Truth or DeGPTion: Evaluating Lie Detection Capabilities of GPT-3.5 through Fine-Tuning on Personal Opinions, Autobiographical Memories, and Intentions

Tanner Graves
tanneraaron.graves@studenti.unipd.it
ID: 2073559

Marco Uderzo
marco.uderzo@studenti.unipd.it
ID: 2096998

Francesco Vo
francesco.vo@studenti.unipd.it
ID: 2079413

Mehran Faraji
mehran.faraji@studenti.unipd.it
ID: 2071980

Claudio Palmeri
claudio.palmeri@studenti.unipd.it
ID: 2062671

Abstract

This paper aims at evaluating the capabilities of GPT3.5 in the task of Lie Detection. This is done through the fine-tuning of GPT-3.5 on three English-language datasets encompassing personal opinions, autobiographical memories, and future intentions. Fine-tuning of LLMs consists in adapting a pre-trained language model to a specific task by further training the model on task-specific data, thereby enhancing its ability to generate contextually relevant and coherent text in line with the desired task objectives. In our investigation, the objective is to discern and classify instances of truth or deception.

1. Introduction

Numerous academic publications consistently prove that the human capacity to discriminate between veracity and deception rests at chance-level proficiency. Consequently, an escalating interest has emerged in the application of machine learning (ML) methodologies, particularly those leveraging the Transformer architecture, to enhance the accuracy of truthfulness prediction for statements. The intrinsic aptitude of ML models for pattern recognition empowers them to discern subtle cues that elude human perception. This paper employs OpenAI's GPT-3.5 Large Language Model (LLM) as the focal point, starting with an assessment of the base model's performance. Subsequently, an in-depth examination will be conducted on a fine-tuned GPT-3.5 model, specifically calibrated using the Personal Opinion Dataset (Deceptive Opinions), the Autobiographical Memories Dataset (Hippocorpus), and the Future Intentions Dataset, following *'Scenario 3'* from Loconte et al.[**?**]

1.1. Personal Opinion Dataset

The participants of this study were divided in 4 groups (HIT's) and asked to provide either a truthful or a deceptive opinion on the following topics: Abortion, Cannabis Legalization, Euthanasia, Gay Marriage, Policy on Migrants. An extract from the table is shown below.

Domain	HIT1	HIT2	HIT3	HIT4
Abo	D	T	D	T
CL	T	T	D	D
Eut	T	D	T	D
GM	T	D	T	D
PoM	D	T	D	T

1.2. Autobiographical Memories Dataset

This dataset contains 6854 diary-like short stories about salient life events gathered in 3 steps from 2 groups of people (A,B). The study was conducted as following:

- Stage 1: Group A writes 15-25 sentence truthful stories with a 2-3 sentence summary and a timeframe

- Stage 2: Group B is tasked to write an imaginative story with the summary of group A as a prompt
- Stage 3: After 2 months group A is asked to retell the story starting with their summary as a prompt

At the end a questionnaire was posed.

	# stories	# sents	# words
recalled	2,779	17.8	308.9
imagined	2,756	17.5**	274.2**
retold	1,319	17.3*	296.8**
total	6,854		

1.3. Future Intentions Dataset

The Future Intentions Dataset is comprised of 1640 examples. The study from which this dataset was collected was conducted as following.

All participants are divided into either the truthful or deceptful group. The former were asked to describe a non-work-related activity that would be doing in the next seven days answering the following questions:

- Q1: "Please describe your activity as specific as possible"
- Q2: "Which information can you give us to reassure us that you are telling the truth"

The latter were given 3 activities from the former group, asked which ones didn't apply to them and get randomly assigned one of those. After, they were required to answer Q1, Q2 like the former group. At the end a questionnaire was posed.

Veracity	Activity	Statement given by participant
Truthful	Going swimming with my daughter	We go to a Waterbabies class every week, where my 16-month-old is learning to swim. We do lots of activities in the water, such as learning to blow bubbles, using floats to aid swimming, splashing and learning how to save themselves should they ever fall in. I find this activity important as I enjoy spending time with my daughter and swimming is an important life skill.
Deceptive	Going swimming with my daughter (assigned)	I will be taking my 8-year-old daughter swimming this Saturday. We'll be going early in the morning, as it's generally a lot quieter at that time, and my daughter is always up early watching cartoons anyway (5 am!). I'm trying to teach her how to swim in the deep end before she starts her new school in September as they have swimming lessons there twice a week.

