Can We Detect Truthfulness?
Exploring Machine Learning Approaches

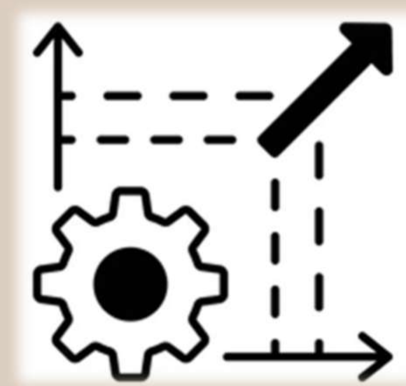
Content

Business Case

- **Widespread Access to Information**
  - Digital media has made information highly accessible, but this ease also enables the rapid spread of misinformation and fake news

- **Impact of Fake News**
  - Fake news can influence public opinion, elections, financial markets, and even incite violence due to the lack of fact-checking on online platforms

- **Faster Spread Than Truth**
  - Difficult to control once released



- **Need for Scalable Solutions**
  - Digital media becomes the main news source for many

- **Machine Learning as a Solution**
  - Offers real-time, scalable detection of fake news by analyzing large volumes of content quickly using Natural Language Processing (NLP)

Our Objective

To develop a model to assess the truthfulness of a sentence, helping individuals, government bodies, news outlets to make informed decision about whether a piece of text is likely true or false. Recognizing that real world data is often nuanced and not black or white, this study will classify truthfulness on a spectrum – ranging from true, mostly true, half true, barely true and false rather than treating it as a binary prediction.

To explore the different machine learning techniques, we designed a multi-class classification task to assess its ability in detecting statements that are contextually similar but subtly different.

Motivation

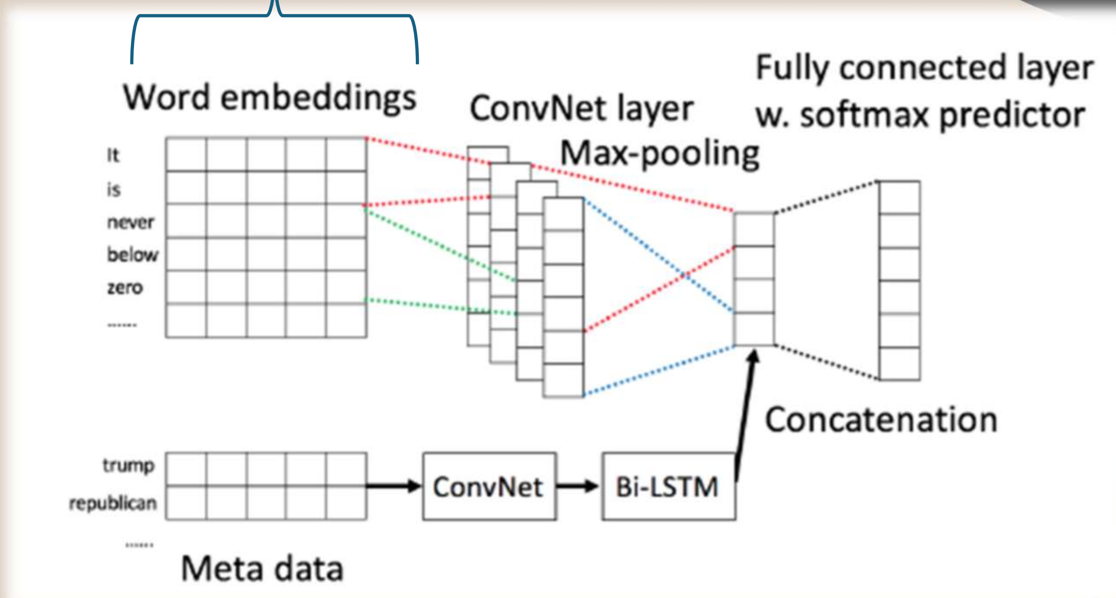
**“Liar, Liar Pants on Fire”:
A New Benchmark Dataset for Fake News Detection**

William Yang Wang
Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106 USA
william@cs.ucsb.edu

Models	Valid.	Test
Majority	0.204	0.208
SVMs	0.258	0.255
Logistic Regress0ion	0.257	0.247
Bi-LSTMs	0.223	0.233
CNNs	0.260	0.270
Hybrid CNNs		
Text + Subject	0.263	0.235
Text + Speaker	**0.277**	0.248
Text + Job	0.270	0.258
Text + State	0.246	0.256
Text + Party	0.259	0.248
Text + Context	0.251	0.243
Text + History	0.246	0.241
Text + All	0.247	**0.274**

The evaluation results on the LIAR dataset. The top section: text-only models. The bottom: text + meta-data hybrid models.

Pre-trained 300-Dim word2vec embeddings from Google News



The proposed hybrid Convolutional Neural Networks framework for integrating text and meta-data.

