Sarcasm Detection on Reddit

Group 18:

Amas Lua Yong Da
Chua Wee Kian, Glendon
Divakaran Haritha
Lee Joon Jie
Senthil Araavind

TABLE OF CONTENTS

1. Introduction

Motivation

" Sarcasm detection is an important component in many natural language processing (NLP) systems, directly relevant to natural language understanding, dialogue systems, and text mining. " *A Large Self-Annotated Corpus for Sarcasm*

Sarcasm is defined as the **use of irony to mock or convey contempt**.

Automatic detection of sarcasm may help to prevent internet users from misinterpreting comments which enables a more friendly user base.

Motivation

Another practical use of sarcasm detection is to avoid PR disasters

Credits: Image from https://www.india.com/viral/indigo-airlines-misplaced-this-guy-luggage-and-couldnt-even-understand-his-sarcastic-tweet-2097264/

The Task

This project aims to build effective models to determine if a comment made by a Reddit user is sarcastic.

We wish to find out which features are correlated to sarcasm and we thus attempt to perform NLP and feature engineering to help with the task.

4 models will be explored and compared to find out which one can best predict sarcasm. In addition, we perform ensembling on the models to see if it improves performance.

Prior Work

- The paper *A Large Self-Annotated Corpus for Sarcasm* explores using logistic regression with 3 text representation methods

Method	all-bal*
Bag-of-Words	73.2
Bag-of-Bigrams	75.8
Sentence Embedding	71.0
Human (Average)	81.6
Human (Majority)	92.0
Random	50.0

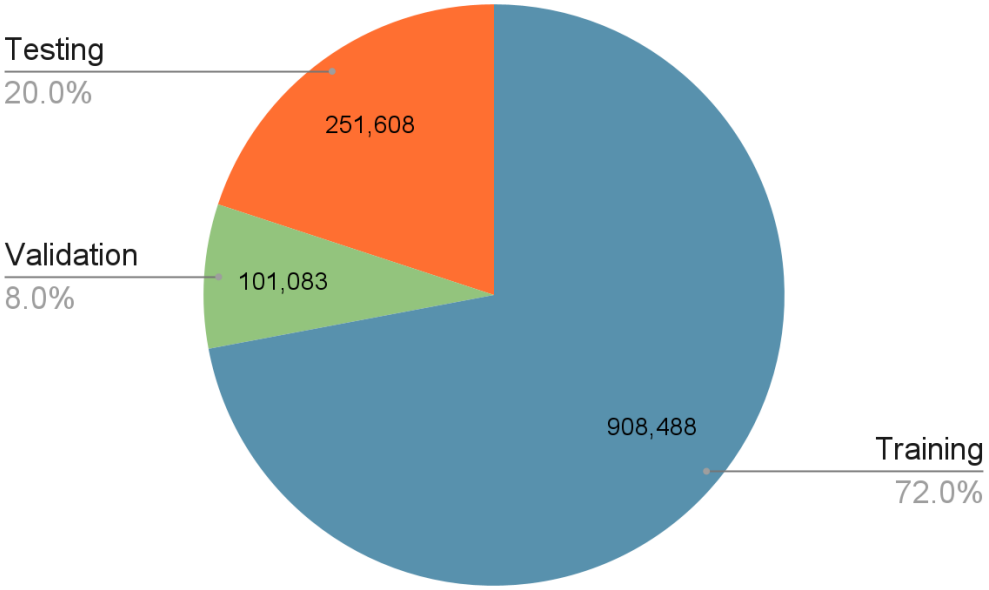
Table: Accuracy percentage of baseline methods for sarcasm detection

2. Data

Data Source

- **Self-Annotated Reddit Corpus**' from *A Large Self-Annotated Corpus for Sarcasm*
- Over 1 million comments scraped from Reddit
- Balanced training and test sets with instances that are labelled with 1 (sarcastic) or 0 (not sarcastic)
- Sarcastic comments are identified with the '/s' tag
- Useful features: comment, parent comment, subreddit, upvotes, downvotes
- Primary data is the comment text but we also explore incorporating other potentially useful features

Data Split


Testing
20.0%
251,608
Validation
8.0%
101,083
908,488
Training
72.0%

3. General Approach

Models

01
SVM
Margin classifier

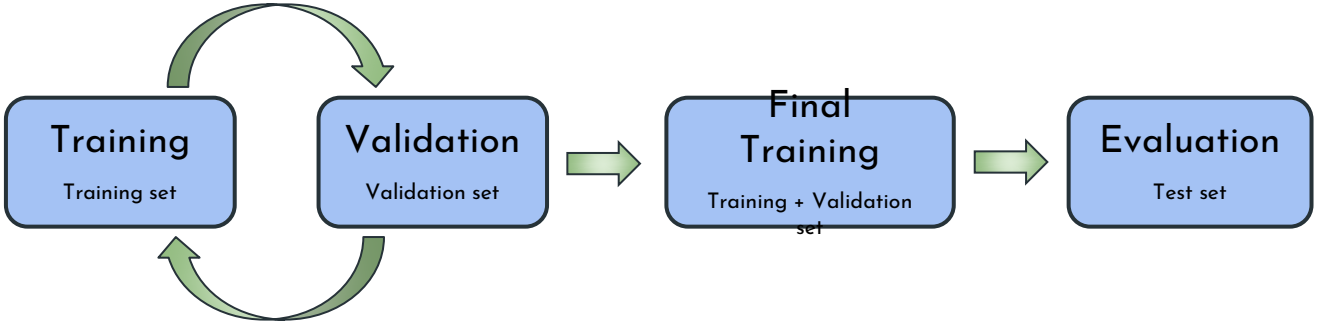
02
Logistic Regression
Statistical classifier

03
LSTM
Gated RNN

04
BERT
Pre-trained NN

05
Ensemble
Weighted voting

Training Process



Evaluation Metrics

- Binary classification task with balanced labels
- Precision and recall both important and should be considered

Accuracy	Precision	Recall	F1 Score	AUC-ROC
Proportion of correct predictions	Proportion of positive predictions that are actual positive instances	Proportion of actual positive instances that are predicted positive	Harmonic mean of precision and recall	Measure of a model's ability to distinguish between classes as its threshold varies

4. Models

Logistic Regression
&
Support Vector Machine

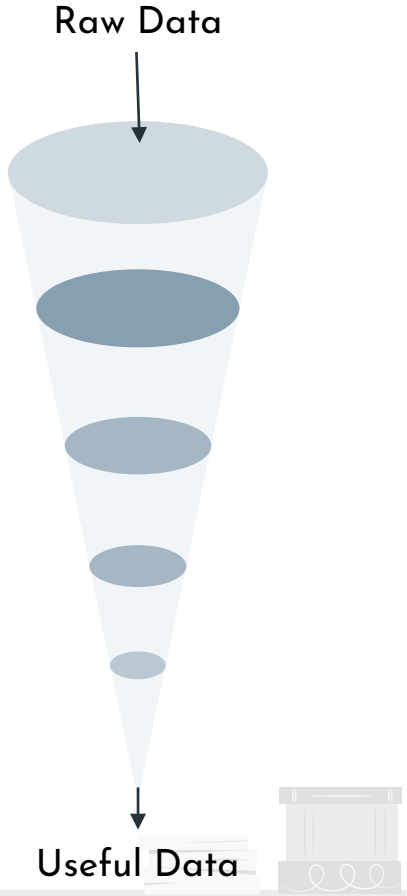
Feature Engineering

Extract most relevant features from the dataset

- Capitalized Frequency
- Punctuation Frequency
- Word Count

Clean raw data

- Remove punctuation
- Change all characters in text to lower-case
- Columns with empty entries filled with empty strings