2. Methods

2.1. Dataset Preprocessing

In our methodology, we adopted *Scenario 3* as outlined by Loconte et al. [**?**]. This scenario involved the aggregation of three distinct train and test sets from *Scenario 1*, which encompassed statements related to opinions, memories, and future intentions. Subsequently, the model underwent a fine-tuning process using the aggregated sets. The objective of *Scenario 3* was to assess the model's overall capability to classify both truthful and deceptive statements across diverse contexts. Notably, in this scenario, the model was not fine-tuned to a specific type of statement; instead, it was evaluated on opinions, memories, and future intentions collectively, treating them as a randomly shuffled set of statements to be classified without bias towards any particular category.

The implementation of this process involved the merging of individual datasets into a unified one after cleaning and shuffling them. The resulting dataset was then partitioned into training, validation, and test sets. It is essential to highlight that we intentionally did not utilize the entire initial dataset. The decision to employ a smaller training dataset was aimed to computationally simplify the training process of the LLM, while still ensuring effective learning and generalization.

As a final step, the datasets were formatted into `JSON` format to align with the expected input format of the OpenAI API, ensuring seamless integration with the API itself. For a detailed understanding of the implementation code, please refer to the Appendix section of this paper, where the Python code is provided for reference.

Regarding the fine-tuning process, the two different models were trained using a smaller and bigger dataset as follows:

- a subset of the original dataset (Train: 210 examples; Validation: 45 examples; Test: 45 examples).
- the full dataset (Train: 6765 examples; Validation: 1450 examples; Test: 1450 ??????? examples).

2.2. GPT-3.5 Fine-Tuning

In our study, we trained the model using the OpenAI API and evaluated its performance through testing and comparison with GPT-3.5. We also conducted additional experiments to understand how engineering the system prompt affects overall performance. One specific observation we made was regarding the baseline GPT-3.5's tendency to provide verbose or indecisive answers. When faced with statements that could be easily classified as genuine or deceptive, the model tended to provide detailed responses. However, in cases where it lacked sufficient information to confidently categorize a statement, the model often chose not to give a definitive answer.

To exemplify this behavior, let's consider the following scenario: when the model encounters uncertainty regarding a statement, it chooses not to offer a definitive classification. This characteristic underscores the model's cautious approach when confronted with a potential lack of information, prioritizing the avoidance of incorrect responses over providing answers that may be inaccurate.

```
User: "Each and every abortion
is essentially a tragedy. The
potential mother will suffer
unforeseen consequences. Society
as a whole will be deprived of
the potential it could have
received from the new life."


Baseline GPT-3.5: "There is no
objective truth to the statement
as it expresses subjective opinions
and beliefs about abortion. It cannot
be definitively classified as
'True' or 'False'."
```

To mitigate this issue, we engineered a system prompt designed to discourage the verbose or indecisive behavior exhibited by the model. This tailored prompt also served the crucial purpose of clearly conveying the task requirements to the model. Each time the model encountered an example query, it was presented with this prompt to guide its understanding and response.

Recognizing the importance of efficiency in both training and query costs, we took care to ensure that the instructions in the prompt were concise. The goal was to minimize any unnecessary token overhead that could contribute to an increased cost associated with training and queries.

```
System Prompt to Fine-Tuned GPT-3.5:

"You are an expert capable of
discerning truthful from deceptive
opinions based on speech patterns.
Definitively classify the following
statement as 'True' or 'False',
based on the likelihood the statement
represents a genuinely held belief
or a deception."
```

During the fine-tuning process, the model was incentivized to adopt the expected behavior and adhere to the specified output format. This reinforcement mechanism ensured that the model learned and internalized the desired characteristics, minimizing the likelihood of verbose or indecisive responses.

