Table of Content

SIOP Machine Learning Competition 2024

First, I want to express my thanks for organizing such a remarkable challenge. It was indeed a formidable task.

We did not optimize the code for ***asynchronous calls***, which may result in extended execution times for the entire task set. If you encounter any issues, please don't hesitate to inform me.

We employed a variety of methods and models from different providers. Our code includes a ***utils script***, where we developed utility functions such as LLM calling, LLM authentication, data loading, *EvalAI* submissions, and more.

Within the *utils* script, you have the opportunity to input your authentication details for the various providers. Although we used numerous providers during the model development phase, we chose a few models for testing phase.

The following LLM models were utilized in the testing phase:

- OpenAI's GPT models
- OpenAI's Completion Models
- Anthropic's Claude 3 Models
- Google's Gemini and Palm Models from AI Studio
- Google's Gemini 1.5 Pro models from Vertex AI (Google Cloud), including:
    - Gemini-experimental

The following models were not utilized in our final predictions, though they are present in our utils scripts. You may add placeholder keys for these providers:

- Coherer's Coral Models
- Mistral's Mixtral Model

We recommend setting up a Python environment and running the ***requirements.txt*** file to install the necessary packages via pip. If you face any difficulties, please let me know.

Set Up Environment

To set up a Python environment and install dependencies from a requirements.txt file, you can follow these steps. These instructions are applicable for Unix-based systems (Linux/macOS) and Windows, with slight modifications as noted.

1- Create Virtual Environments

Navigate to your project directory where you want the environment to be set up.

Create the virtual environment by running:

For Unix-based systems:

python3 -m venv myenv

For Windows:

python -m venv myenv

myenv is the name of the environment directory. You can name it anything you like.

2 - Activate the virtual environment:

For Unix-based systems:

source myenv/bin/activate

For Windows:

myenv\Scripts\activate

3- Install Packages from the *requirements.txt* File

Ensure you have a ***requirements.txt*** file in the project directory. This file contains a list of packages needed for the project, potentially with specific versions.

With the virtual environment activated, install the packages using pip:

pip install -r requirements.txt

Task 1: Empathy Prediction

For us, empathy prediction was a significant challenge in this competition for several reasons. First, most LLM models struggled to distinguish simple greetings, positively worded phrases, and slightly cheerful sentiments from genuinely empathetic content. This might be because of the associative nature of these models, as they tend to correlate certain phrases, words, and tokens with empathetic contents, dismissing the intent behind the texts. Second, we assume that predicting whether a text or dialog contains empathy requires a second-order meta-cognitive perspective taking skills, where one must understand the positions of both the writer and the receiver of the text, mentalizing the writer's intent and the receiver's emotional state. This was challenging for LLM models, as they lacked the necessary capacity for true perspective-taking and mental models that guided their reasoning. In short, while humans can intuitively take such a perspective, LLMs struggled so dearly in our experiments.

To address these challenges, we experimented with various methods and LLM models. In our final approach, we ensembled the predictions of multiple LLM models and applied a self-consistency prompting method to enhance empathy prediction accuracy. Here are the models we developed.

Pre-Processing Steps

1) Pre-processing: Feedback Extraction, Empathic Phrase Extraction, Syntactic Chain-of-Though (COT) Generation

1-1) Actual Feedback Extraction

As I mentioned above, one of the problems the LLMs struggle with was to associate specific phrases and words with empathic contents. Especially when the text contains positive phrases, warm greetings, other sign-off parts, LLM assumes empathic tones. To overcome this issue, we first extracted the actual feedback parts, discarding the greetings and other signoffs. To do so, we generated an `**extractive summary task**` and used simple zero-shot prompting with a powerful model such as Anthropic' Opus, OpenAI's gpt-4-0125-preview, or Google's Gemini 1.5-pro. We used the extracted texts in some of the downstream LLM tasks, which helped us develop more robust models. Please see our prompt for this task in the prompts folder.

1-2) Empathic Phrase Extraction

We noticed that some texts contain more empathic phrases than others, and thus, the density of these empathic sentences can also indicate the overall empathy in the text. Therefore, we generated a similar `**extractive summary task**` and utilized similar zero-shot prompting for extracting empathic sentences— omitting greetings and other cliché parts such as "hope you are having a wonderful travel," etc. We used these extracted empathic sentences in some downstream tasks with LLMs. Please see our prompt for this task in the prompts folder.

