Advnced Deep Learning
Assignment 3

Dengke Chen

May 28, 2024

1 Part B. Recurrent Neural Networks

1.1 Introduction

In this study, aim to evaluate the generalization capabilities of two types of recurrent neural networks—Elman (Vanilla) RNNs and LSTM—on a formal language task. The formal language is $a^n b^n c^n$, where sequences consist of a followed by b and then c, with equal counts of each character. Investigate how well these models can classify whether sequences belong to this language, particularly for sequences longer than those seen during training.

1.2 Task & Data

Task:

The task is to classify sequences as members or non-members of the language $a^n b^n c^n$. We generate a dataset of sequences with lengths up to 20 for training and validation, ensuring balanced classes (members and non-members). For evaluation, we generate sequences with lengths from 21 to 100 to test the models' ability to generalize to longer sequences.

Data:

Training Dataset (saved in the formal_language_dataset.csv file):

Generate strings in the form of $a^n b^n c^n$ and label them as 1 (members).

Generate random strings of the same alphabet that do not follow the $a^n b^n c^n$ pattern and label them as 0 (non-members). Ensure variability to cover different types of errors.

Ensure the dataset includes strings of length 20 or less.

Balance the dataset to avoid bias due to skewed distribution by sampling an equal number of members and non-members. Generated 10,000 samples with equal numbers of members and non-members (5000 members and 5000 non-members). This balanced dataset prevents bias towards either class during training.

Sample Data:
caababaaacacb,0
aabbcc,1
caaabccaacbbcaccaab,0

aaaaabbbbbccccc,1
aaaabbbbcccc,1
acbccbbacbaa,0
aababbbb,0
aaaaabbbbbccccc,1

Test Dataset (saved in the formal_language_long_dataset.csv file):
Only generate strings in the form of $a^n b^n c^n$ and label them as 1 (members).
Ensure the dataset includes strings of length 21 to 100.
Generated 5400 member samples.

Sample Data:
aaaaaaabbbbbbbccccccc,1
aaaaaaaaaaabbbbbbbbbbbccccccccccc,1
aaaaaaaabbbbbbbbcccccccc,1
aaaaaaabbbbbbbccccccc,1
aaaaaaaaabbbbbbbbbccccccccc,1
aaaaaaaaaabbbbbbbbbbcccccccccc,1
aaaaaaaaaaaaabbbbbbbbbbbbbccccccccccccc,1
aaaaaaaaaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbbbbbbbbbbbbcccccccccccccccccccccccccc,1

The average length of the training sequences is 10.7145, and the average length of the test sequences is 60.

1.3 Predicitve Models

1. Elman RNN:

The Elman RNN is a basic form of a recurrent neural network where each neuron feeds into the next, and information from previous timesteps can influence the current output.

Initialization:

```
def __init__(self, input_size, hidden_size, output_size):
    super(ElmanRNN, self).__init__()
    self.hidden_size = hidden_size
    self.rnn = nn.RNN(input_size, hidden_size, batch_first=True)
    self.fc = nn.Linear(hidden_size, output_size)
```

• input_size: Number of input features per timestep. For character sequences encoded as integers, this is 1 because each character in the sequence is represented as a single integer.
• hidden_size: Number of units in the hidden layer. This controls the capacity of the model.
• output_size: Number of output units. For binary classification, this is 1 because we are performing binary classification (whether the sequence belongs to the language $a^n b^n c^n$ or not).