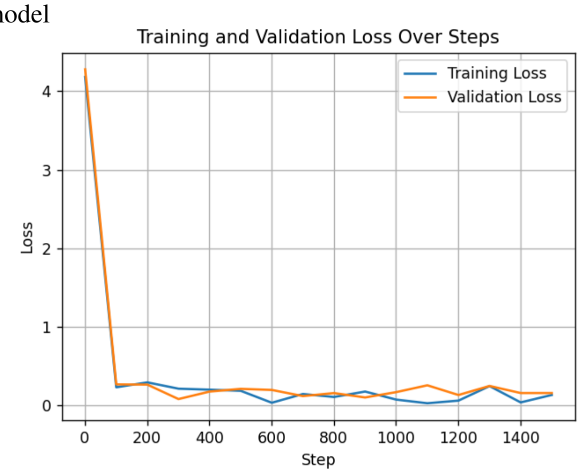
The hyperparameters used for fine-tuning the models are declared in the tables below, together with the fine-tuning training losses over the steps.

2.2.1 300-Model Fine-Tuning

change the below table to hyperparams of actual 300-model

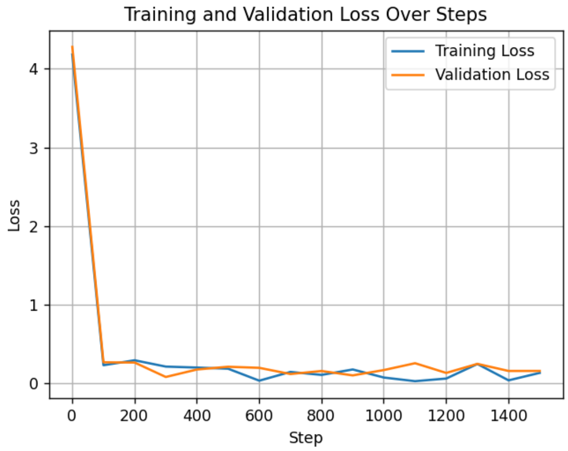
Model	Hyperparameter	Value
Model 1	Learning Rate	0.001
	Epochs	50
	Batch Size	32
Model 2	Learning Rate	0.01
	Epochs	30
	Batch Size	64

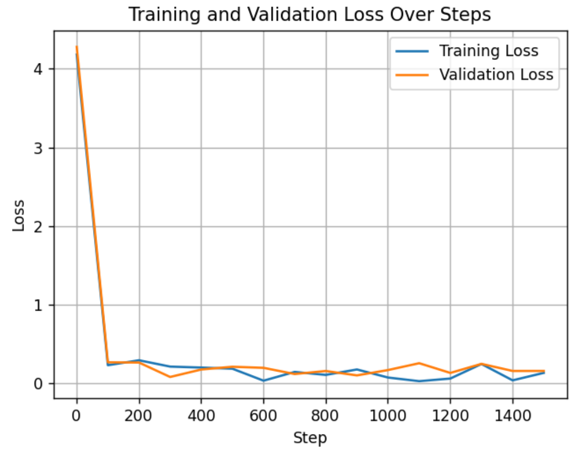
change the below graph to training loss of actual 300-model



2.2.2 Full-Model Fine-Tuning

Model	Hyperparameter	Value
Model 1	Learning Rate	0.001
	Epochs	50
	Batch Size	32
Model 2	Learning Rate	0.01
	Epochs	30
	Batch Size	64





2.2.3 N-CV-Model Fine-Tuning

k-fold cross validation is a practice to be talked about carefully when it comes to experiments with cost constraints. Indeed, such validation becomes very expensive, since OpenAI API calls are charged with a fee. In particular, GPT-3.5 Turbo's pricing is $0.0010 per 1,000 tokens for input and $0.0020 per 1,000 tokens for output, where 1000 tokens equal about 750 words. To keep costs low, we decided to perform 3-fold cross validation only on the smaller model. Of course tough, we would expect a slightly better performance using the full model. (Also explain)

add the illustration of the splits

change the below table to hyperparams of actual cv-model

Model	Hyperparameter	Value
Model 1	Learning Rate	0.001
	Epochs	50
	Batch Size	32
Model 2	Learning Rate	0.01
	Epochs	30
	Batch Size	64

change the below graph to training loss of actual cv-model

3. Results

3.1. Lie Detection Task Performance

Initially, our evaluation began with the assessment of the non-fine-tuned baseline GPT-3.5 model, resulting in an accuracy of 71.1%. (**maybe also put in table the class accuracies**). Subsequently, we fine-tuned two distinct models: the `300-Model` and the `Full-Model`, meaning that the models were trained with 300 examples and the full training set respectively. Upon completion of the fine-tuning process, we subjected both models to the same evaluation methodology.

