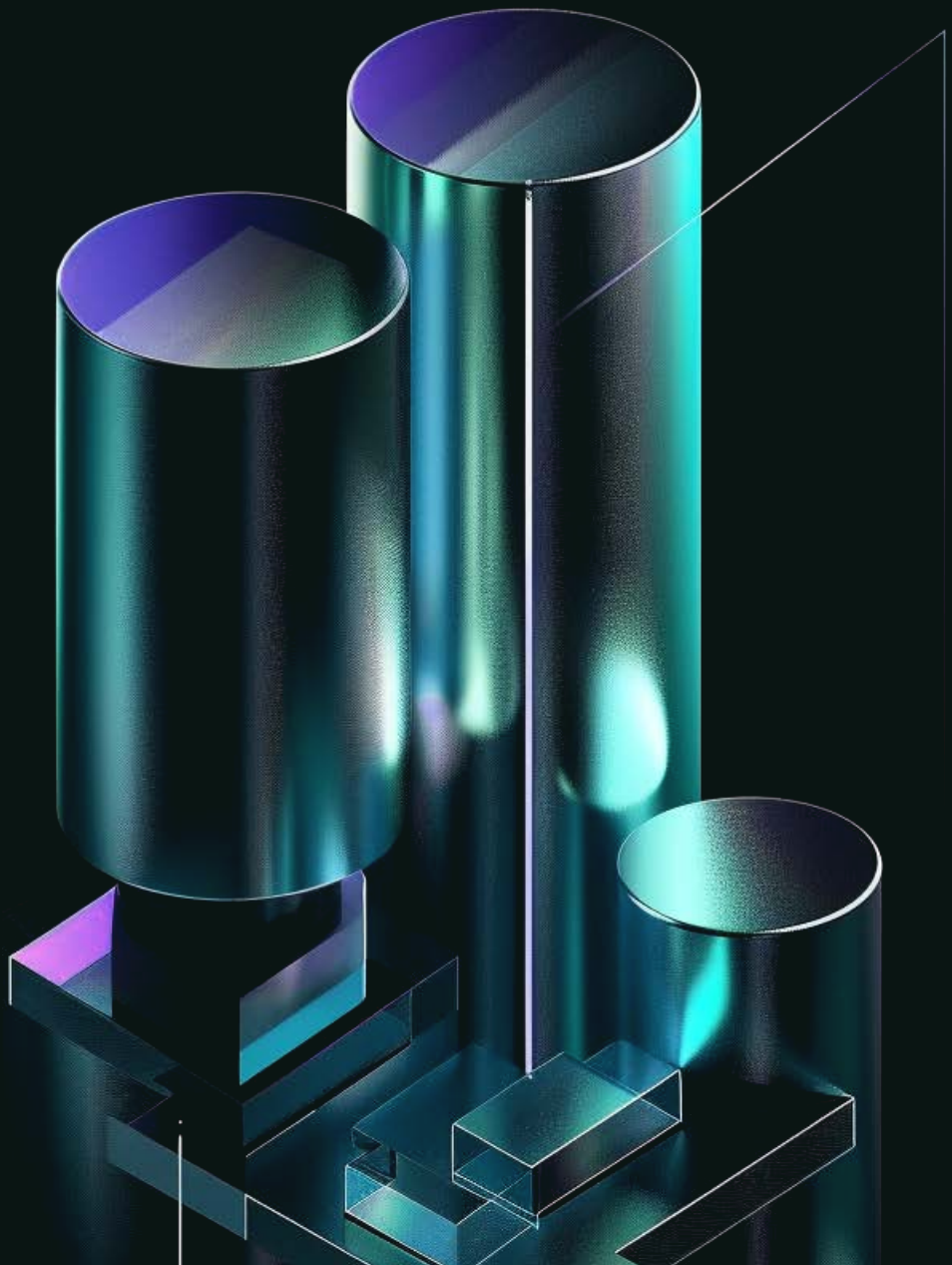
How to Evaluate Conversational AI for Politeness

A Framework for Measuring Attributes of Conversation

GIOVANI TAVARES CAROLINA PEREIRA
RAFAEL MEDEIROS RENATO VICENTE



May 2024

ai@willowtreeapps.com
1-888-329-9875
Boston | Charlottesville | Columbus | Durham
Lisbon | Porto Alegre | São Paulo | Vancouver

WILLOWTREE®
a TELUS International Company

Table of Contents

1. Introduction

Testing and evaluating for conversational attributes – politeness, empathy, helpfulness, compassion, attentiveness – are critical steps in developing and deploying conversational AI systems, both for human and machine-generated responses. Businesses that understand how these conversational attributes positively impact customer satisfaction, retention, and loyalty have long monitored them as standard practice [1], [2], [3].