1-3) Syntactic COT Generation for Knowledge Distillation: Teacher to Student LLM Models

In one of our experiments, we created a Knowledge Distillation task using Large Language Models (LLMs). The goal was to teach smaller, more efficient models, such as GPT-3.5-turbo or

Mistral's tiny, how to reason by leveraging the knowledge and capabilities of larger, more powerful models like Anthropic' s Opus and OpenAI's GPT-4, which served as teacher models.

To achieve this, we generated synthetic Chains-of-Thought (COTs) using the stronger models. For each training example, we provided the model with feedback text and label information. We then prompted the model with the following instruction:

> "*For the given text, we asked 100 humans, and they classified this text as {label_information}. Analyze each sentence step-by-step and develop reasoning on why humans classified this text as {label_information}."*

(See the actual prompt in PROMPT X for more details.) By using the true label information for each training example, we developed syntactic COTs. These COT examples were then used in a downstream task for the Knowledge Distillation task with LLMs. The purpose was to teach the reasoning process employed by the stronger models to the smaller models, enabling them to benefit from the knowledge and capabilities of their larger counterparts.

Task 1: Prediction Models

1) Model 1a: Text Completion Task: Context Learning with N-Shot Examples with Foundational Models

In Model 1, we use the power of context learning with N-shot examples. We randomly sampled N examples from the training data, creating N/2 examples for positive classes and N/2 examples for negative classes. Then, we generate simple `**text completion**` prompting for the foundational model, such as babbage-002, davinci-002, gpt3-5-instruct, and palm (bison-002 from Google) to predict the target label. We repeated this process ***t*** times (about 5 or 7, choose odd number to calculate consistency classes) with different random samples. The temperature for these models was set to .30-.40. Target classes are encoded as "***Direct***" feedback or "***Encouraging***" feedback to reduce the associative connotation of the word `**empathy**'. In this task, we use "**empathic phrase extractions**" and "**feedback extractions**" rather than full text. When we use the full text, the model suffers. The model accuracy was on the training data .60 - .66.

Example prompt:

{ id: 174; text: The additional commentary you are providing makes it difficult to find the objective facts of your findings while working with a tight deadline. I would like to have a discussion with you what ideas you may have to help make your reports more concise so the team can meet their deadlines.; Class: ***Direct*** }

{ id: 161; text: In order for the process to flow more smoothly I need you to focus and limit your report to data and technical information. Recommendations and opinions could be sent to me on a weekely basis. Also, I would like to sit down and coach you on report writing when I am back. That will be a great development for your career.; Class: Encouraging }

"{ id: 192; text: 1. Please try to be concise in reports and mention facts that teams can refer . We love opinions but lets save those for our brainstorming discussions. 2. For Business writing as you are getting started to help you set up for success we are nominating you for a training program so that your reports are way more effective.; Class: Direct }"

{ id: 86; text: This Beta project is key for the company, we need to success all together. In that respect, key priorities are to build concise reports and with strong business writing. Terry has been within the company for 5 years and is the best one to be consulted to upskill in these areas. Could you please liaise with him and get more quick wins from him. It will be very impactful in your career.; Class: Encouraging }

{ id: 116; text: From what I understand, they would like them to be more concise and straight to the point, as well as more business focused. I recommend you reach out to Terry so you both could review in detail one of the reports he submits. This should help you help you align to their expectations. Additionally, i'd be happy to review the reports before you send them off to Terry and provide my feedback.; **Class: [Text Completion]**

2) Model 1b: Context-Learning with Self-Consistency N-Shot COTs using Chat LLM models

In the same setting, we also experimented with the idea of how the strong chat models with Reinforcement Learning with Human Feedback (RLHF) such as gpt-4-0125-preview etc. can learn the class features and generalize this information to apply to a new example. In this model, we did not use any specific target labels such as "***empathic***" or "***not empathic***" but we provided examples of the classes and asked the model to learn from the underlying classification logic. Then, using this classification reasoning to apply to the new target task. By doing so, we enable the model to learn the class labels without a need for explicit labeling.