Implementation - LR & SVM

Two approaches to each model:

Approach 1: Use sarcastic comment_processed only -> **Base** model

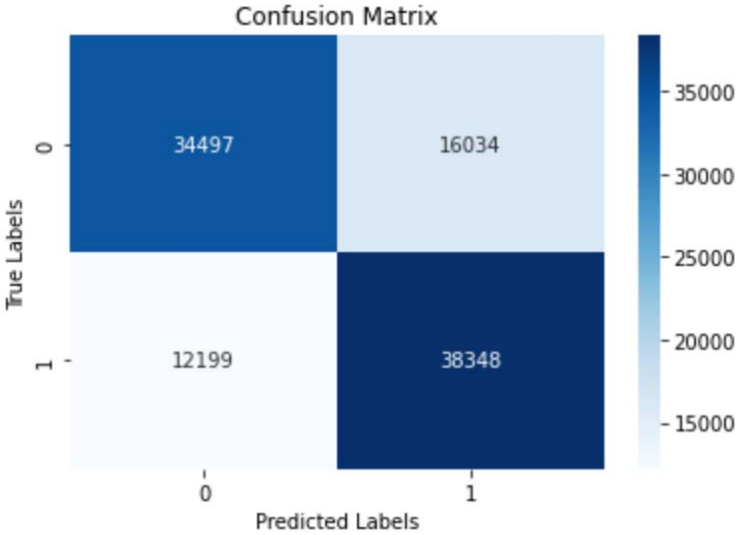
Approach 2: Use numerical features extracted earlier, such as punctuation frequency etc.

Methods used:

- Text Representation using TFIDF-Vectorizer (Term Frequency Inverse Document Frequency)
- Lemmatization of text - did not remove stop words
- Usage of pipeline

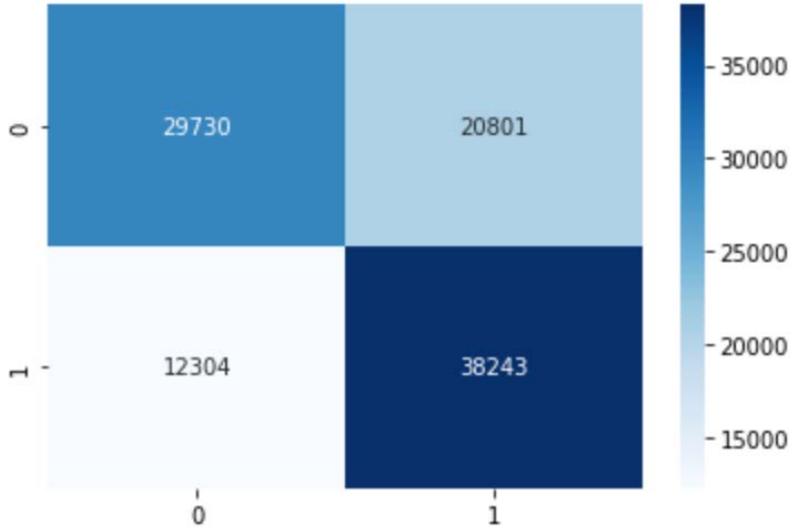
Validation - LR

Using Approach 1:



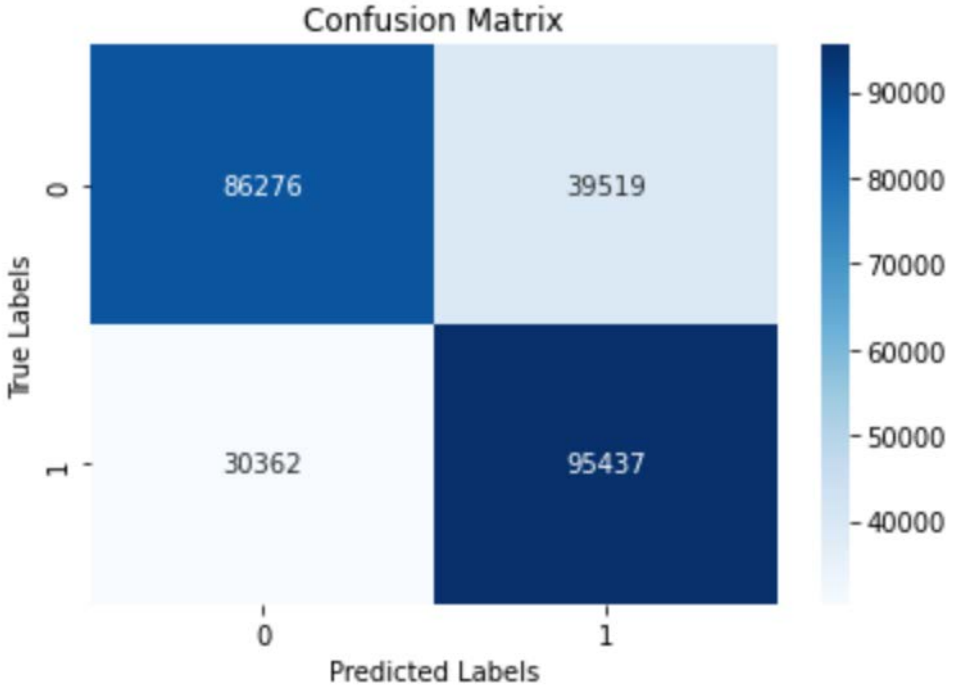
Accuracy	Precision	Recall	F1
0.720	0.738	0.682	0.709

Using Approach 2:



Accuracy	Precision	Recall	F1
0.672	0.707	0.588	0.642

Test Evaluation - LR

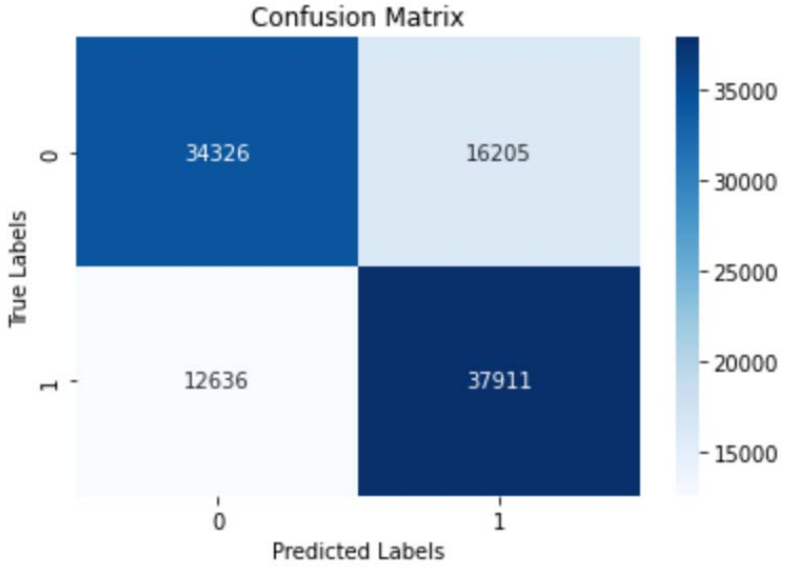


Using Approach 1:

Accuracy	Precision	Recall	F1
0.722	0.739	0.685	0.711

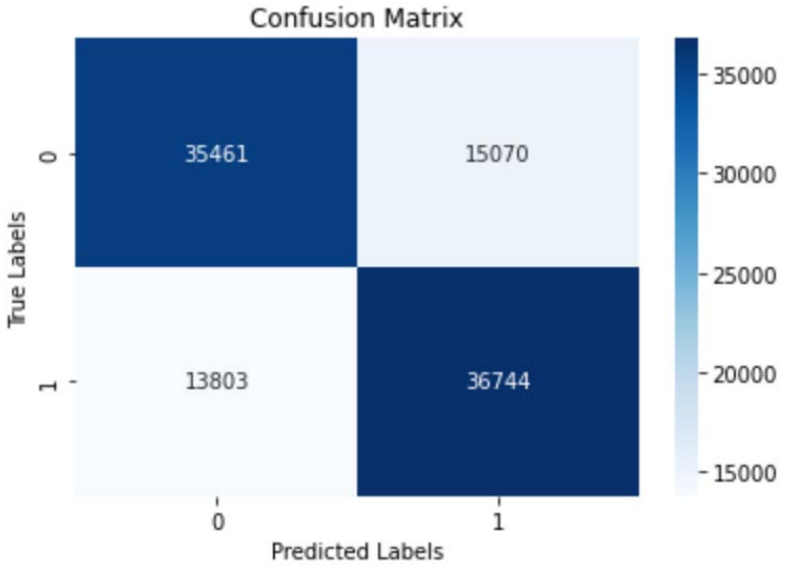
Validation - SVM

Using Approach 1:



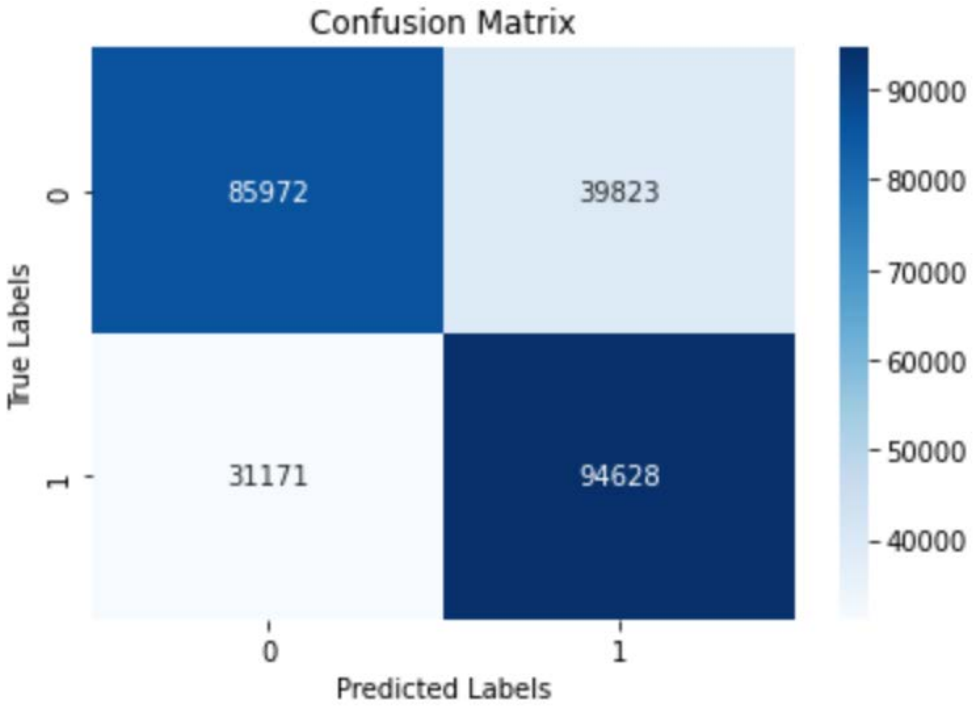
Accuracy	Precision	Recall	F1
0.715	0.731	0.679	0.704

Using Approach 2:



Accuracy	Precision	Recall	F1
0.714	0.720	0.702	0.711

Test Evaluation - SVM



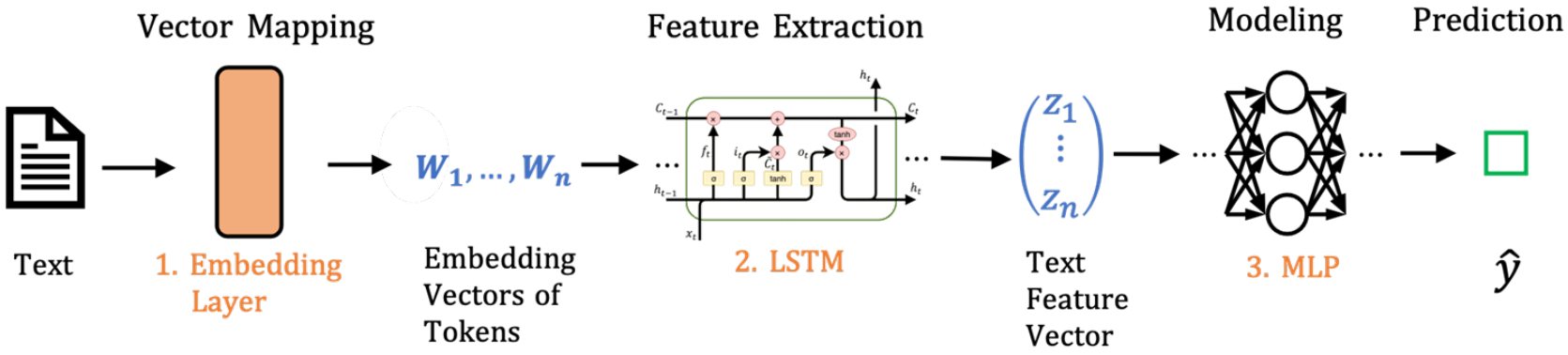
Using Approach 1:

Accuracy	Precision	Recall	F1
0.718	0.734	0.683	0.708

LSTM

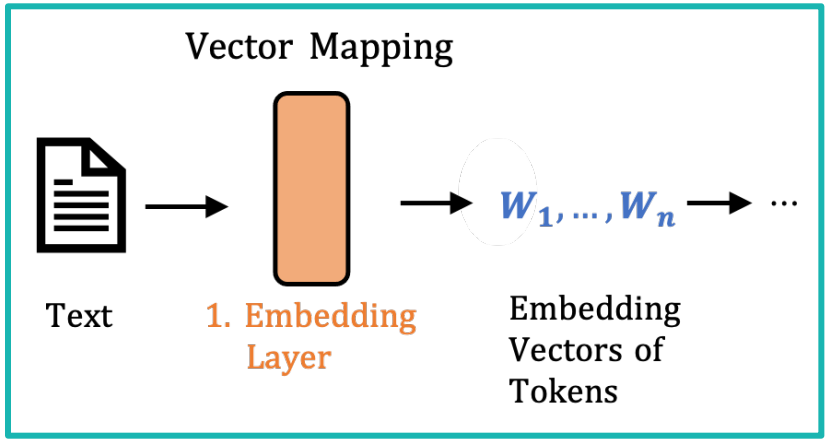
Long Short Term Memory

Architecture



Credits: Image adapted from CS3244 slides

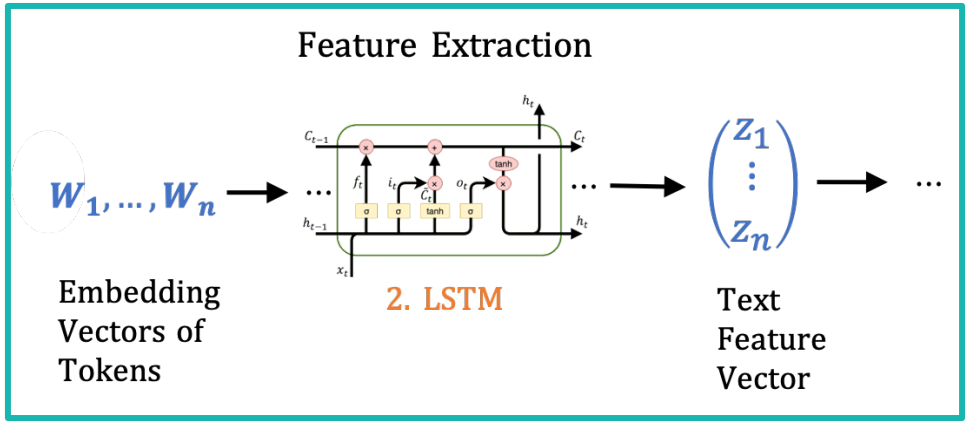
Architecture: Embedding



- Mapping from text tokens into low dimensional vector space
- Vectors can then be used as input to downstream layers
- Randomly initialised and trained with the rest of the network through back propagation

Credits: Image adapted from CS3244 slides

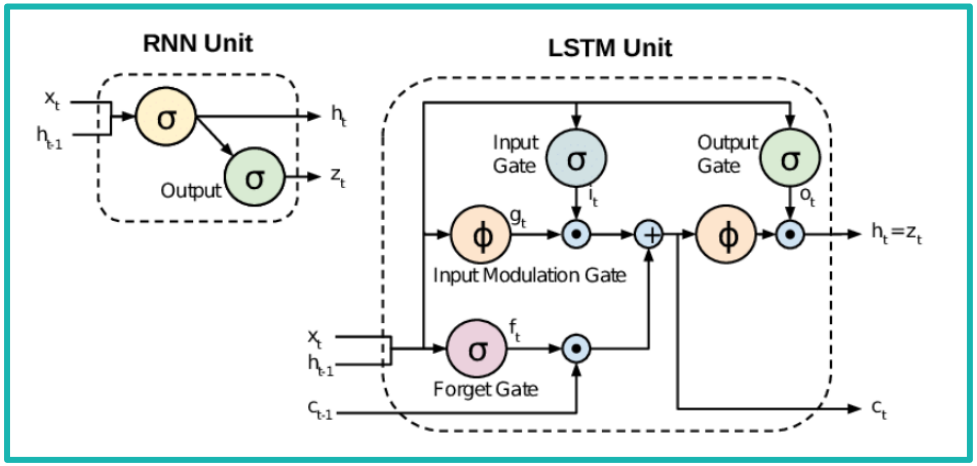
Architecture: LSTM



- Specialised RNN architecture
- Capture subtle patterns in sequences
- Allow conditioning on the entire history without making Markov assumptions
- Consider infinite windows of input while outputting a fixed-sized vector
- In text, word order and context is important
- Feature extractor
- Many-to-One RNN

Credits: Image adapted from CS3244 slides

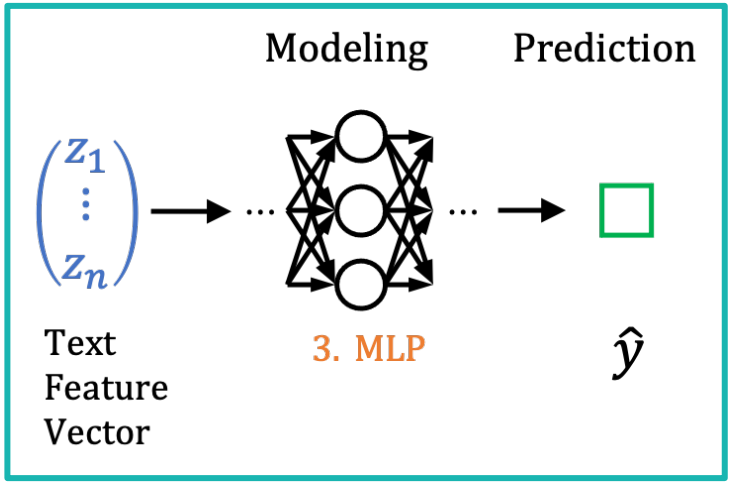
Architecture: LSTM



- Designed to solve the vanishing gradient problem
- Differentiable gating mechanisms to decide what to remember or forget
- Smooth mathematical functions that simulate logical gates
- Able to capture long-term dependencies between word sequences

Credits: Image from [Data Science Duniya]

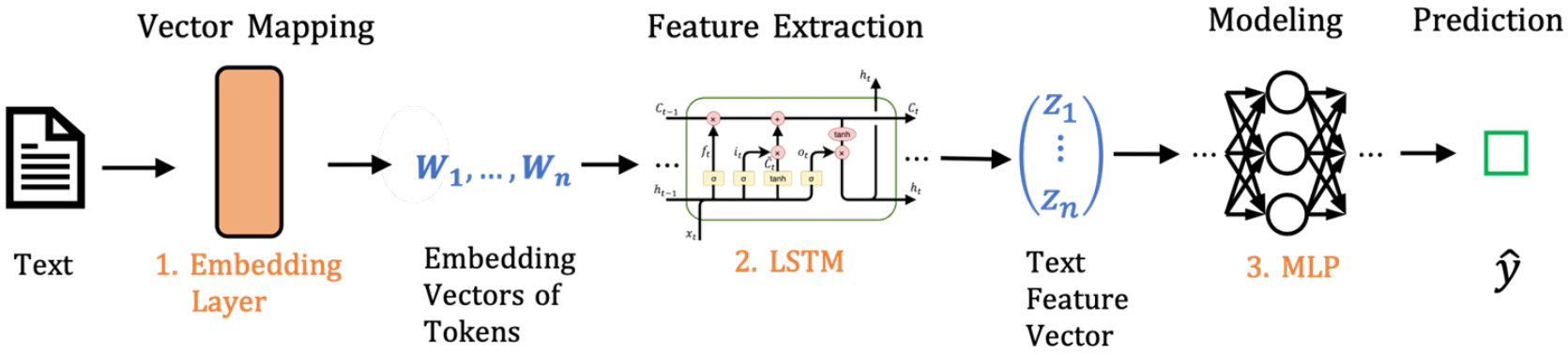
Architecture: MLP



- Uses feature vector of the text input from LSTM to make predictions
- Sigmoid layer with one output to represent the probability of the input being sarcastic
- Trained with binary cross entropy loss based on the actual sarcasm labels