Dataset

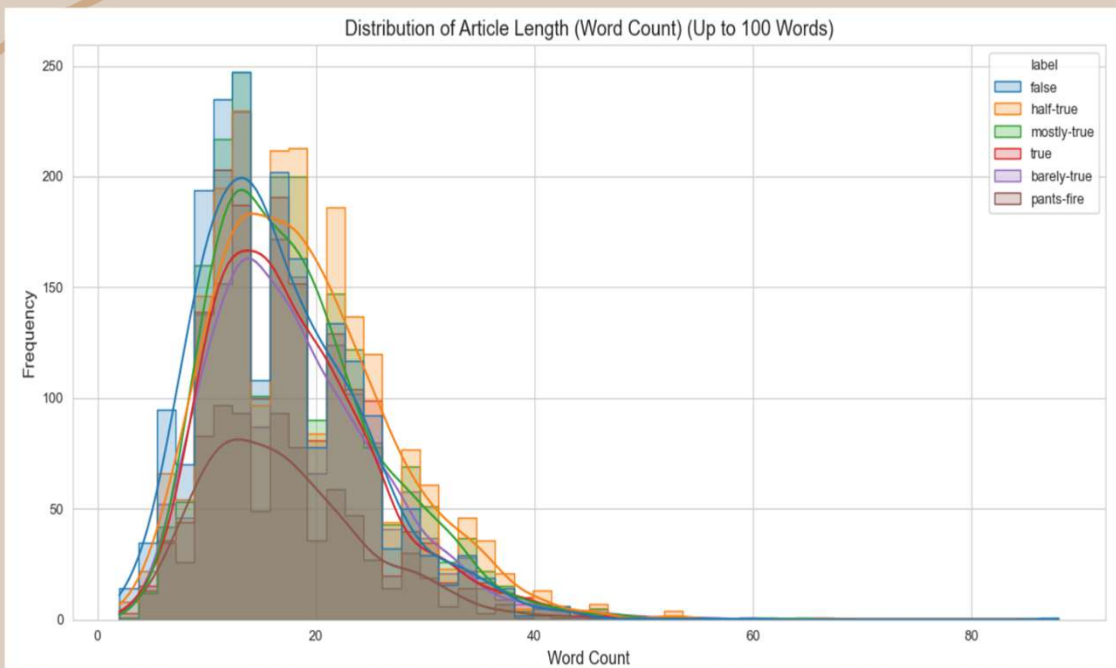
- "Liar, Liar Pants on Fire" - *A New Benchmark Dataset for Fake News Detection (William Yang Wang, 2017)*
- 12,808 rows of manually labelled short statements in various context from POLITIFACT.com
- With 9 columns
  - Id, Label, Text, Category, Speaker, Speaker_Title, State, Party_Affiliation, Total_Credit_Counts
- Labelled into 6 categories of truthfulness
  - pants-fire, false, barely-true, half-true, mostly-true, and true

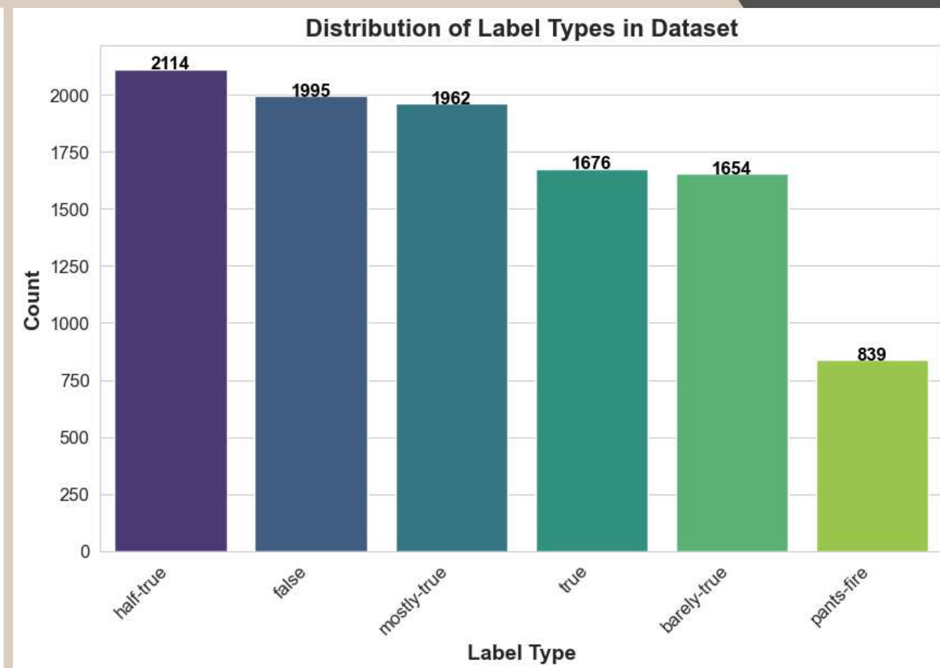
- Other dataset considered were mainly for binary classifications (Fake or real [6k], Combined corpus [80k])

Statement: *"The last quarter, it was just announced, our gross domestic product was below zero. Who ever heard of this? Its never below zero."*
Speaker: Donald Trump
Context: presidential announcement speech
Label: Pants on Fire
Justification: According to Bureau of Economic Analysis and National Bureau of Economic Research, the growth in the gross domestic product has been below zero 42 times over 68 years. Thats a lot more than "never." We rate his claim Pants on Fire!