As expected, the `300-Model` performed worse than the `Full-Model`. The latter exhibited a remarkable improvement in performance, achieving an overall accuracy of 82.2%, whereas the former achieved an overall accuracy of 65.5…(??????) This observation suggests that when fine-tuning, performance positively scales with the size of the training sets. This yielded consistent and substantial enhancements in accuracy when it came to differenciating between truthful and deceptive statements, even across multiple contexts. However, the aforementioned finding raises the problem of the trade-off between overall performance of the model and training costs for fine-tuning. In fact, each OpenAI's API call is subject to a small fee, which, considering the total amount of calls needed and its positive relationship with the dimension of the training set, can result in great training costs. In particular, the practice of k-fold cross-validation is very resource-intensive, and can be quite costly.

In adherence to established Machine Learning best practices, the cross-validation approach allows for a comprehensive assessment of the model's generalization capabilities and robustness across diverse data samples, giving more accurate and reliable results. The choice of `k` is crucial for this process, and also determines the amount of resources that will be used. Ideally, 10-fold cross-validation is to be chosen. Though, to keep the cost low, we decided to perform 3-fold cross validation on the `300-Model`. Considering this, it was expected that the cross-validated smaller model would perform worse than the Full-Model, and we could safely assume that, without training costs as a factor, cross-validating the `Full-Model` would have performed way better, closely to the non-cross-validated `Full-Model`. The cross-validated model (name?) achieved ... accuracy ...

The following table provides a detailed breakdown of the models' performances, specifically focusing on accuracy:

Model	Overall Accuracy
Baseline GPT-3.5	71.1%
300-Model	65.5%
Full-Model	82.2%
400-CV-Model	... %

Model	Dataset	Accuracy
Baseline GPT-3.5	Personal Opinions	x%
	Autobiographical Memories	x%
	Future Intentions	x%
300-Model	Personal Opinions	79.1%
	Autobiographical Memories	67.3%
	Future Intentions	50.2%
Full-Model	Personal Opinions	86.3%
	Autobiographical Memories	82.3%
	Future Intentions	69.7%
400-CV-Model	Personal Opinions	86.3%
	Autobiographical Memories	82.3%
	Future Intentions	69.7%

4. Discussion

In the present study, we delved into the effectiveness of Large Language Models, specifically GPT-3.5, in the task of lie detection. We considered both the baseline non-fine-tuned GPT-3.5 and two instances fine-tuned accordingly with *'Scenario 3'* from Loconte et al.[**?**]. Our primary objective was to investigate the model's capacity to learn and generalize intrinsic linguistic representations of deception across a spectrum of contexts. This investigation was conducted using three distinct datasets, each comprising statements related to personal opinions, autobiographical experiences, and future intentions, thereby ensuring a comprehensive exploration of the model's capabilities.

Despite the baseline GPT-3.5 model's inherent capability to surpass human performance - where human capabilities typically hover around chance level - the fine-tuning process on the non-cross-validated `Full-Model` further enhanced its performance by 11.1% in terms of accuracy. This substantial improvement underscores the efficacy of the model to capture the nuances of deceptive language use.

Upon closer examination of performance by class, a noteworthy finding emerged: the models exhibited suboptimal performance on the Future Intentions Dataset, achieving a modest 69.7% accuracy. This specific performance dip in one dataset had a discernible impact on the overall accuracy of the model.

Of particular interest was the observation that training GPT-3.5 on a smaller subset of data compromised results when compared to training on a larger set. This observation has significant implications, making the fine-tuning process problematic in terms of training costs, expecially when following best practices like performing k-fold cross-validation.