Forward Pass:

```
def forward(self, x, lengths):
    h0 = torch.zeros(1, x.size(0), self.hidden_size).to(x.device)
```

```
        packed = nn.utils.rnn.pack_padded_sequence(x, lengths,
            batch_first=True, enforce_sorted=False)
        out, _ = self.rnn(packed, h0)
        out, _ = nn.utils.rnn.pad_packed_sequence(out, batch_first=True)
        out = self.fc(out[torch.arange(out.size(0)), lengths - 1])
        return out
```

• Input: x is the batch of input sequences, and lengths is the actual lengths of these sequences.
• Hidden State Initialization: h0 is initialized to zeros.
• Packing the Sequence: pack_padded_sequence converts the padded sequences into a packed sequence object.
• RNN Layer: Processes the packed sequence. The RNN outputs hidden states at each timestep.
• Unpacking the Sequence: pad_packed_sequence converts the packed output back to padded sequences.
• Final Output: We take the hidden state at the last valid timestep for each sequence and pass it through a fully connected layer to get the final output.

2. LSTM:
LSTM networks are an advanced form of RNNs designed to handle the vanishing gradient problem. They include additional gates to control the flow of information.

Initialization:

```
    def __init__(self, input_size, hidden_size, output_size):
        super(LSTM, self).__init__()
        self.hidden_size = hidden_size
        self.lstm = nn.LSTM(input_size, hidden_size, batch_first=True)
        self.fc = nn.Linear(hidden_size, output_size)
```

• Similar to the Elman RNN, but we use an LSTM layer instead of an RNN layer.

Forward Pass:

```
    def forward(self, x, lengths):
        h0 = torch.zeros(1, x.size(0), self.hidden_size).to(x.device)
        c0 = torch.zeros(1, x.size(0), self.hidden_size).to(x.device)
        packed = nn.utils.rnn.pack_padded_sequence(x, lengths,
            batch_first=True, enforce_sorted=False)
        out, _ = self.lstm(packed, (h0, c0))
        out, _ = nn.utils.rnn.pad_packed_sequence(out, batch_first=True)
        out = self.fc(out[torch.arange(out.size(0)), lengths - 1])
        return out
```

• Hidden and Cell State Initialization: initialize both the hidden state h0 and the cell state c0 with zeros.
• The rest of the process is similar to the Elman RNN, using packing and unpacking functions to handle variable-length sequences.

1.4 Experimental Setup

1. Data Generation: Create balanced datasets for training (10,000 samples (5000 members and 5000 non-members) of length ≤ 20) and testing (5400 samples of length 21 to 100).

2. Data Splitting

To evaluate the models effectively, I split the dataset into training, validation, and test sets:

1. Training Set: Used to train the models.
2. Validation Set: Used during hyper-parameter search to evaluate and select the best model configurations.
3. Test Set: Used to assess the generalization performance of the final models.

Why Split the Data?

• Training Set: The training set was used to update the model weights. It comprised 80% of the total dataset, ensuring that the models had enough data to learn from.

• Validation Set: The validation set was critical for hyper-parameter tuning. By evaluating the models on this set, we could select the best hyper-parameters without overfitting to the training data. This set comprised 20% of the total dataset.

• Test Set: The test set consisted of sequences of length 21 to 100 (5400 samples), which were not seen during training. This set was used to evaluate the models' ability to generalize to longer sequences. The average length of the test sequences was calculated to be representative of the longer sequences.

3. Hyper-Parameter Search

I performed a grid search to find the best hyper-parameters for both the Elman RNN and LSTM models. The hyper-parameters I searched included:

• Hidden size: [25, 50, 100]
• Learning rate: [0.001, 0.01, 0.1]
• Batch size: [32, 64, 128]

The grid search process involved the following steps:

1. Initialize the Model: For each combination of hyper-parameters, I initialized the model with the given hidden size, learning rate, and batch size.
2. Train the Model: I trained the model for a small number of epochs (5 epochs) to quickly assess its performance.
3. Evaluate the Model: After training, we evaluated the model on the validation set to determine its accuracy.
4. Select Best Parameters: We selected the combination of hyper-parameters that resulted in the highest validation accuracy.

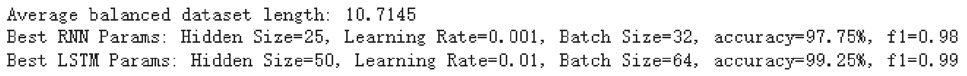
This approach ensured that explored different configurations to find the optimal set of hyper-parameters for each model, thereby allowing a fair comparison between the Elman RNN and LSTM models.