Credits: Image adapted from CS3244 slides

Architecture

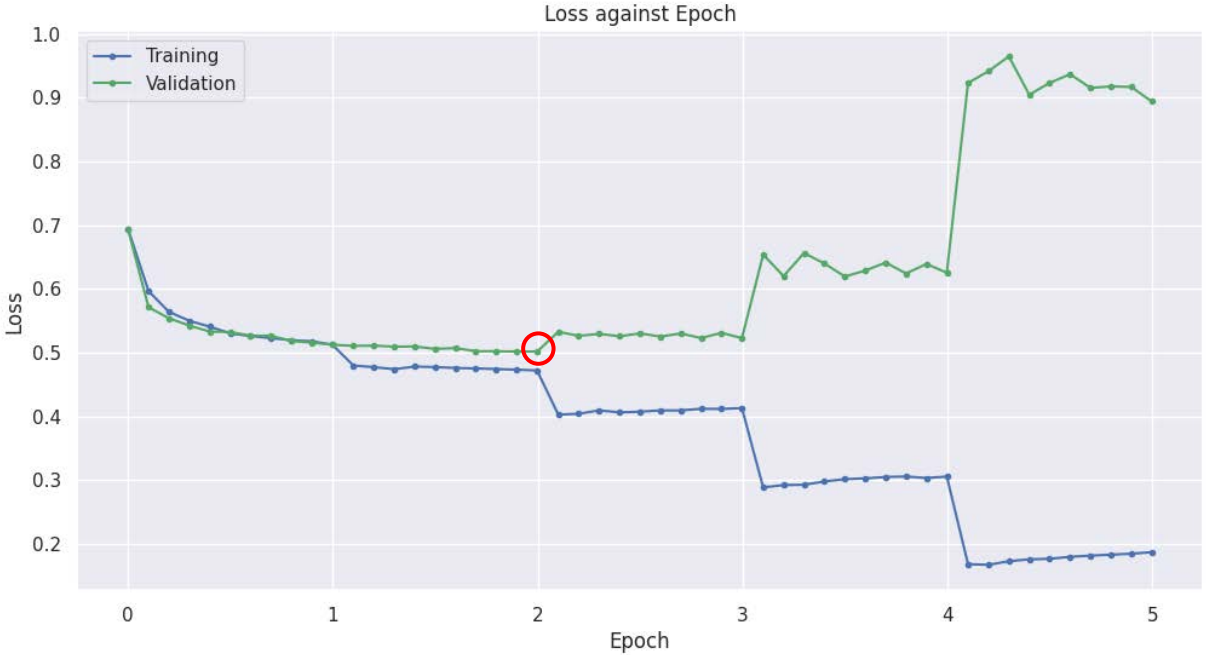


Credits: Image adapted from CS3244 slides

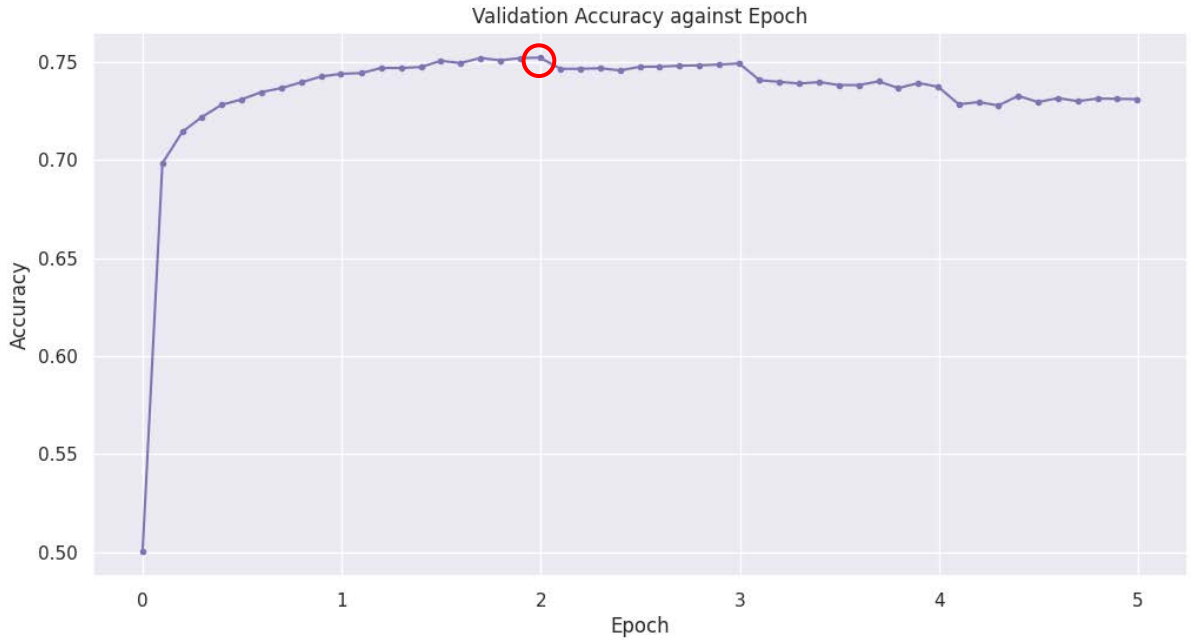
Implementation

- Built using **PyTorch**
- Hyperparameters
  - Number of epochs: **2**
  - Batch size: **512**
  - Learning rate: **0.001**
  - Optimizer: **Adam**
  - Embedding vector dimension: **300**
  - MLP dropout regularisation: **0.30** probability
  - MLP activation function: **ReLU**
  - MLP hidden layers: **1**
  - Decision Threshold: **0.50** (chosen to maximise TPR-FPR)

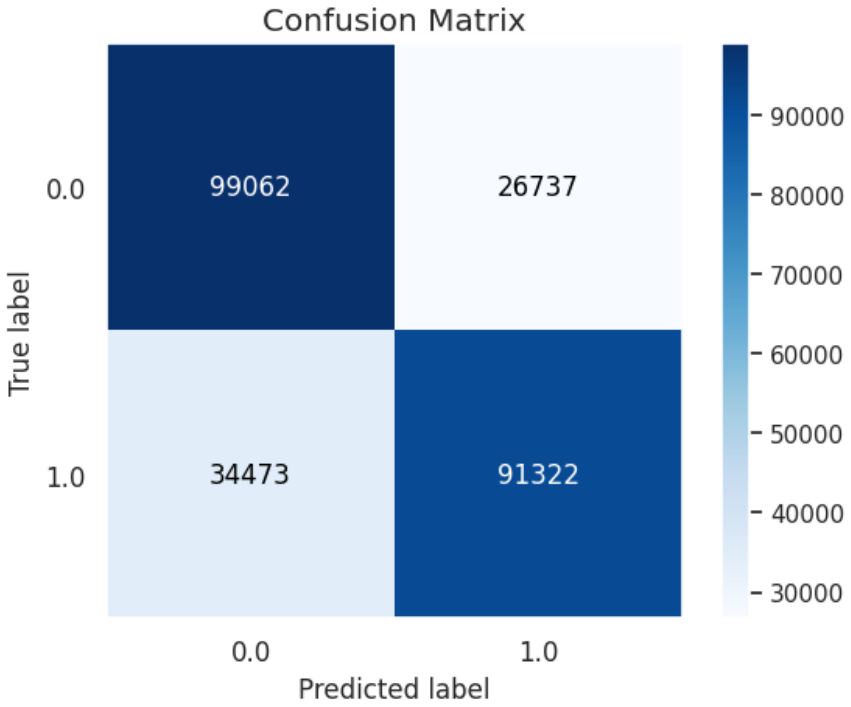
Validation


Loss against Epoch
Training
Validation
Loss
Epoch
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
1
2
3
4
5

