help the model to focus on the relevant parts of the feature sequences. While it is possible to compute all combinations of attention and concatenate them with the LSTM output, our limited data size put a constraint on the number of parameters in our model (to avoid overfitting). Thus, we implemented 4 different types of attention that only perform attention on the textual feature and the video feature. The attention scores are calculated by combining the outputs of the LSTM models, applying softmax to obtain the weights. These weights are multiplied with the output of the LSTM that takes in video features, which returns a weighted sum of the video features. This is then concatenated together along with the outputs from the three LSTM models. Therefore, using the same hidden size of 300, the concatenated result will have size 1200. The final model structure is shown in Figure 3.

The four methods of calculating attentions are listed below

1. hidden_out: Dot product of textual LSTM hidden states and video LSTM outputs
2. out_out: Dot product of textual LSTM outputs and video LSTM outputs
3. hidden_hidden: Dot product of textual LSTM hidden states and video LSTM hidden states
4. out_out_general: Dot product of textual LSTM outputs after a matrix transformation (in the form of a learnable fully connected layer) and video LSTM outputs

Finally, we noticed that the criterion *BCEWithLogitsLoss* takes into account of the sigmoid. Therefore, we switched the criterion from *BCELoss* to *BCEWithLogitsLoss* and removed the final sigmoid layer in the LSTM model.

Comparing the four attention mechanics as we can see in Table 4, all but hidden_hidden have shown significant improvements in performance, with out_out achieving the highest f-1 score of 0.7027 on the test set, a 9% improvement compared to the logistic regression and a 5% improvement compared to the LSTM model without attention. From Figure 4, we can see that hidden_hidden overfits on the training data and hence performs poorly on dev and test data. While none of the attention types performs better than LSTM without attention on the dev set, other than hidden_hidden, all the attention types does better than all the baselines on the test data.

LSTM w/ Attention	Accuracy	Precision	Recall	F1 score
hidden_out	0.6232	0.9667	0.5370	0.6905
out_out	0.6812	0.8667	0.5909	**0.7027**
hidden_hidden	0.6522	0.5667	0.6071	0.5862
out_out_general	0.7101	0.7000	0.6563	0.6774

Table 4: Test Set Performance of LSTM with Attention

Hyperparameter	Value	Accuracy	Precision	Recall	F1 score
Learning rate	0.01	0.5362	1	0.5362	0.6981
	0.001	0.6811	0.8648	0.653	**0.7441**
	0.0005	0.6666	0.8648	0.64	0.7356
	0.0001	0.6811	0.5405	0.8	0.6451
Hidden size	100	0.6521	0.8378	0.6326	0.7209
	200	0.6086	0.5405	0.6666	0.597
	300	0.6811	0.8648	0.653	**0.7441**
	400	0.5797	0.4324	0.6666	0.5245
Batch size	8	0.5797	0.4324	0.6666	0.5245
	16	0.5797	0.3783	0.7	0.4912
	32	0.6811	0.8648	0.653	**0.7441**
	64	0.6376	0.7567	0.6363	0.6913

Table 5: Different Hyperparameters tried on the Development set to select the ones that gave the highest F1 score. For each hyperparameter tried, the other two are kept constant.

4.3 Hyperparameter Selection

Refer Table 5, to see all the hyperparameters we tried to get the best model. For tuning a given hyperparameter, we keep the others set to a constant value, for example, for fine tuning the learning rate, the hidden size was fixed to 300 and the batch size to 32. We have used learning rate as 0.001, hidden size as 300 and batch size as 32 for the experiments and results.

4.4 Error Analysis

4.4.1 Quantitative Analysis

When evaluated on the Test set, the best model LSTM with out_out attention does very well on detecting sarcasm for real sarcastic data, but performs poorly in detecting non-sarcastic data apart from sarcastic ones. As seen in Figure 5, there are more false positives than true negatives, hence we can conclude the model learns a bias towards labeling data as sarcastic. We can be almost sure that when the model predicts a data-point as non-sarcastic, it would not be sarcastic. However, the reverse is not true.