Our prompt was:

You are an expert in classifying feedback and understanding the rationale behind classifications. Here's your task:
1. Analyze Examples: You will be given a series of example feedback texts, each with its corresponding one of the classes (e.g., {class0} or {class1}). Carefully examine these examples to understand the reasons why they belong to their respective classes. For each class, provide three clear reasons under <reasoning> tags, focus on the content, tone, and structure of the feedback, and how the person who might recieve it would feel.
<reasoning>
Example for Class 1
Reason 1: ...
Reason 2: ...
Reason 3: ...
</reasoning>

2. Develop Classification Logic: Analyze the reasons you developed to identify the underlying logic that determines why a feedback text should be assigned to a specific class. Consider similarities and differences between examples in each class.
3. Apply Logic to Target Text: Read the final target text. Apply the classification logic you developed to analyze this text. Note similarities and differences between the target feedback text and the provided examples. Continue refining your logic as needed. Remember to focus on the content, tone, and structure of the feedback, and most importantly, how the person who might receive it would feel.
4. Provide Reasoning and Classification: Document your analysis of the target text step-by-step under <reasoning></reasoning> tags, explaining the logic that led to your decision.
5. Final Classification: Under the <classification>'class'</classification> tag, return the classification result for the final text ({class0} or {class1})."

Similar to *Model 1a*, we sampled the training data N times. We created a balanced sample for each class N/2. Then, we repeated this process for each target text ***t*** times to get self-consistency results. In development process, we used various RHLF chat models, such as gpt3-5-turbo, gpt-4-0125-preview, haiku, sonnet, Gemini 1, and Gemini 1.5-pro. The temperature was set to 0.30-0.40. We used "feedback extractions" for this task. In the final test, we use gpt-4. The model accuracy for this method in development and training was around .55 to .60.

3) Model 2: Elo Ratings with the Nearest Neighbors

This is one of the ***interesting*** methods that we devised, hoping to talk about that in the presentation. We got the inspiration for these methods from the RHFL literature. The idea behind this model is to compare each of the texts against each other, record the wins and losses of the texts, and then calculate the Elo ratings for each text. However, as you may have noticed, comparing each text to another entails a combination of numerous pairwise comparisons, which can be computationally expensive for large datasets. To mitigate this, we introduced the concept of "Nearest Neighbors" into our model. Instead of comparing each text to every other text, we only compare it to its nearest neighbors based on a cosine similarity metric. This approach significantly reduces the number of comparisons while still maintaining the integrity of the Elo rating system. By using this method, we were able to efficiently demarcate the decision boundaries of very similar text from each other's.

Here is the overall pipeline:

1- Sample a test target text from the test set.
2- Calculate the cosine similarity of the target text to all remaining data points and order them from most similar to least similar.
3- Randomly select one candidate to compare from the top-k candidates (k=5).
4- For the target text and the sampled text, ask the LLM the following prompt:

Prompt

<Instruction>Which feedback would you prefer? Choose feedback that makes you feel motivated to work harder and understood, rather than being judged. Return either Text 1 OR Text 2</Instruction>

<Task>

<Text 1>
{text1}
</Text 1>

<Text 2>
{text2}
</Text 2>

</Task>

5- Record the winning text as {id1, id2, winning id}.
6- After running this game N times (600 or more), calculate the Elo Ratings of the texts.
7- Rank the texts based on their Elo Ratings and normalize the scores using SoftMax.

Overall, we ran this method with ***gpt-4-0125-preview*** with 600 runs. The temperature was set to .40. We used **full text** to compare the feedback from each other. This method interestingly and delightfully produces very high scores on training and development runs, having an accuracy score on training ranging from .60 to .75.

4) Model 3: Knowledge Distillation : When teachers fail to teach, students fail to learn.

In this model, we aimed to use smaller models with the N-shot COTs examples produced by the stronger models. We generated the examples COTs by using a large and stronger teacher model as we defined in the reprocessing part. Then, we use these examples with a smaller but efficient model to predict target text. However, this model often fails to learn from the examples to generate significant results. We did not use this model in our final prediction. We added this code as an example of a failed attempt.

5) Model 4: Long-Context Learning with Gemini 1.5-pro | Theory of Mind

In the final model, we use long-context learning. Previous research has shown that transformer-based models are good at learning from context. Especially, Gemini 1.5 pro demonstrates a strong context learning capacity. Based on that, we fed all the training examples into the context, ordered them randomly in each iteration, and asked Gemini 1.5 pro to learn from the class decisions. We also added a twist to make the theory of mind component, so that make the model take perspective of the receiver. Interestingly, Gemini 1.5-pro demonstrated strong learning potential.