Validation

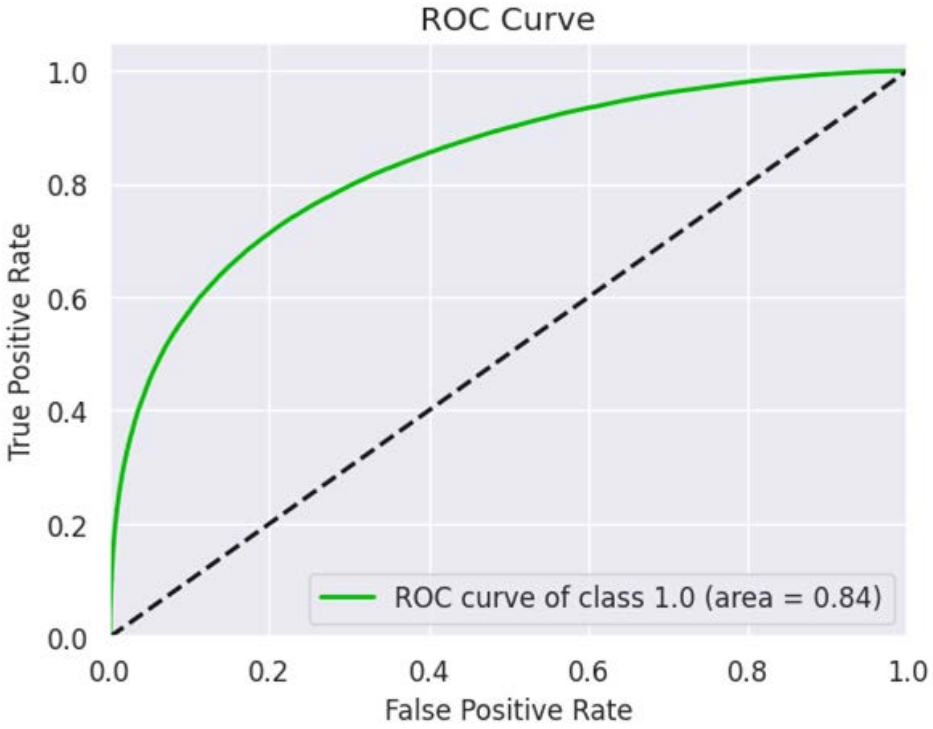

Validation Accuracy against Epoch
Accuracy
0.75
0.70
0.65
0.60
0.55
0.50
0
1
2
3
4
5
Epoch

Evaluation



Accuracy	Precision	Recall	F1
0.757	0.774	0.726	0.749

Evaluation

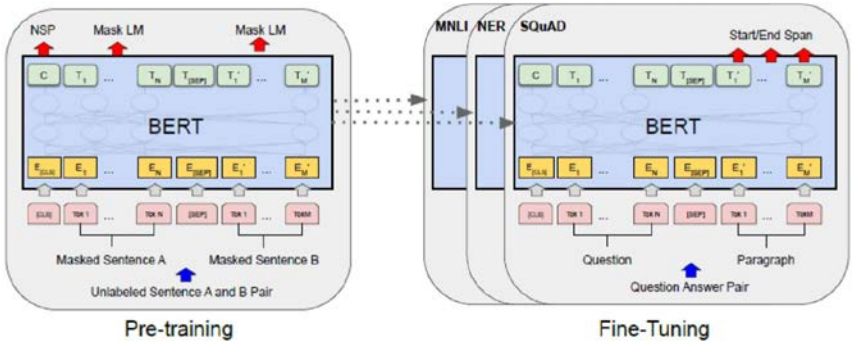


AUC Score
0.836

BERT

Bidirectional **E**ncoder **R**epresentations from **T**ransformers

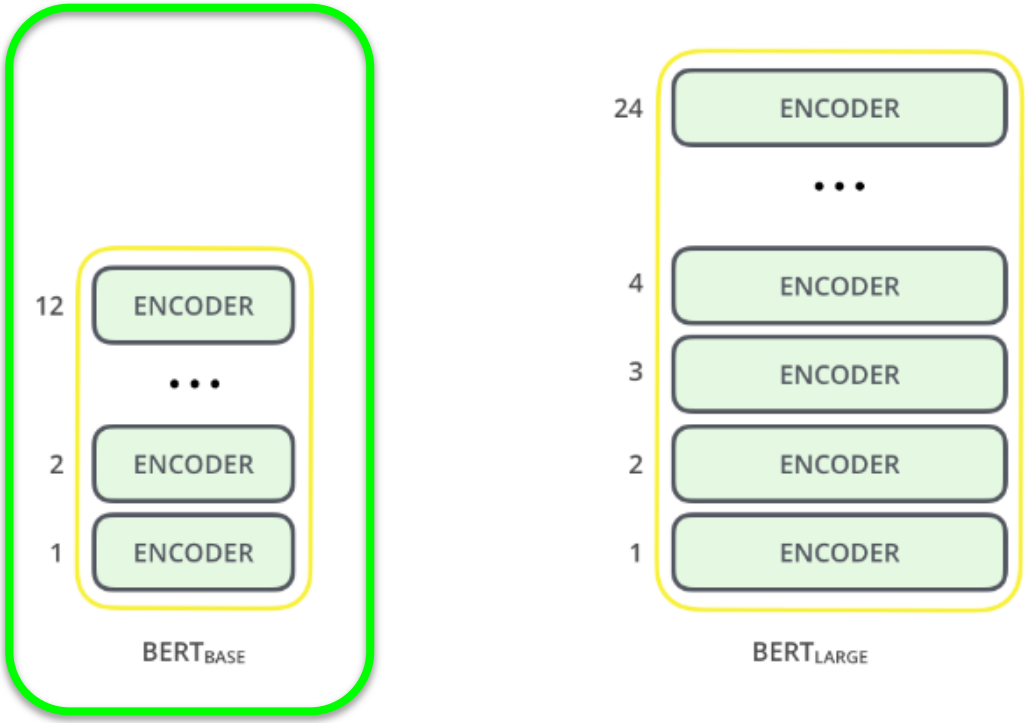
BERT



- State of the art NLP model published by Google in 2018
- Applies bidirectional training of the Transformer model to perform language modelling

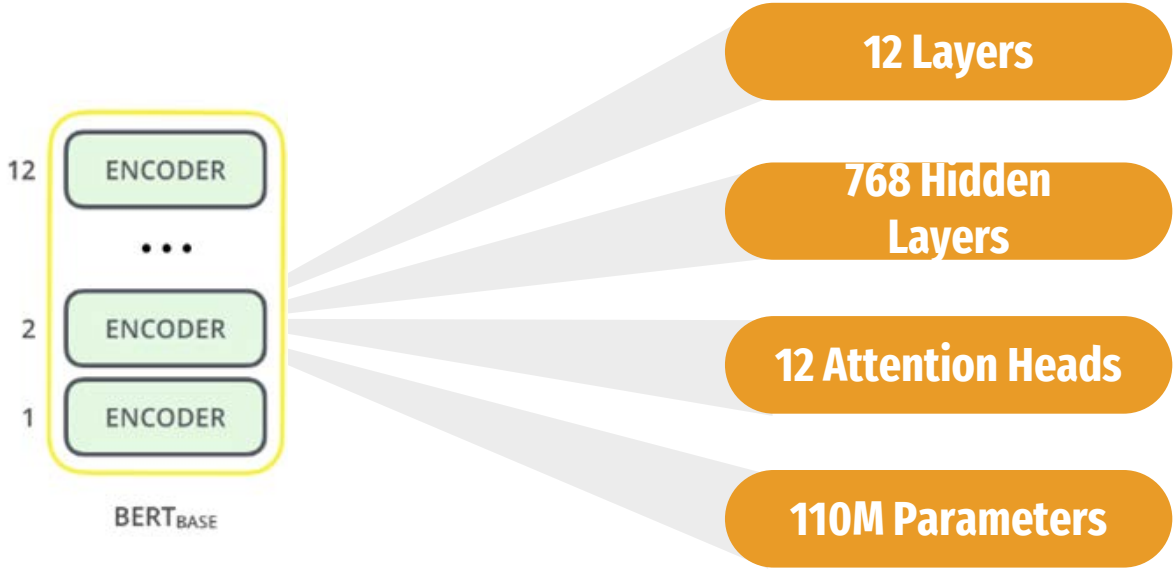
Credits: Image from the BERT paper, NAACL 2019 "BERT: Pre-training of deep bidirectional transformers for language understanding"