4.4.2 Qualitative Analysis

On conducting some qualitative Analysis, we find that most of the data which was labeled as False Positive had a common feature of the audio being

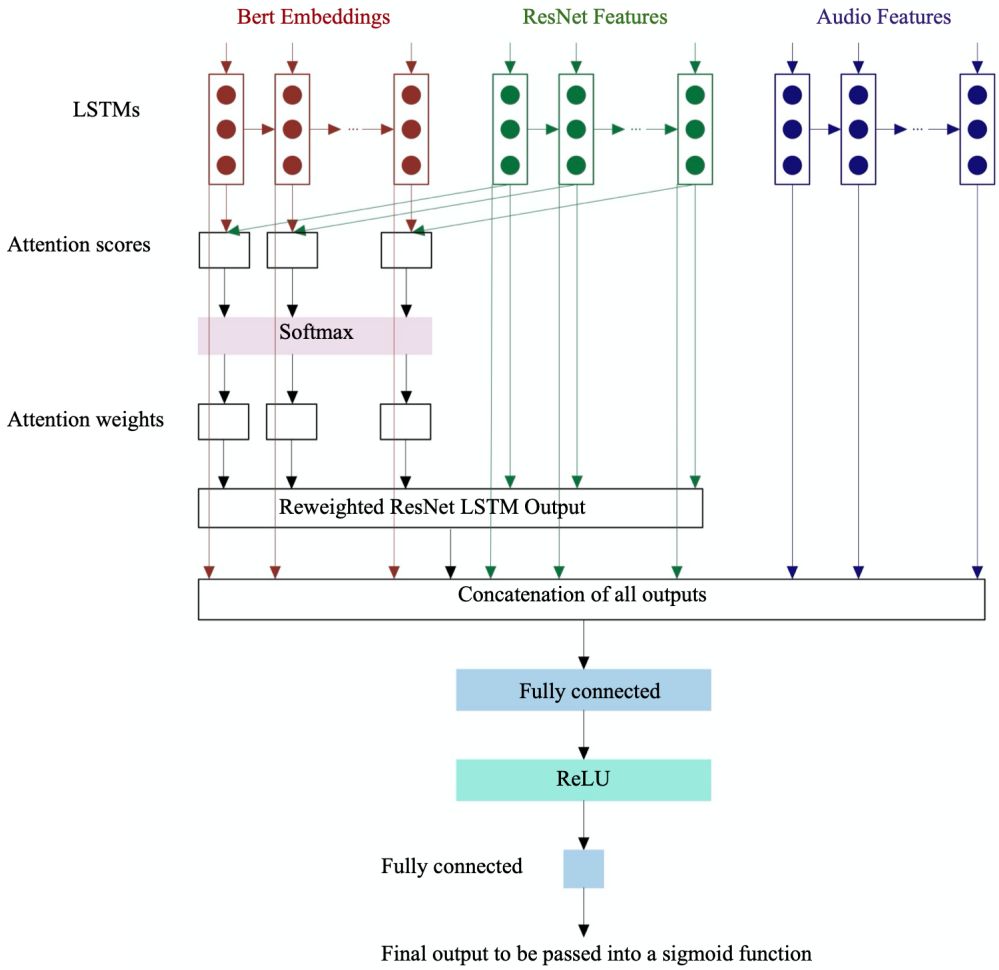

Figure 3: Model Architecture for LSTM with out_out Attention

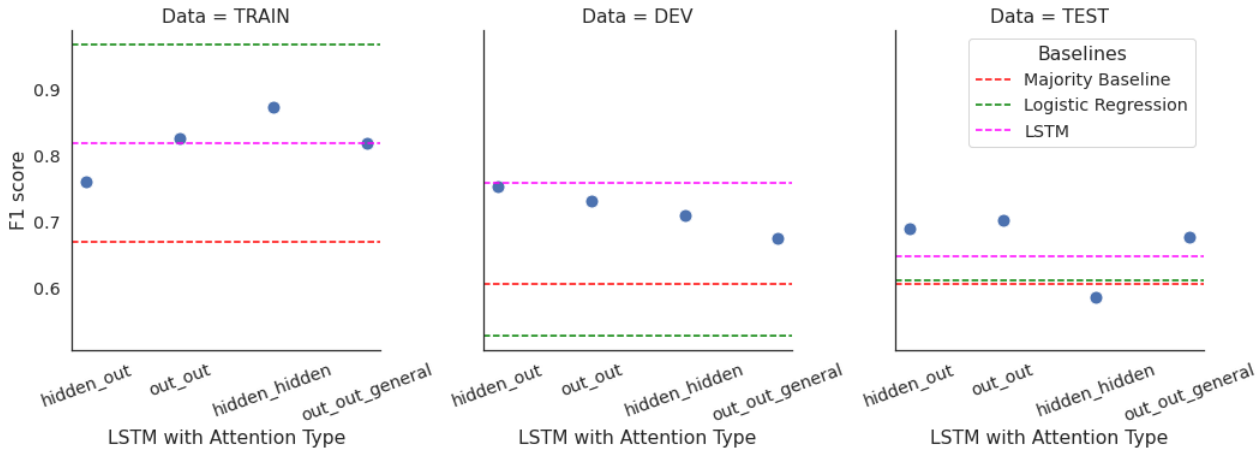

Figure 4: Comparing results for different Attention types implemented with LSTM for Train, Dev and Test datasets, we can see that LSTM with all the attention types except for hidden_hidden perform better than simple baselines and LSTM without attention.

high pitched. We did not include speaker as a feature in our model as we figured out it would lead to a bias in the model as some characters are more likely to be sarcastic than others. However, the model still picks some bias towards some characters (eg. Sheldon, Chandler). Many video clips

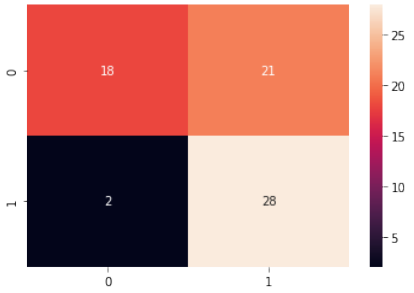

Figure 5: Confusion Matrix for the Best Model (LSTM with out_out attention) evaluated on the Test set

had a joke followed by a laugh which the model wrongly labeled as sarcastic. Some of the test data was very difficult to be recognised as sarcastic or non-sarcastic just from the utterance, that is where the context feature plays an important role. For example, the speaker saying "I am fine" and context being that the speaker just found out his ex-girlfriend is going on to date his friend, here we could not have understood the sarcasm just from the utterance at all. This is the cause for the two False Negatives on the Test set.

5 Conclusions