When taking the `300-Model` into consideration, the overall accuracy gain compared to the baseline GPT-3.5 was only of ... %

The cross-validated model, instead, ...

Future improvement on the project could be performing 10-fold cross-validation on the `Full-Model` to have a better understanding of the full capabilities of the model in detecting deception. Moreover, the use of stylometric analysis could be leaveraged to assess the problem even more thoroughly.

5. TO DO

- RAW DATA FILE!!
- fine tuning explaination
- cross-validation on smaller model (so we are in the same setting as Loconte, even if it might cost a bit more)
- training hyperparams (see table)

6. Code Availability

The datasets used and all the code used for this project is available at the following GitHub Repository.

References

[1] Loconte et al. Verbal lie detection using large language models. 2023.

7. Appendix

We encourage to browse the Python Code directly from the GitHub repository at the following link.

7.1. Python Code: Scenario 3 Dataset Preprocessing

The following code preprocesses the dataset according to Scenario 3. The original Python Notebook is located at the following link in our GitHub Repository

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split

dc = pd.read_csv('DecOp_data_EN_500.csv', sep=',', encoding='UTF-8')

d1 = []
for i in range(dc.shape[0]):
  row = dc.iloc[i]
  d1.append({'ID': row['ID'],
      'age' :row['age'], 'gender': row['gender'],
  'sent' : row['A'].replace('\n', " ") ,
  'labels'  : row['GT.A']})

  d1.append({'ID': row['ID'],
      'age' :row['age'], 'gender': row['gender'],
  'sent' : row['E'].replace('\n', " ")  ,
  'labels'  : row['GT.E']})
  d1.append({'ID': row['ID'],
      'age' :row['age'], 'gender': row['gender'],
  'sent' : row['GM'].replace('\n', " ")  ,
  'labels'  : row['GT.GM']})

  d1.append({'ID': row['ID'],
      'age' :row['age'], 'gender': row['gender'],
  'sent' : row['Pom'].replace('\n', " ")  ,
  'labels'  : row['GT.Pom']})

  d1.append({'ID': row['ID'],
      'age' :row['age'], 'gender': row['gender'],
  'sent' : row['CL'].replace('\n', " ")  ,
  'labels'  : row['GT.CL']})

decop = pd.DataFrame.from_records(d1)

decop = decop[['ID', 'sent', 'labels']]
decop['type'] = 'A'

decop

hc = pd.read_csv('hcV3-stories.csv', sep=',', encoding='UTF-8')

mem = hc[hc['memType']!='retold']
mem = mem.dropna(subset=['story', 'memType'])
mem = mem[['story', 'memType']]
mem['memType'][mem['memType']=='recalled'] = 'T'
```

```
mem['memType'][mem['memType']=='imagined'] = 'F'
mem = mem.rename(columns={'story': 'sent', 'memType': 'labels'})
mem['type'] = 'B'

mem

intent = pd.read_csv('sign_events_data_statements.csv', encoding="UTF-8")

intent.loc[intent['outcome_class']=='t', 'outcome_class'] = 'T'
intent.loc[intent['outcome_class']=='d', 'outcome_class'] = 'F'
intent['q1'] = intent['q1'].apply(lambda x: x.replace('\n', ''))
intent = intent.rename(columns={'q1': 'sent', 'outcome_class': 'labels'})
intent = intent[['sent', 'labels']]
intent['type'] = 'C'

intent

k = 300
seed = 42

'''
shuffled_df = decop.sample(frac=0.1, random_state=seed).copy()
sample_decop = shuffled_df.iloc[:k,].copy()
sample_decop.drop('ID', axis=1, inplace=True)
sample_decop

data = pd.concat([sample_decop, mem.iloc[:k,].copy(), intent.iloc[:k,].copy()],
                 ignore_index=True)
'''

decop.drop('ID', axis=1, inplace=True)

data = pd.concat([decop, mem, intent], ignore_index=True)
data_shuffle = data.sample(frac=1, random_state=seed+27).reset_index(drop=True)
data_shuffle['labels'] = data_shuffle['labels'].map({'T': 1, 'F': 0})
data_shuffle.rename(columns={'sent': 'text', 'labels': 'label', 'type': 'set'},
                                                                    inplace=True)