Architecture



Credits: Image from https://jalammar.github.io/illustrated-bert/

BERT base Architecture



Credits: Image from https://jalammar.github.io/illustrated-bert/

Transfer Learning

01

BERT is pre-trained on **Masked Language Modelling** and **Next Sentence Prediction** over a massive unlabelled plain text corpus comprising *BooksCorpus* and *English Wikipedia*

02

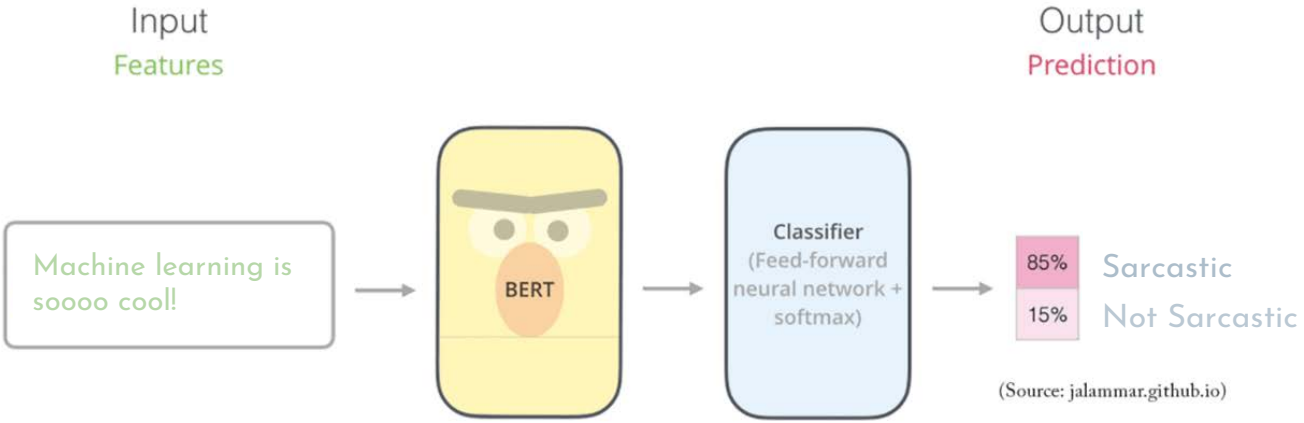
Base-layer of knowledge accumulated and stored from pre-training can be applied to solve other different yet related NLP problems

Why transfer learning?

Better starting point and improved baseline performance

Speeds up training process

Fine-tuning and training



Pre-trained models like BERT can be easily fine-tuned to perform various NLP tasks including **sentiment analysis.** We fine-tuned BERT on the training and validation datasets of labelled reddit comments **specifically** for our sarcasm classification task.

Credits: Image from https://jalammar.github.io/illustrated-bert/

Implementation

- Built using **PyTorch-transformers**
- Model: **BERT base uncased**
- Hyperparameters
  - Training batch size: **32**
  - Learning rate: **0.000005**
  - Number of epochs: **2**
  - Optimizer: **AdamW**
  - Feed-forward neural network
    - **1** Hidden layer of size: **50**
    - Dropout layer for regularisation: **0.30** probability
    - Activation function: **ReLU**
  - Decision Threshold: **0.535** (chosen to maximise TPR-FPR)

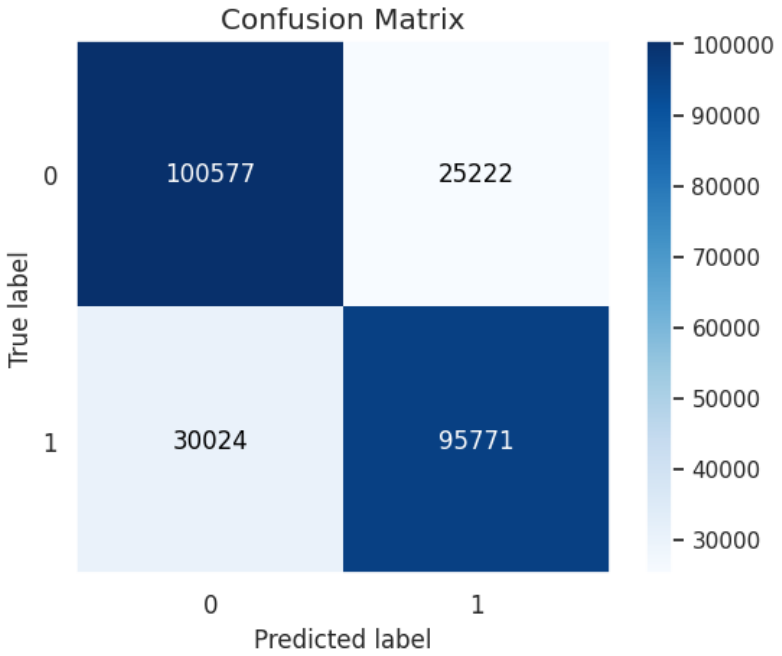
BERT's output for the **[CLS]** token of is fed into an additional feed-forward neural network to classify each sentence as sarcastic or not sarcastic.