Sarcasm detection is a practical problem that can lead to multiple applications in the real world. In this term project we mainly built a Multimodal LSTM model with cross-model attention to perform the binary classification task and we are able to reach a F1-score of 0.7027 on the testing set, which is higher than the F1-score presented in the published paper (Castro et al., 2019). Admittedly, the MUStARD Dataset we used is not a large Dataset, we are constrained by the limited data when building the attention scheme of the final model as well as when selecting the hyperparameters that relate to the complexity of the LSTM like hidden size. When performing hyperparameter selection, we do observe that a hidden size of 400 lead to lower F1 score even than the Majority class baseline. We expect to explore more on cross-model attention scheme with more data.

6 Acknowledgements

We deeply thank our Mentor TA Artemis Panagopoulou for all of her invaluable advice regarding baseline model selections and possible extensions we could work on. She was always able to point us in the right direction and remind us about practical problems like computing resources.

We also want to give credits to the authors of the MUStARD Dataset as we have utilized their extracted features on the three different type of input to avoid being limited by our restrained computing resources.

References

Nastaran Babanejad, Heidar Davoudi, Aijun An, and Manos Papagelis. 2020. Affective and contextual embedding for sarcasm detection. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 225–243, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Santiago Castro, Devamanyu Hazarika, Verónica Pérez-Rosas, Roger Zimmermann, Rada Mihalcea, and Soujanya Poria. 2019. Towards multimodal sarcasm detection (an _Obviously_ perfect paper). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4619–4629, Florence, Italy. Association for Computational Linguistics.

Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. 2019. Uniter: Universal image-text representation learning.