4. Training and Validation: Train each model for 10 epochs with the best-found hyper-parameters and verify the performance of the final trained model on validation set.

5. Evaluation: Evaluate the models on the test dataset using accuracy and F1 score metrics. Additionally, plot the performance over varying sequence lengths.

1.5 Results & Discussion

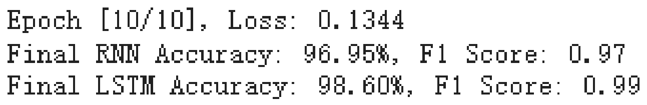
Best Hyper-Parameters:


```
Average balanced dataset length: 10.7145
Best RNN Params: Hidden Size=25, Learning Rate=0.001, Batch Size=32, accuracy=97.75%, f1=0.98
Best LSTM Params: Hidden Size=50, Learning Rate=0.01, Batch Size=64, accuracy=99.25%, f1=0.99
```

Best RNN Params: Hidden Size=25, Learning Rate=0.001, Batch Size=32, accuracy=97.75%, f1=0.98

Best LSTM Params: Hidden Size=50, Learning Rate=0.01, Batch Size=64, accuracy=99.25%, f1=0.99

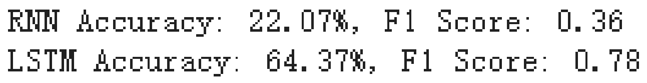
Validation Performance of the final trained model on Validation Set:


```
Epoch [10/10], Loss: 0.1344
Final RNN Accuracy: 96.95%, F1 Score: 0.97
Final LSTM Accuracy: 98.60%, F1 Score: 0.99
```

Final RNN Accuracy: 96.95%, F1 Score: 0.97

Final LSTM Accuracy: 98.60%, F1 Score: 0.99

Generalization Performance on Long Sequences:


```
RNN Accuracy: 22.07%, F1 Score: 0.36
LSTM Accuracy: 64.37%, F1 Score: 0.78
```

RNN Accuracy: 22.07%, F1 Score: 0.36

LSTM Accuracy: 64.37%, F1 Score: 0.78

Plot performance (accuracy and F1) over sentence length (only n members of the language of length 21 to100.):

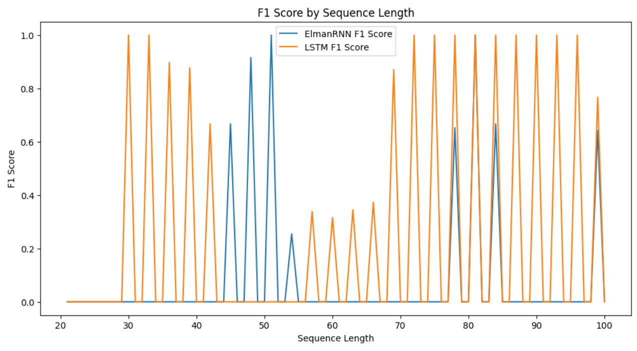


Figure 1: performance (F1) over sentence length

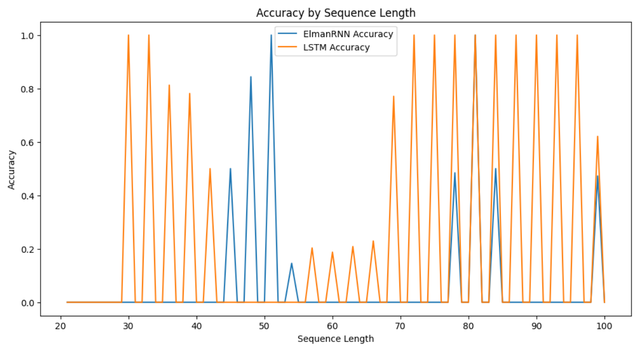


Figure 2: performance (accuracy) over sentence length

Performance by Sequence Length:

• The performance of the two models on short language sequences (length less than 20) is significantly better than that of long language sequences (length longer than 21).