Statement: *"Under the health care law, everybody will have lower rates, better quality care and better access."*
Speaker: Nancy Pelosi
Context: on 'Meet the Press'
Label: False
Justification: Even the study that Pelosi's staff cited as the source of that statement suggested that some people would pay more for health insurance. Analysis at the state level found the same thing. The general understanding of the word "everybody" is every person. The predictions dont back that up. We rule this statement False.

Data Exploration

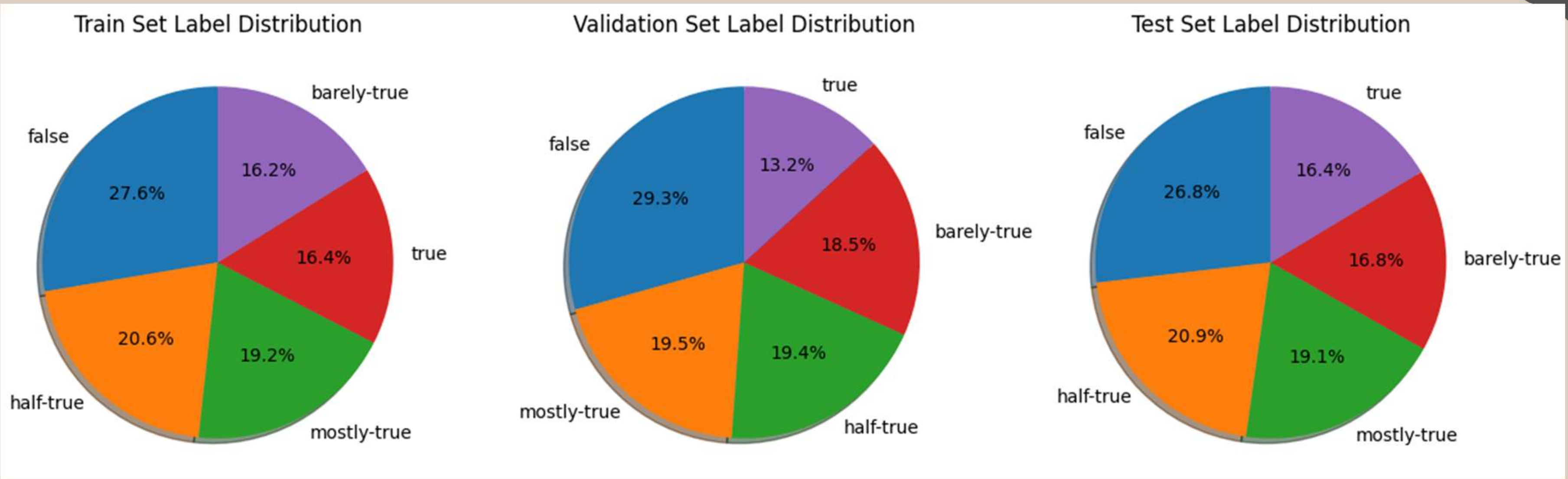




Most statements are fairly short, under 40 words, and this trend is consistent across all labels

Label distribution is relatively balanced

Data Exploration



Distribution of labels across train, validation and test sets are also similar

Why BERT Base Uncased?

BertForSequenceClassification:

- Uses [CLS] representation as input to a dense layer to predict
- Records in dataset are usually short (1–2 sentences)
- [CLS] token does a good job of capturing the meaning of short inputs

Why BERT over DistilBert?

	BERT	DistilBERT
Layers	12	6
Pretraining Task	Next Sentence Prediction, Masked Level Modelling	Masked Level Modelling
Size (million parameters)	110	66