Here is our prompt for the model:

Prompt

> Please predict the target text and determine whether it belongs to class 1 or class 2. Analyze the examples, understand the reasoning behind the classification, and make your final decision for the target text. To do this, first extract the distinct points from the examples that explain why the classes are structured the way they are. When you analyze the classes, focus on the style, tone, emotive content, directness, and solution-oriented content. Also, add a "theory of mind" section where you analyze how the person who will receive these two classes of feedback might feel and be motivated. Then, develop a reasoning logic with 5 questions. Apply these 5 questions to the final target and make your final decision on whether the target text belongs to class 1 or class 2. Finally, decide the probabilities of classification into the class 1 and class 2.
> ---
> # Examples
> {examples}
> ---
> # Target Text
> {target_text}

For the final test, we generated five (t = 5) predictions with this model, randomly shuffling the order of the samples. It was quite interesting to see that random shuffling changed the probability distribution. The temperature was set to 0.30-0.40. In the training set, we observed an average accuracy of 0.51 - 0.62. We used these predictions as one of the weak predictions in the final predictions set.

6) Final Predictions

In the final predictions, we gathered the predictions from Model 1a, Model 1b, Model 2, and Model 4. For each model, we did multiple predictions by changing the order of the n-samples or slightly up and down temperatures. In line with self-consistency literature, we then ensembled the predictions and calculated the mode of the predictions. We also experimented with the means of the predictions (Majority Voting etc). We first normalized the prediction mean scores and calculated the SoftMax for the final aggregated predictions. We used a cut-off value of 0.60 to be more conservative, as we observed that many models tend to overestimate the empathic content.

Task 2: Fairness Prediction

In the fairness task, we developed a self-consistency-based N-shot learning model with GPT-4-0125-preview. The temperature was set to 0.5. This model, compared to others, was simpler and produced decent scores for us. Our prompt was:

Prompt

Question: Which policy statement is the much fairer option?
Option 1: We recommend employees start a support group to share and address workplace concerns.
Option 2: Grievance Resolution Committee: A committee addresses formal grievances and ensures a fair resolution process.
Answer: second

Question: Which policy statement is the much fairer option?
Option 1: A mentorship program pairs employees with experienced colleagues, providing guidance and preventing conflicts.
Option 2: We've implemented a complaint card system to track and address employee concerns effectively.
Answer: first

....

Question: Which policy statement is the much fairer option?
Option 1: Conflict Resolution Apps: We offer technology solutions for reporting and tracking conflicts.
Option 2: Employees can request a mediator to assist in resolving issues with their supervisor, ensuring fair dispute resolution.
Answer: **['target]'**

We generated t predictions (t=5-11) by randomly ordering the examples. In our training, we achieved an accuracy of 0.78 - 0.90. In the dev set, we observed a similar score.

Task 3: Interview Completion

In the interview sentence completion, we develop a relatively simple approach too. We generated interview sentence completion with gpt-4-0125-preview with zero-shot prompting technique. We generated n-samples (n=5 to n=10) for each interview, and we picked the sentence that has the highest cosine similarity score (we also did an experiment with Rouge-F) with the previous conversation. For the multiple prediction, we concatenate the text. We used the following prompt to generate completion sentences, with a temperature of .60 to .80.

Prompt

Objective:
In this scenario, you are engaged in a job interview where the interviewer has asked you a series of questions. Up to this point, you've responded to three questions, each time adopting a specific writing style characterized by particular sentence structures, lengths, and stylistic choices, including intentional grammatical or stylistic errors for consistency.
Task:
You are to answer the interviewer's final question. Your primary task is to analyze the writing style reflected in your previous responses, including sentence structure, length, common mistakes, and any identifiable personality traits or background information that can be inferred. This analysis will form the basis for crafting your response to the final question, ensuring that it is stylistically consistent with your earlier answers.
Conversation Extract:
```
{conversation}
```
Final Question:
```
{final_question}
```
Instructions:
Analyze the Writing Style: Review the sentence structure, length, and common errors in the earlier responses. Take note of any personality traits or background information that can be inferred from the responses.
Mimic the Style: Craft your response to the interviewer's final question by mimicking the writing style used in your previous answers. This includes maintaining similar sentence structures, lengths, and reproducing any grammatical or stylistic quirks.
Focus on the Question: Your response should directly address the interviewer's final question without deviating into unrelated context or explanations.
Consistency is Key: Ensure that the response length and level of detail are consistent with your previous answers, reflecting a uniform approach in presentation and substance.
Incorporate Unique Quirks: If your previous responses included specific writing quirks or mistakes, incorporate these elements into your final response to maintain a cohesive and consistent writing style throughout the conversation.