data_shuffle = data_shuffle.iloc[:k,]

train_data, temp_data = train_test_split(data_shuffle, test_size=0.3, random_state=42)

val_data, test_data = train_test_split(temp_data, test_size=0.5, random_state=42)

train_data.to_json(f'train_data_{k}.json', orient='records', lines=True)

val_data.to_json(f'val_data_{k}.json', orient='records', lines=True)

test_data.to_json(f'test_data_{k}_class.json', orient='records', lines=True)

data_shuffle.shape

train_data.shape
```

7.2. Python Code: GPT-3 Fine-Tuning

The following code fine-tunes GPT-3 using the official OpenAI APIs. The original Python Notebook is located at the following link in our GitHub Repository

```
import os
from openai import OpenAI
import pandas as pd
import json

key = os.environ.get("OPENAI_API_KEY")
client=OpenAI(api_key=key)

import json

def sanitize_to_utf8(input_str):
    """
    Sanitizes a string field to UTF-8.
    """
    return input_str.encode('utf-8', 'replace').decode('utf-8')

def process_jsonl_file(input_jsonl_path, output_jsonl_path):
    """
    Processes a JSONL file line by line, sanitizes each line to UTF-8,
    and saves it to a new JSONL file.
    """
    with open(input_jsonl_path, 'r', encoding='utf-8') as input_file, \
         open(output_jsonl_path, 'w', encoding='utf-8') as output_file:

        for line in input_file:
            # Parse JSON line
            json_obj = json.loads(line)

            # Sanitize data
            sanitized_json_obj = {key: sanitize_to_utf8(value) if isinstance(value, str)
                                  else value for key, value in json_obj.items()}

            # Write sanitized JSON object to new JSONL file
            output_file.write(json.dumps(sanitized_json_obj) + '\n')

"""Generating intermediate 1500 example sets.

TODO: Preserve what dataset examples come from so we can breakdown performance by dataset.
"""

# Paths to your files
input_jsonl_path = 'train_data_full.json'
output_jsonl_path = 'output.jsonl'

# Process the JSONL file
process_jsonl_file(input_jsonl_path, output_jsonl_path)

sys_prompt = "You are an expert capable of discerning truthful from deceptive opinions
                                                                  based on speech patterns."
```

```
def gen_finetune(input, output, test=False):
    with open(input, 'r', encoding='utf-8') as data_in, \
        open(output, 'w') as gpt_out:
        for i, line in enumerate(data_in):
            user_prompt = json.loads(line)['text']
            sys_reply = "True" if json.loads(line)["label"] == 1 else "False"
            if not test:
                payload = {"messages": [{"role": "system", "content": sys_prompt},
                {"role": "user", "content": user_prompt}, {"role": "assistant",
                                                            "content": sys_reply}]}
            else:# exclude response from test set
                payload = {"messages": [{"role": "system", "content": sys_prompt},
                {"role": "user", "content": user_prompt}]}
            gpt_out.write(json.dumps(payload) + '\n')

gen_finetune('train_data_300.json', 'train300_gpt.jsonl')
gen_finetune('val_data_300.json', 'val300_gpt.jsonl')

gen_finetune('train_data_1500.json', 'train1500_gpt.jsonl')
gen_finetune('val_data_1500.json', 'val1500_gpt.jsonl')

gen_finetune('train_data_full.json', 'train_full_gpt.jsonl')
gen_finetune('val_data_full.json', 'val_full_gpt.jsonl')

"""Example of querying model for classification.
Example on model trained on 300 example set.
"""

test_lie = "One morning three months ago I was in a hurry and tripped on the steps
while running to take a shower before work. I ended up fracturing 4 metatarsal bones
that required two surgeries to fix. I was really having a hard time not being able
to walk whenever I wanted to. I really had such a bad attitude at the beginning
because I was so used to being independent. Now that I have recovered I have a
new appreciation for my ability to walk. I feel like the whole time I couldn't
walk I was thinking about how much I took that ability for granted. But now I
choose to walk more than I ever had before. When I walk the dogs I go further
out of my way just to enjoy the ability to do it. I am completely recovered
and I am going to take this as a lesson learned. Nothing is more important
that my personal health. I need to make sure that even if I am running late,
I need to take my time and be careful. Instead f making it to work on time,
I ended up missing weeks of work. Now all I do at work is try and catch up
with everything I missed. It was really nice that people at work came and
visited me at the hospital. I really appreciated all the flowers and candy
and food that was delivered. I think that this showed me how loved I truly am."