• The LSTM outperforms the RNN across all sequence lengths, demonstrating better generalization to longer sequences.

1.6 Conclusion

RNNs: Effective for shorter sequences but struggle with longer dependencies due to the vanishing gradient problem.
LSTMs: Robust performance across various sequence lengths, effectively handling nested and long-term dependencies.
LSTMs demonstrate superior performance over Elman RNNs in both validation and generalization tasks. The LSTM's ability to maintain higher accuracy and F1 scores for longer sequences indicates its effectiveness in capturing long-term dependencies inherent in the formal language $a^n b^n c^n$. LSTMs are better suited for tasks requiring the modeling of long-term dependencies. The stark difference in generalization ability highlights the importance of model selection for tasks with nested dependencies.

2 Part C. Zero-shot Prompting

2.1 Task & Data

Task:
To test the zero-shot performance of large language models (LLMs) on predicting whether a tweet belongs to a parody account or a real politician account.

Data:
Two tweets were used for this task:
Tweet 1: ”It’s the #GimmeFive challenge, presidential style.” shared by the real politician account @BarackObama.
Tweet 2: ”It’s up to you, America, do you want a repeat of the last four years, or four years staggeringly worse than the last four years?” shared by the parody account @SecretMitRomney.

2.2 Experimental Setup

Each tweet was evaluated using two different templates. Template 1 was formulated as a question, and Template 2 was formulated as an instruction.

Template 1-question:

- Prompt 1: Is the following tweet from a parody account or a real politician account? [Tweet: It’s the #GimmeFive challenge, presidential style.]
- Prompt 1: Is this tweet posted by a real politician or a parody account? [Tweet: It’s up to you, America, do you want a repeat of the last four years, or four years staggeringly worse than the last four years?]

Template 2-instruction:

- Prompt 2: Classify the following tweet as ’real’ if it is from a real politician account or ’parody’ if it is from a parody account: [Tweet: It’s the #GimmeFive challenge, presidential style.]
- Prompt 2: Label the following tweet as ’real’ if it is from a real politician account or ’parody’ if it is from a parody account: [Tweet: It’s up to you, America, do you want a repeat of the last four years, or four years staggeringly worse than the last four years?]

2.3 Evaluation

Evaluation Method:
The evaluation was conducted on the Chatbot Arena platform using the ”Arena (battle)” mode where models are selected randomly and compared

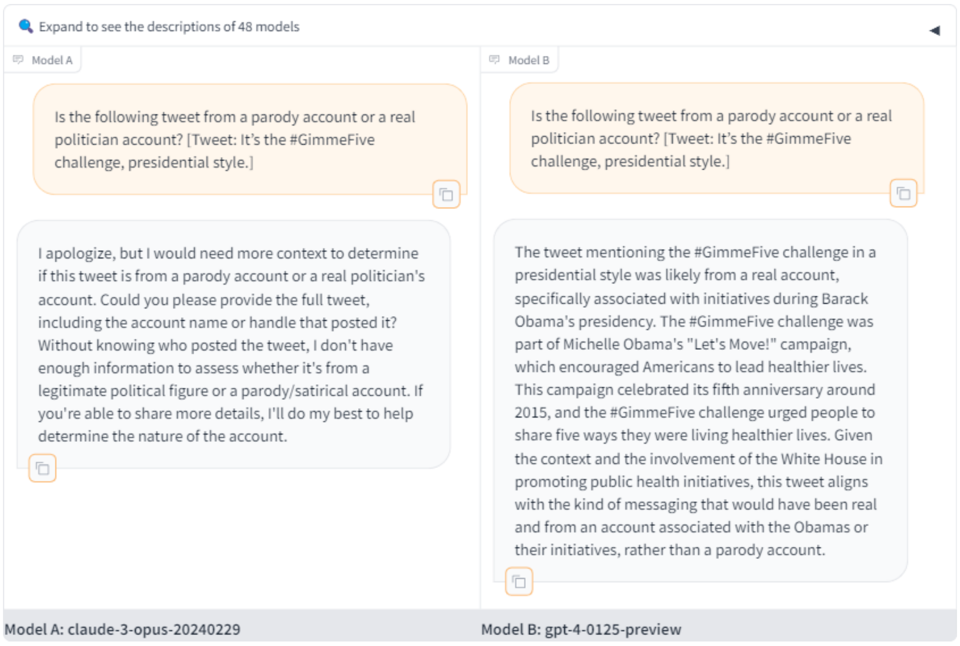
Models Evaluated:

Claude-3-opus-20240229
• Open-source: No • Developer: Anthropic
GPT-4-0125-preview
• Open-source: No • Developer: OpenAI
GPT-4-0613
• Open-source: No • Developer: OpenAI
Mistral-large-2402
• Open-source: Yes • Developer: Mistral AI
Command-r-plus
• Open-source: No • Developer: Cohere
Gemini-advanced-0514
• Open-source: No • Developer: Google DeepMind
Command-r
• Open-source: No • Developer: Cohere
GPT-4o-2024-05-13
• Open-source: No • Developer: OpenAI

2.4 Results & Analysis

Tweet 1: ”It’s the #GimmeFive challenge, presidential style.”

Template 1-question



Is the following tweet from a parody account or a real politician account? [Tweet: It's the #GimmeFive challenge, presidential style.]

I apologize, but I would need more context to determine if this tweet is from a parody account or a real politician's account. Could you please provide the full tweet, including the account name or handle that posted it? Without knowing who posted the tweet, I don't have enough information to assess whether it's from a legitimate political figure or a parody/satirical account. If you're able to share more details, I'll do my best to help determine the nature of the account.

Is the following tweet from a parody account or a real politician account? [Tweet: It's the #GimmeFive challenge, presidential style.]

The tweet mentioning the #GimmeFive challenge in a presidential style was likely from a real account, specifically associated with initiatives during Barack Obama's presidency. The #GimmeFive challenge was part of Michelle Obama's "Let's Move!" campaign, which encouraged Americans to lead healthier lives. This campaign celebrated its fifth anniversary around 2015, and the #GimmeFive challenge urged people to share five ways they were living healthier lives. Given the context and the involvement of the White House in promoting public health initiatives, this tweet aligns with the kind of messaging that would have been real and from an account associated with the Obamas or their initiatives, rather than a parody account.

Model A: claude-3-opus-20240229

Model B: gpt-4-0125-preview

Figure 3: Tweet 1 - Template 1 - Claude-3-opus-20240229 vs. GPT-4-0125-preview

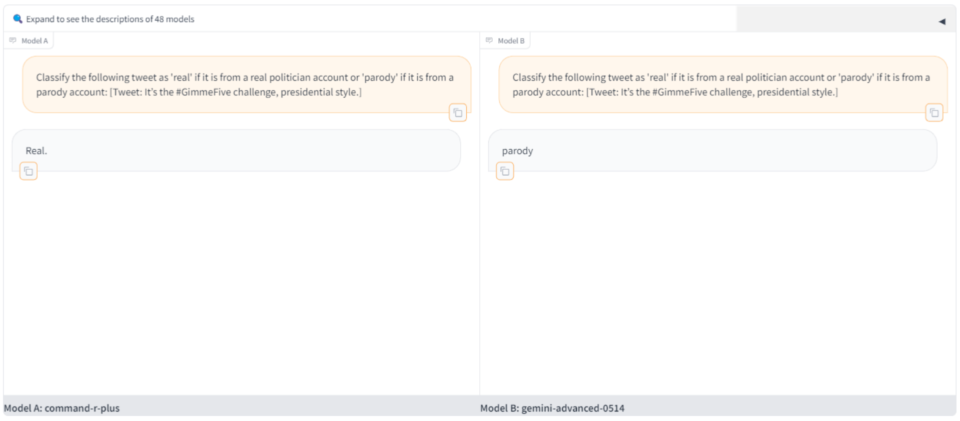
• Prompt 1: Is the following tweet from a parody account or a real politician account? [Tweet: It’s the #GimmeFive challenge, presidential style.]