Generalise better

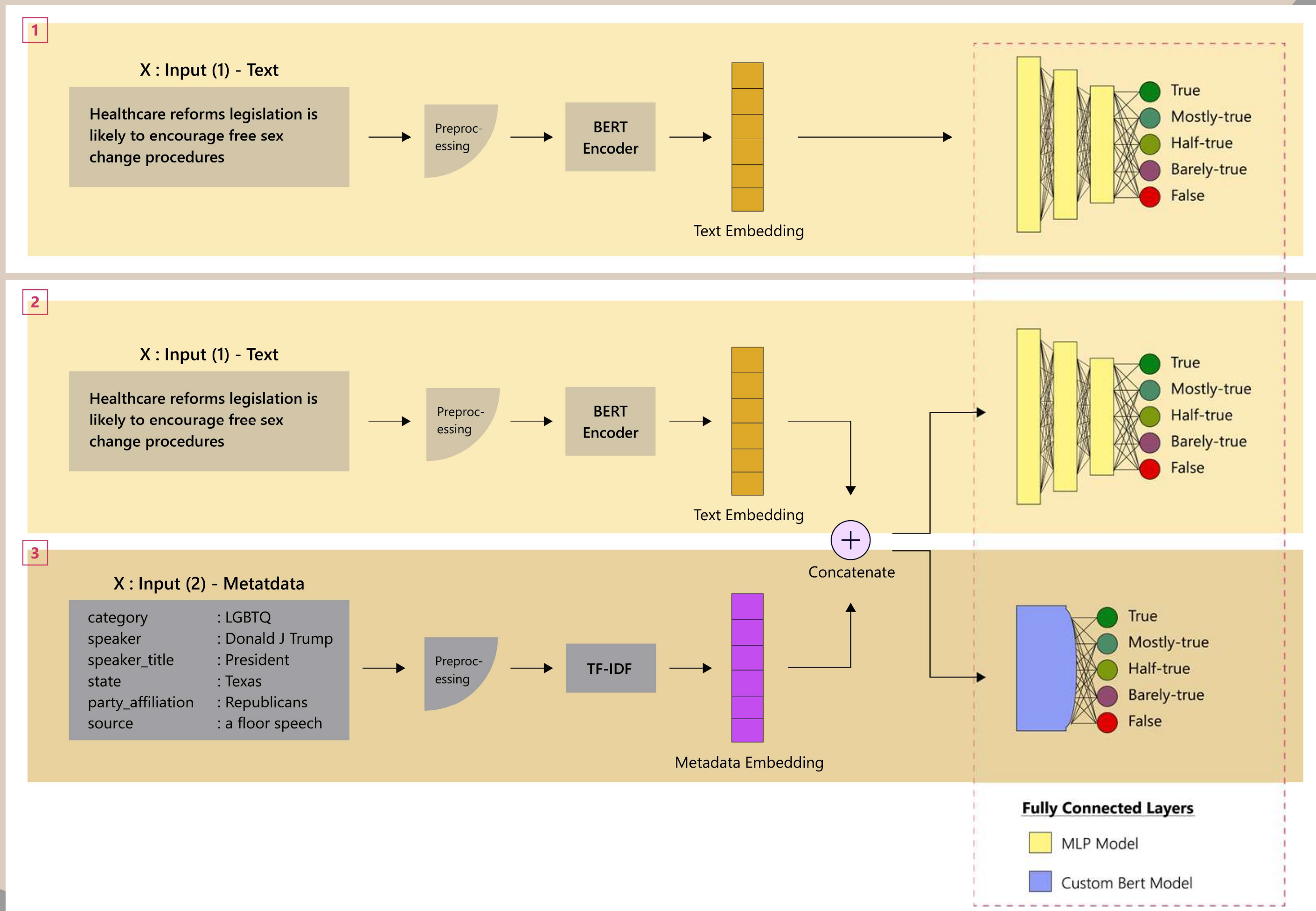
Bert Base Uncased:

- Trained to ignore case
- Dataset includes real-world, noisy text where casing is inconsistent or missing

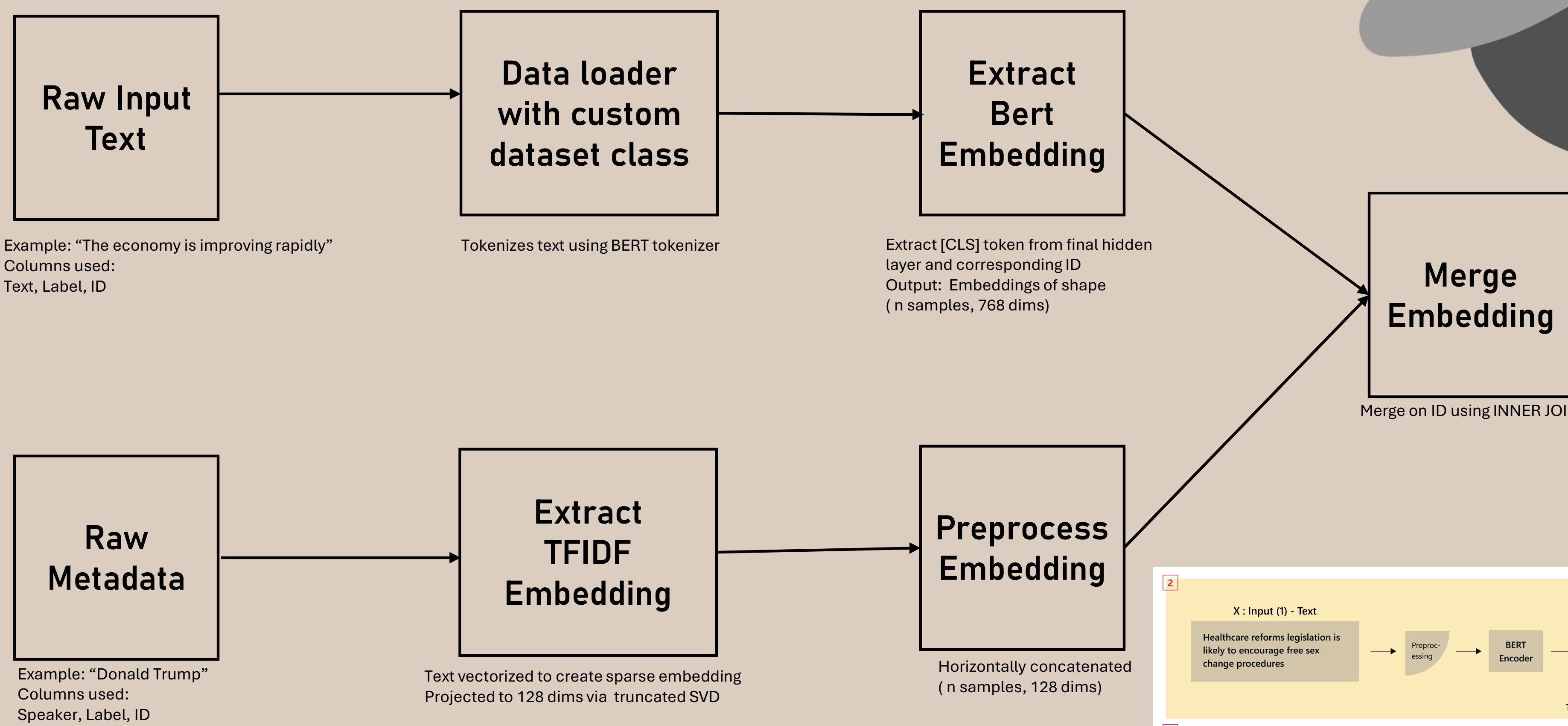
Expectation: “Obama care is…”
Reality: “obama Care is… ”