test_truth = "Each and every abortion is essentially a tragedy.
The potential mother will suffer unforeseen consequences. Society as a whole will
be deprived of the potential it could have received from the new life."

response = client.chat.completions.create(
  model="ft:gpt-3.5-turbo-0613:personal::8S542QSs",
  messages=[
```

```
    {"role": "system", "content": sys_prompt},
    {"role": "user", "content": test_lie},
    {"role": "user", "content": test_truth},
  ]
)
response.choices[0].message.content

mdls = {
    '300':"ft:gpt-3.5-turbo-0613:personal::8S542QSs",
    '3.5':"gpt-3.5-turbo",
    '1500':"ft:gpt-3.5-turbo-0613:personal::8TAkdwiX"
    }

def predict(text, model, sys_prompt):
    response = client.chat.completions.create(
    model=model,
    messages=[
        {"role": "system", "content": sys_prompt},
        {"role": "user", "content": text}
    ]
    )
    return response.choices[0].message.content

test_truth

test_lie

def eval(test_file, mdl, return_df = False, sys_prompt=sys_prompt, by_class=False):
    msgs = []
    preds = []
    trues = []
    replies = []
    set = []
    with open(test_file, 'r') as f:
        for ex in f:
            ex = json.loads(ex)
            msg = ex['text']
            #pred = predict(msg, mdl) == 'True' #'True' if predict(msg) else 'False'
            reply = predict(msg, mdl, sys_prompt) #for debug
            pred = reply == 'True'
            true = ex['label'] == 1
            msgs.append(msg)
            preds.append(pred)
            trues.append(true)
            replies.append(reply) # debug
            if by_class:
                set.append(ex['set'])
    if by_class:
         test_df = pd.DataFrame.from_dict({'msg': msgs, 'reply':replies,
                            'preds': preds, 'target': trues, 'set':set})
    else:
        test_df = pd.DataFrame.from_dict({'msg': msgs, 'reply':replies,
                            'preds': preds, 'target': trues})
```

```
    acc = (test_df['preds'] == test_df['target']).mean()
    if return_df:
        return (test_df, acc)
    else:
        return acc


test_truth

sys_prompt_base = sys_prompt + ' Definitively classify the following statement
                as \'True\' or \'False\', based on the likelihood  the statement
                represents a genuinely held beleif or a deception.'
#base_df, base_acc = predict('test_data_300.json', mdls['3.5'])
predict(test_truth, mdls['300'], sys_prompt=sys_prompt)

test300_df, test300_acc = eval('test_data_300.json', mdls['300'], True,
                                                            sys_prompt=sys_prompt)
print(test300_acc)
test300_df

test300_acc

test35_df, test35_acc = eval('test_data_300.json', mdls['3.5'], True,
                                                        sys_prompt=sys_prompt_base)
print(test35_acc)
test35_df

len(test1500_df)

# Takes a lot of time, cost = $1
#test1500_df, test1500_acc = eval('test_data_1500.json',mdls['1500'],True,
                                                            sys_prompt=sys_prompt)
print(test1500_acc)
test1500_df

test1500_df.to_csv('results1500.csv')

"""Model / Dataset Bias analysis:"""

print('Model truth classification rate: ')
print(test300_df['preds'].mean())
print('Dataset truthful statement rate: ')
print(test300_df['true'].mean())

"""Model performance by dataset"""

sys_prompt

class_df, class_acc = eval('test_data_1500.json', mdl=mdls['1500'],
                            sys_prompt=sys_prompt, return_df=True, by_class=True)

class_df['correct'] = class_df['preds'] == class_df['target']
class_df.groupby('set')['correct'].mean()
```

```
class_df.to_csv('results_class.csv')
```