=> Improved accuracy by ~2%

Validation

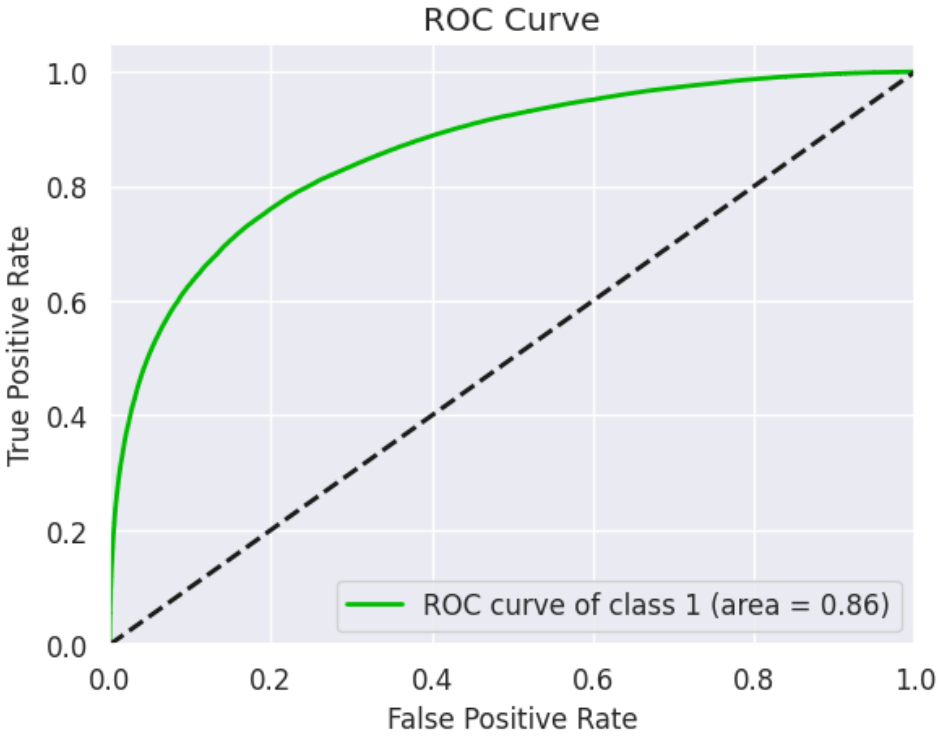
Model	Num of Epoch	Learning Rate	Batch Size	
BERT base uncased	1	2e-5	16	precision recall f1-score support 1 0.7692 0.7263 0.7472 50531 0 0.7409 0.7822 0.7610 50547 accuracy 0.7543 101078 macro avg 0.7551 0.7543 0.7541 101078 weighted avg 0.7550 0.7543 0.7541 101078
BERT base uncased	1	2e-4	16	precision recall f1-score support 1 0.0000 0.0000 0.0000 50531 0 0.5001 1.0000 0.6667 50547 accuracy 0.5001 101078 macro avg 0.2500 0.5000 0.3334 101078 weighted avg 0.2501 0.5001 0.3334 101078
BERT base uncased	1	2e-5	8	Classification Report: precision recall f1-score support 1 0.7500 0.6889 0.7317 50531 0 0.7215 0.8059 0.7613 50547 accuracy 0.7474 101078 macro avg 0.7508 0.7474 0.7465 101078 weighted avg 0.7508 0.7474 0.7465 101078
BERT base uncased	1	2e-5	32	precision recall f1-score support 1 0.7808 0.7301 0.7546 50531 0 0.7467 0.7951 0.7701 50547 accuracy 0.7626 101078 macro avg 0.7637 0.7626 0.7624 101078 weighted avg 0.7637 0.7626 0.7624 101078

Evaluation



Accuracy	Precision	Recall	F1
0.783	0.781	0.780	0.780

Evaluation



AUC Score
0.862

Ensemble

Ensemble

- Weighted voting of the 4 models
- BERT (most performant model) has 2 votes, rest of the models have 1 vote each

Accuracy	Precision	Recall	F1
0.776	0.795	0.744	0.769

5. Analysis

Evaluation Summary

	Accuracy	Precision	Recall	F1 Score	AUC-ROC
SVM	0.718	0.734	0.683	0.708	-
Logistic Reg	0.722	0.739	0.685	0.711	0.722
LSTM	0.757	0.774	0.726	0.749	0.836
BERT	0.783	0.781	0.780	0.780	0.862
Ensemble	0.776	0.795	0.744	0.769	-

SVM ≈ Logistic Reg < LSTM < Ensemble ≈ BERT

Model Comparison

LSTM and BERT perform better than SVM and logistic regression in all metrics

- Ability to learn non-linear target function
- Number of parameters in models
- Examples where LSTM and BERT predicted correctly but SVM and LR did not :
  - " thank you NY for giving us Trump "
  - " As an american, I can proudly say the rest of the world doesn't matter "

Model Comparison

BERT achieves the best performance overall

- Advantage of transfer learning
- Base-layer of knowledge from pre-training
- Examples where BERT predicted correctly but LSTM, SVM and LR did not :
    - “ Wasn't aware that gold had a lowly 13Bn market cap. ”
    - “ Funny original meme haha :). ”
    - “ I see you understood my point perfectly. ”

Model Comparison

Ensemble scores higher than BERT for precision

- Voting system outputs positive only if majority of the models are in agreement
- Using the F0.5 metric, the ensemble scores slightly higher than BERT (0.784 vs 0.780)
- Diverse hypotheses of the individual models may decrease variance
- Consider choosing ensemble over BERT if precision is a priority for some practical use

Prior Work Comparison

- **SVM** and **Logistic Regression** performed slightly worse than baseline methods in terms of **accuracy**
- **LSTM**, **BERT** and **Ensemble** models performed better
- Overall, our models all performed worse than average humans

Method	all-bal*
Bag-of-Words	73.2
Bag-of-Bigrams	75.8
Sentence Embedding	71.0
Human (Average)	81.6
Human (Majority)	92.0
Random	50.0

Table: Accuracy percentage of baseline methods for sarcasm detection

Difficulty of Task

- Sarcasm detection is a difficult task
- Written text is not able to convey sarcasm as well as speech
- More difficult than normal sentiment analysis
- Insufficient to use the positive/negative degree of words to detect sarcasm
- Difficulty of picking up sarcasm is likely the reason why recall < precision for all models

Difficulty of Task

Excerpt from *The Big Bang Theory*

Leonard: *"Hey, Penny. How's work?"*
Penny: *"Great! I hope I'm a waitress at the Cheesecake Factory for my whole life!"*
Sheldon: *"Was that sarcasm?"*
Penny: *"No."*
Sheldon: *"Was that sarcasm?"*
Penny: *"Yes."*

Penny's words have overall positive sentiment on the surface but is actually sarcastic and has the opposite meaning

Limitation of Data

“ Despite our efforts to filter noisy ‘/s’ labels, there remain instances where no simple rule reliably eliminates incorrect labels. ” *A Large Self-Annotated Corpus for Sarcasm*

- False positives - comment is incorrectly labeled as sarcastic due to presence of ‘/s’ tag
  - ‘/s’ may have other connotations, e.g. ‘<s>…</s>’ code in HTML
  - Example: “ Yeah I know lol it's a joke that's why I put /s ” was wrongly labelled sarcastic
- False negatives - comment is sarcastic, but not annotated with ‘/s’ tag
  - More difficult to detect than false positives
  - User may be unaware of the ‘/s’ convention
  - User may choose not to include the ‘/s’ tag

6. Conclusion

Conclusion

- Further analysis of data features
  - Subreddit may affect whether the comment is sarcastic or not
- Applications of this model in real world scenarios
  - Notify people if a comment seems to be highly sarcastic
  - Sarcastic text generator

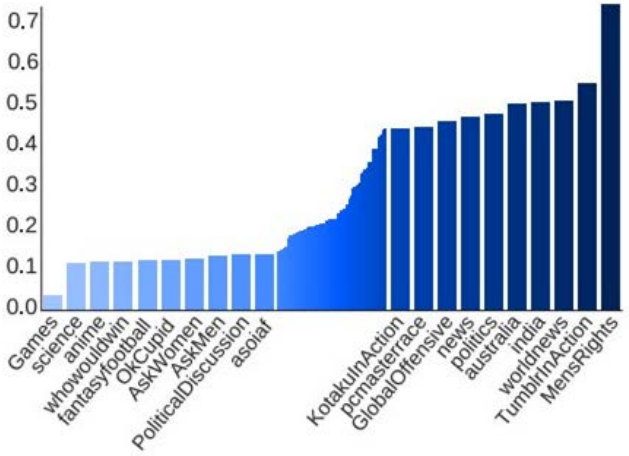


Figure: Sarcasm percentage for subreddits with more than a million comments in dataset

THANK YOU

CREDITS: This presentation template was created by Slidesgo, including icons by Flaticon, and infographics & images by Freepik