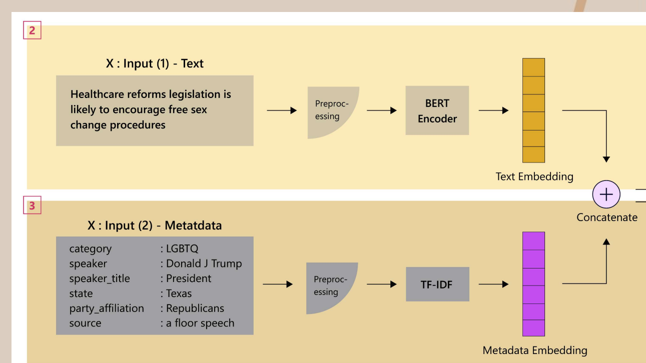
Bert Base Cased relies on proper casing to interpret names and sentence structure

Pipelines


1
X : Input (1) - Text
Healthcare reforms legislation is likely to encourage free sex change procedures
Preproc-essing
BERT Encoder
Text Embedding
True
Mostly-true
Half-true
Barely-true
False
2
X : Input (1) - Text
Healthcare reforms legislation is likely to encourage free sex change procedures
Preproc-essing
BERT Encoder
Text Embedding
Concatenate
True
Mostly-true
Half-true
Barely-true
False
3
X : Input (2) - Metatdata
category : LGBTQ
speaker : Donald J Trump
speaker_title : President
state : Texas
party_affiliation : Republicans
source : a floor speech
Preproc-essing
TF-IDF
Metadata Embedding
True
Mostly-true
Half-true
Barely-true
False
Fully Connected Layers
MLP Model
Custom Bert Model