• Model A (Claude-3-opus-20240229): Provided an ambiguous response requesting more context.

• Model B (GPT-4-0125-preview): Correctly identified the tweet as likely from a real politician account, associated with Barack Obama’s initiative.

Winner: Model B (GPT-4-0125-preview)

Template 2-instruction


Expand to see the descriptions of 48 models

Model A

Classify the following tweet as 'real' if it is from a real politician account or 'parody' if it is from a parody account: [Tweet: It's the #GimmeFive challenge, presidential style.]

Real.

Model A: command-r-plus

Model B

Classify the following tweet as 'real' if it is from a real politician account or 'parody' if it is from a parody account: [Tweet: It's the #GimmeFive challenge, presidential style.]

parody

Model B: gemini-advanced-0514

Figure 4: Tweet 1 - Template 2 - Command-R-Plus vs. Gemini-advanced-0514

• Prompt 2: Classify the following tweet as 'real' if it is from a real politician account or 'parody' if it is from a parody account: [Tweet: It's the #GimmeFive challenge, presidential style.]

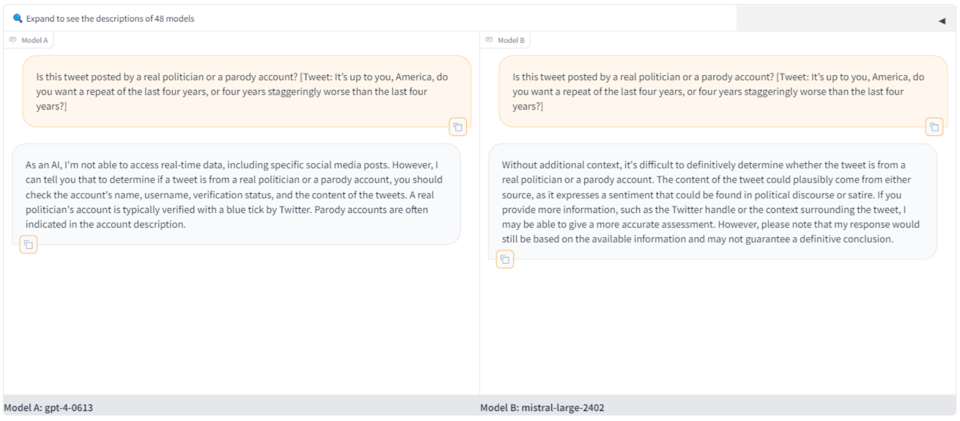
• Model A (Command-R-Plus): Real

• Model B (Gemini-advanced-0514): Parody

Winner: Model A (Command-R-Plus)

Tweet 2: "It's up to you, America, do you want a repeat of the last four years, or four years staggeringly worse than the last four years?"

Template 1-question


Expand to see the descriptions of 48 models

Model A

Is this tweet posted by a real politician or a parody account? [Tweet: It's up to you, America, do you want a repeat of the last four years, or four years staggeringly worse than the last four years?]

As an AI, I'm not able to access real-time data, including specific social media posts. However, I can tell you that to determine if a tweet is from a real politician or a parody account, you should check the account's name, username, verification status, and the content of the tweets. A real politician's account is typically verified with a blue tick by Twitter. Parody accounts are often indicated in the account description.

Model A: gpt-4-0613

Model B

Is this tweet posted by a real politician or a parody account? [Tweet: It's up to you, America, do you want a repeat of the last four years, or four years staggeringly worse than the last four years?]