Important: Only write the deliverable in the format `Deliverable: `. Do not include any other formatting or any background information. Do not explain your thought process.
Deliverable:
A written response to the interviewer's final question that is stylistically consistent with the earlier parts of the conversation, maintaining the same unique writing characteristics and errors for a coherent presentation throughout the interview scenario.

This is one of our most reliable methods. We get a steady interview score of about .50 to .53 in the train, dev, and test sets.

Task 4: Clarity Prediction

This task was another challenging task for our group. In our training data, we got a very high score. In the dev set, we also saw decent scores. But in the test set, our score was not at the point where we wanted to be.

In this task, we developed two step models. Here are the details.

1) Model 1: Rule Extraction and Developing Expert Ratings

Using the Meta-Prompting technique, we first identify patterns in the datasets. We provide the Opus model with all the training datasets along with their scores and ask for the reasons why some texts receive high or low clarity scores. Based on this, we identify series of rules that influence an item's clarity score. For example, an item phrased with a double negative is likely to receive a lower score, and the use of passive voice also leads to a lower score etc.

Meta Prompt

You will be analyzing a set of survey item texts and their corresponding clarity scores to identify patterns that distinguish items that received high clarity scores from those that received low scores.

Here are the survey item texts:
<survey_item_and_clarity_scores>
{survey_items}
</survey_item_and_clarity_scores>

Please carefully review the survey item texts and their scores. Look for characteristics and patterns that tend to result in high clarity scores vs low clarity scores. Focus on things like:
- Word choice and vocabulary level
- Sentence structure and complexity
- Specificity and concreteness of language
- Logical flow and organization of ideas
- Grammatical correctness
- Appropriate length and level of detail
- Avoidance of jargon or insider terminology
- Clarity of action words and key terms
- Use of examples or analogies to explain concepts
- Formatting and visual presentation
- Specificity and clarity in the languge used

Aim to identify at least 10 or more distinct reasons that seem to explain why certain items are rated as more or less clear. For each reason you identify, provide a brief explanation and cite 1-2 examples similar to provided text, but not the same as the examples and provide their clarity scores. Provide both low and high clarity examples for each reason.

Present your findings in a <reasons> section, with each reason contained in its own <reason> tag like this:
<reasons><reason>
Reason 1: [explanation]
[example 1]
[example 2]
</reason><reason>
Reason 2: [explanation]
[example 1]
</reason>
...
</reasons>
Focus on the most important and impactful reasons. Avoid redundancy between reasons.
Let me know if you need any clarification or have questions!

After extracting these rules, we developed a rule-based Large Language Model (LLM) system to generate binary scores for each item. For a given target text, we pose a series of yes/no questions independently. The scores from all these questions are then summed up to generate an expert rating for use in downstream tasks.

We repeated this process t-times with different LLM models, i.e., gpt-3.5-turbo, sonnet, opus, gpt-4, etc. We recorded these scores as ***'expert' ratings***, labeled ***'expert 1'*** and '***expert 2***', ... '***expert n***'

2) Model 2: Learning from Expert Ratings

In the final stage of this pipeline, we used the expert ratings with a final stronger LLM model. We prompted the model as:

Prompt

This is an expert rating task to assess the clarity of the written survey item. As an expert in the field, please rate the clarity of the survey item. The clarity score ranges from 1 to 5, where 1 is the least clear and 5 is the clearest text.
Before judging, please examine why the other experts might rate this item as they did. Remember that the expert might be 'wrong.'
You MUST develop step-by-step reasoning under the <reasoning> tag and articulate your intuition tag.
Finally, give your ratings a with a float point with <rating></rating> tag.

Items might receive higher clarity rating if they...
- Use simple, concise language and sentence structure. Avoid unnecessary complexity or wordiness.
- Minimize the use of negative wording and double negatives.

- Express a single, focused idea or action in each item. Don't combine multiple concepts or make items too broad.
- Choose concrete, specific words and phrases over vague or abstract language. Be direct and to the point.
- Do not use awkward phrasing or grammatical errors can make an item confusing and harder to answer accurately.
- Use active sentence voice structure not passive voice.
- Do not use any vague qualifier, such as `average sort of` etc.

Here is the survey item you have to rate.
<text>
{target_text}
</text>

Here is experts ratings:
<expert_ratings>
{expert_ratings }
</expert_ratings>

Although in the train and dev set, we hit .80. In the test set, our score was lower than expected. Still, we were in the range of .60 to .70.