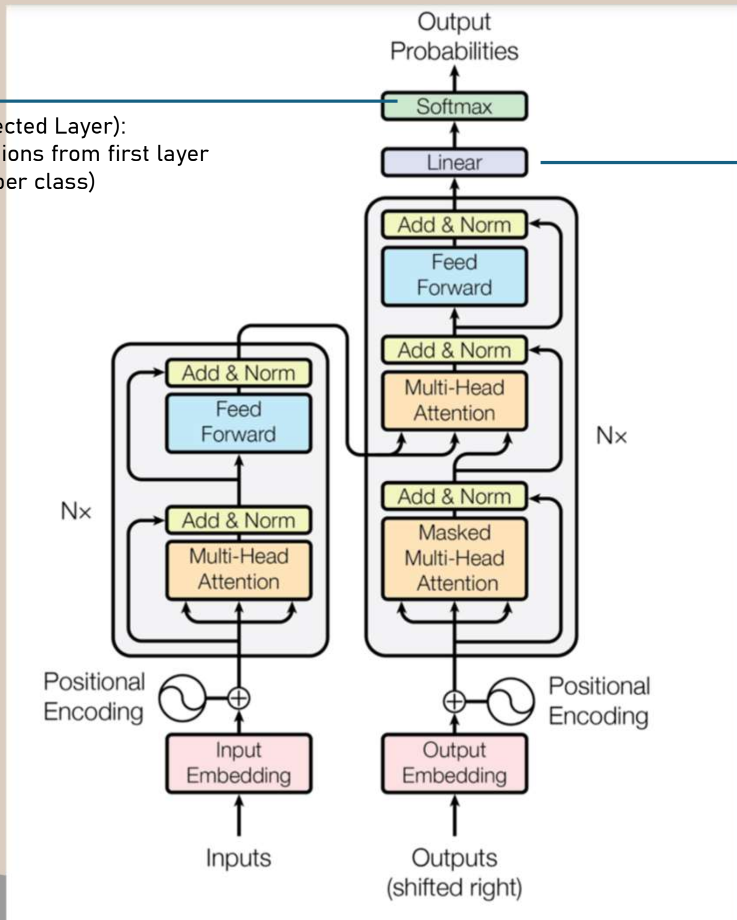
Preprocessing Pipeline





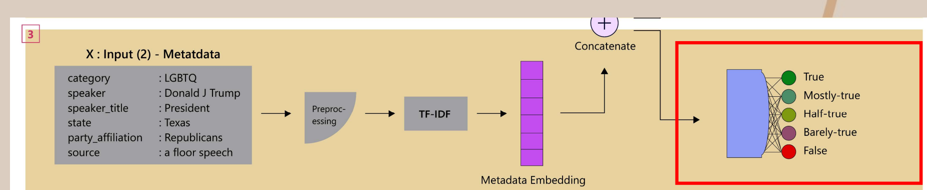
Custom BERT Architectural Design

Replaced with:
- Output layer (Fully Connected Layer):
- Input: hidden representations from first layer
- Output: Final logits (one per class)



Replaced with:
First linear layer (Fully Connected Layer)
- Input: [CLS] Embedding with TFIDF
- Output: Hidden representations (256 dims)
- Dropout to reduce overfitting

Bayesian Optimization used for Hyperparameter tuning

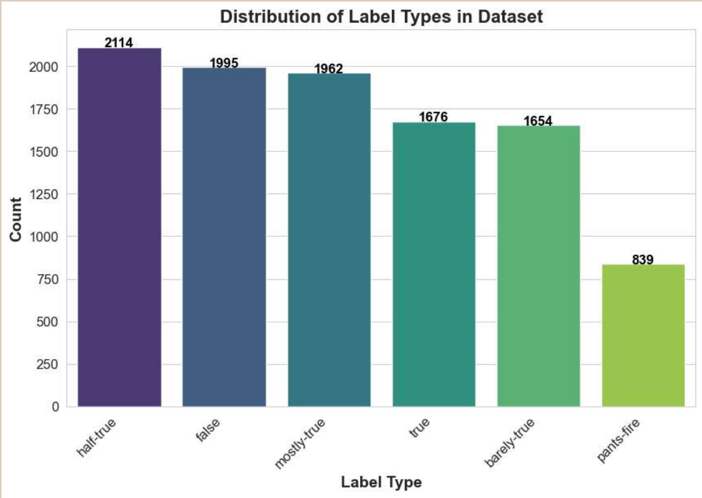


Credit: https://www.analyticsvidhya.com/blog/2021/06/why-and-how-to-use-bert-for-nlp-text-classification/

General Preprocessing

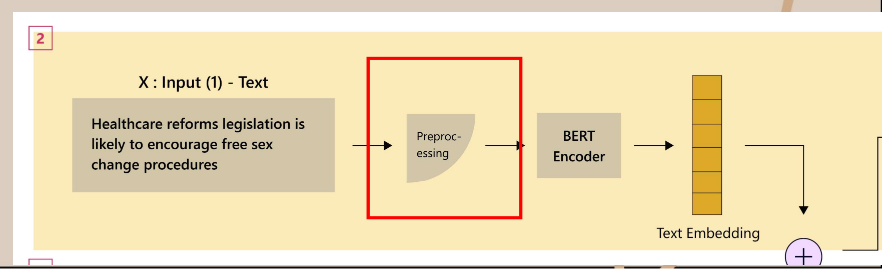
- Dropped ‘total credit counts’ columns
- Combined ‘pants-fire’ and ‘false’ under ‘false’ column
- Empty fields replaced with ‘unknown’
- Removal of duplications in ‘text’ column across train, validation and test datasets

```
Column 1: the ID of the statement ([ID].json).
Column 2: the label.
Column 3: the statement.
Column 4: the subject(s).
Column 5: the speaker.
Column 6: the speaker's job title.
Column 7: the state info.
Column 8: the party affiliation.
Column 9-13: the total credit history count, including the current statement.
9: barely true counts.
10: false counts.
11: half true counts.
12: mostly true counts.
13: pants on fire counts.
Column 14: the context (venue / location of the speech or statement).
```