Without additional context, it's difficult to definitively determine whether the tweet is from a real politician or a parody account. The content of the tweet could plausibly come from either source, as it expresses a sentiment that could be found in political discourse or satire. If you provide more information, such as the Twitter handle or the context surrounding the tweet, I may be able to give a more accurate assessment. However, please note that my response would still be based on the available information and may not guarantee a definitive conclusion.

Model B: mistral-large-2402

Figure 5: Tweet 2 - Template 1 - GPT-4-0613 vs. Mistral-large-2402

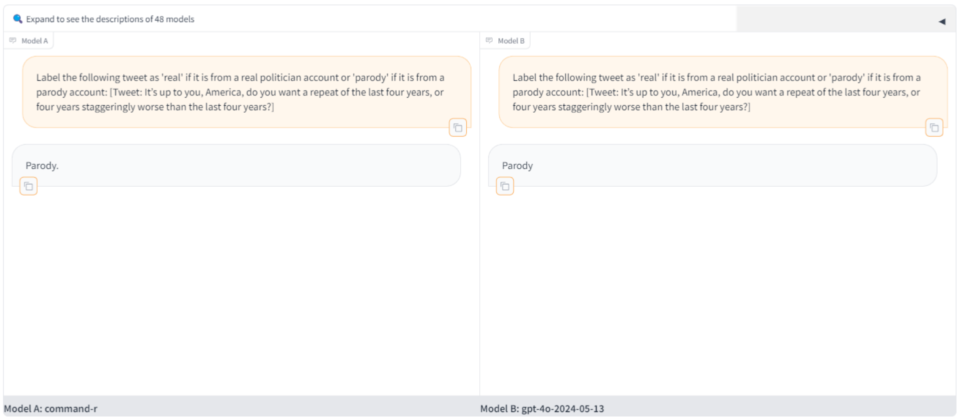
• Prompt 1: Is this tweet posted by a real politician or a parody account? [Tweet: It's up to you, America, do you want a repeat of the last four years, or four years staggeringly worse than the last four years?]

• Model A (GPT-4-0613): Provided an ambiguous response, indicating the need to check the account details.

• Model B (Mistral-large-2402): Provided an ambiguous response, indicating the need for more context.

Winner: Neither, both are bad.

Template 2-instruction



Label the following tweet as 'real' if it is from a real politician account or 'parody' if it is from a parody account: [Tweet: It's up to you, America, do you want a repeat of the last four years, or four years staggeringly worse than the last four years?]

Parody.

Label the following tweet as 'real' if it is from a real politician account or 'parody' if it is from a parody account: [Tweet: It's up to you, America, do you want a repeat of the last four years, or four years staggeringly worse than the last four years?]

Parody

Model A: command-r

Model B: gpt-4o-2024-05-13

Figure 6: Tweet 2 - Template 2 - Command-R vs. GPT-4o-2024-05-13

• Prompt 2: Label the following tweet as ’real’ if it is from a real politician account or ’parody’ if it is from a parody account: [Tweet: It’s up to you, America, do you want a repeat of the last four years, or four years staggeringly worse than the last four years?]

• Model A (Command-R): Parody

• Model B (GPT-4o-2024-05-13): Parody

Winner: Tie

Observations: Proprietary models provided more contextually accurate answers. Instruction-based prompts were generally more reliable for zero-shot classification tasks.

The evaluation demonstrated that while some models performed better than others, consistency and accuracy were achieved in most cases using both question and instruction templates. Notably, proprietary models like GPT-4-0125-preview provided more contextually accurate responses, while open-source models like Mistral-large-2402 required more information to make a determination. The instruction template (Template 2) yielded clearer and more straightforward results. The evaluation highlighted that while proprietary models like GPT-4 variants performed better in context-based questions, both proprietary and open-source models could handle instruction-based prompts accurately. Instruction-based prompts are more reliable for zero-shot classification tasks when context is limited.