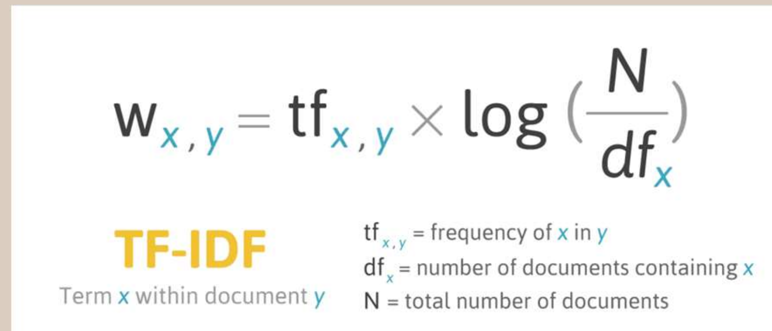
Text Preprocessing

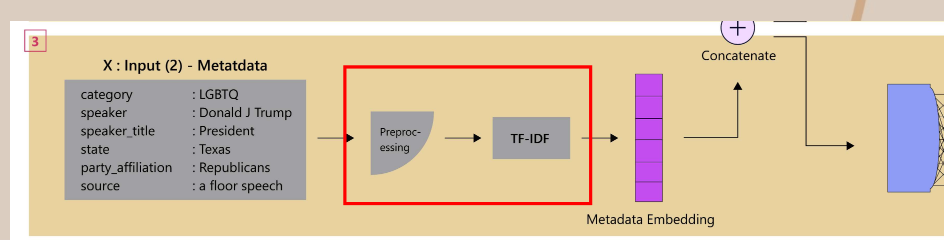
- Text column: Leveraged BERT uncased
- Pretrained on massive corpus of text data
  - Wikipedia – ~2500 million words
  - Google Books – ~800 million words
- Minimal preprocessing:
  - Removal of HTML tags
  - Filtered out special characters
  - Conversion of HTML entities (e.g. &amp → &)
  - Standardisation of quotations (“ ” → ‘ ’)



Metadata Preprocessing

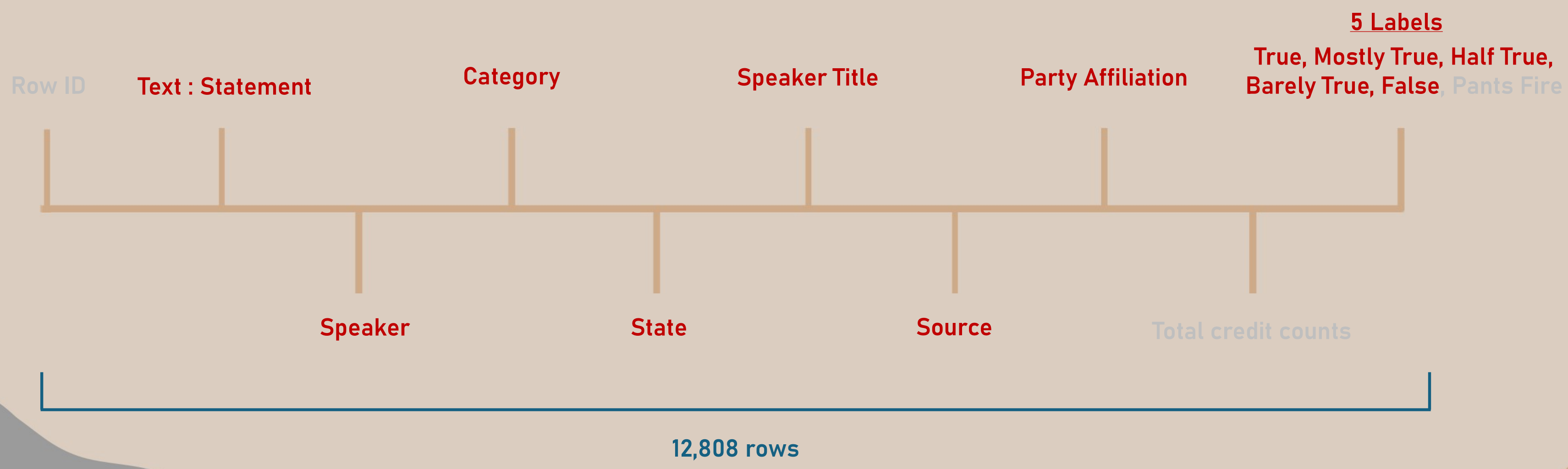
- Metadata Columns: TF-IDF vectorization
- Preprocessing prior to vectorization:
  - Empty fields replaced with ‘unknown’
  - Standardisation of spelling and abbreviations (e.g. U.S.A to USA)
  - Lowercase, lemmatisation and stop word removals
  - Removal of full stops and punctuation marks
  - Sparse to dense representations with truncated SVD



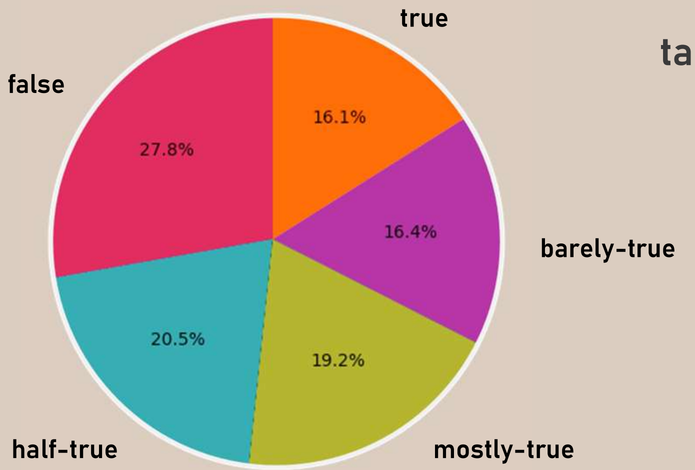


Postprocessed Datasets

- Train dataset – 10,217 rows (80%)
- Validation dataset – 1,263 rows (10%)
- Test dataset – 1,279 rows (10%)



Model Development



Given our Loss Function uses CrossEntropyLoss, class imbalance in target labels can lead to biased predictions

Class weights assigned to emphasize underrepresented classes

$$w_i = \frac{n}{k \cdot n_i}$$

w_i : weight for class i

n : total number of samples in the dataset

k : total number of unique classes

n_i : number of samples in class i

Class weights:

- scaled inversely with class frequency
- normalized by the number of classes
- to reduce bias toward majority classes

Evaluation Metric and Results

How do we choose the best model?

F1 Score

Addresses Trade-offs between

Precision and Recall

RESULTS

- Adding metadata did not improve performance for the MLP models.
  - F1 scores decreased across all MLP models.

Pipeline	Model	Precision	Recall	F1-Score
Text-Only [Baseline]	Text-Only	0.20	0.20	0.16
Text + Metadata [MLP]	Text + Category	0.05	0.20	0.08
	Text + Speaker	0.05	0.20	0.08
	Text + Speaker Title	0.25	0.20	0.10
	Text + Party Affiliation	0.09	0.20	0.12
	Text + State	0.05	0.20	0.08
	Text + Source	0.05	0.20	0.08
	Text + All	0.28	0.20	0.09

RESULTS

- Adding metadata led to a modest improvement in model performance for the custom BERT pipeline.
  - Best performing custom BERT model (pre-class weight implementation) is Text + Speaker Title/Party Affiliation/Source. F1 score increased from 0.16 to 0.18

Pipeline	Model	Precision	Recall	F1-Score
Text-Only [Baseline]	Text-Only	0.20	0.20	0.16
Text + Metadata [Custom Bert]	Text + Category	0.12	0.26	0.16
	Text + Speaker	0.20	0.25	0.16
	Text + Speaker Title	0.24	0.25	0.18
	Text + Party Affiliation	0.19	0.25	0.18
	Text + State	0.16	0.22	0.14
	Text + Source	0.23	0.23	0.18
	Text + All	0.15	0.25	0.17

RESULTS

- Class weight implementation further boosted the performance for certain Custom BERT models.
  - F1 scores for Text + Party Affiliation/Source increased from 0.18 to 0.22/0.20.

Pipeline	Model	Precision	Recall	F1-Score
Text + Metadata [Custom Bert]	Text + Category	0.12 0.26	0.26 0.21	0.16 0.12
	Text + Speaker	0.20 0.16	0.25 0.25	0.16 0.19
	Text + Speaker Title	0.24 0.23	0.25 0.24	0.18 0.17
	Text + Party Affiliation	0.19 0.29	0.25 0.25	0.18 0.22
	Text + State	0.16 0.23	0.22 0.23	0.14 0.17
	Text + Source	0.23 0.22	0.23 0.24	0.18 0.20
	Text + All	0.15 0.21	0.25 0.22	0.17 0.17

Analysis of results across pipelines

- Metadata may enhance model performance depending on the model architecture.
- Custom BERT pipeline models performed better perhaps due to:
  - BERT bidirectional self-attention mechanism – better capture contextual meaning of each word.
  - Pretrained BERT was also trained on a large corpus – deeper understanding of language.
- MLP models suffer performance wise perhaps due to:
  - Fixed non-linearity of MLP – unable to capture nuances of contextual embeddings.

Analysis of results pre/post class weights

- Class weights implementation helped to predict under-represented classes.

Pre-Class Weights

	Precision	Recall	F1-Score
False	0.34	0.82	0.48
Barely True	0.00	0.00	0.00
Half True	0.25	0.12	0.16
Mostly True	0.28	0.35	0.31
True	0.00	0.00	0.00

Post-Class Weights

	Precision	Recall	F1-Score
False	0.36	0.75	0.49
Barely True	0.22	0.21	0.21
Half True	0.24	0.04	0.06
Mostly True	0.26	0.16	0.20
True	0.28	0.15	0.19

Key Limitations

Classification complexity

Ambiguous language blurs class boundaries, leading to overlap

Hard to learn features that clearly separate categories

Class weights

Trade off between majority and minority misclassifications

Dataset

Reliance on a single fact-checking source (PolitiFact)

Annotations may reflect subjective judgments, introducing bias

Future Work

Enhanced Feature Selection

Experiment with different combination of metadata

Concise Classification Approach

Merge ambiguous labels into distinct, broader categories

Advanced Class Balancing

E.g. custom loss function can be implemented to penalise certain misclassification (e.g. underpredictions) more.

Multi-source Datasets

Diverse fact-check sources with varied verification styles

Annotator agreement metrics ensure label consistency and fairness