It makes sense then that the more businesses turn to generative AI, the more they need a framework to evaluate responses for different conversational attributes. That way, they can monitor and fine-tune their generative AI systems to have more engaging, satisfying conversations with customers.

Such a framework makes even more sense when we consider how subtle changes in prompts can substantially influence the behavior and response generated by large language models (LLMs). That makes prompt evaluation crucial in evaluating conversational attributes [4].

The Authors


Giovani Tavares


Carolina Pereira


Rafael Medeiros


Renato Vicente

Connect with WillowTree's Data and AI Research Team (DART):

Michelle Avery | Group VP, AI
WillowTree, a TELUS International Company
michelle.avery@willowtreeapps.com


Let's connect

This paper proposes a framework for evaluating conversational attributes such as politeness, empathy, helpfulness, and compassion. This framework spans three parts:

1. **Attribute classifier or conversation evaluator**: A text classifier that generates a score for the conversation. The main requirement is that it be able to evaluate a conversation regardless of its source (i.e., human or machine-generated).
2. **Test dataset**: We present a strategy to construct and label a test dataset to evaluate the effectiveness of the proposed evaluator.
3. **Prompts dataset**: We detail a strategy for constructing a dataset of user messages that allows measurement of the target attribute on LLMs.

We present the results of testing this framework, starting with focusing our attribute classifier on politeness over other dimensions of conversation. The present research is focused on politeness because this is a crucial attribute for successful customer service. Politeness also had open data available for research and was relatively easy to define on a linear scale [5].

Section 1.1 offers a deeper understanding of politeness, and Section 2 introduces the attribute classifiers we tested.

1.1. Motivation for Politeness

In defining a process for evaluating conversation attributes, we wanted to focus on a specific attribute that 1) represented a relevant, complex feature in human-to-chatbot assistant dialogue utterances and 2) had a syntactic dimension among its multi-dimensional definition. This approach allows the chosen feature to be abstracted and substituted with others.

Moreover, the presence of a syntactic dimension in its definition makes it possible to analyze how AI models detect that syntactic dimension in a context-free environment (i.e., with utterances only). Therefore, this type of feature, despite being complex, can be grasped in conversation pieces without the need to know who's speaking, for example.

With this in mind, politeness is 1) challenging to define because of its multidimensional manifestation rather than just the syntactic one and 2) it's significantly relevant for monitoring human-machine interactions based on natural language conversations.

The fact that politeness has lexical, syntactic, pragmatic, sociocultural, nonverbal, and kinesthetic dimensions [6] makes it an exciting feature to track because it can assess how different AI models perform in tackling a feature with such a complex definition – and how they compare to humans when doing so. Therefore, we decided to use politeness as the attribute to be evaluated in the established framework,

even though you could use another attribute that, similar to politeness, contains multiple dimensions including syntactic ones.

Moreover, the availability of a well-documented and publicly accessible dataset annotated for politeness further solidified our decision to prioritize politeness as a critical feature for evaluation within our framework.

2. Attribute Classifiers: Three Methodologies We Tested

The primary objective of our study is to establish a reliable way to measure particular attributes of conversations, which are often inherently nuanced and complex. In particular, we are interested in conversational attributes that we can establish an ordinal scale to measure. For example, a politeness attribute may range from "impolite" to "neutral" to "polite." Likewise, a helpfulness scale may range from "uncooperative" to "neutral" to "helpful."

One big challenge lies in the fluid nature of these attributes, which are often hard to define and measure precisely. However, we hope that current state-of-the-art (SOTA) LLMs and embedding models can capture these nuances and act as classifiers for these attributes. Combined with a sufficient amount of annotated data, using current LLMs should allow us to build algorithms that perform text classification consistently and domain-independently.

To tackle this problem, the first part of our study focused on creating a classifier for conversation attributes. Its purpose is to measure the desired attribute on a predefined scale reliably. To achieve this, we explored three different classifier strategies:

- **2.1. Embeddings Classifier** – A sentence-embedding classifier
- **2.2. LLM Prompt Classifier** – A few-shot in-context learning prompt classifier
- **2.3. Fine-Tuned LLM Classifier** – An LLM fine-tuned for the classification of a specific conversational attribute (e.g., politeness)

Each approach has distinct requirements and particularities, making it worth unfolding these strategies in detail.

2.1. Embeddings Classifier

For our first approach to creating a conversation attribute classifier, we experimented with sentence embeddings to measure and evaluate desired conversational attributes, as shown in Figure 1 below.

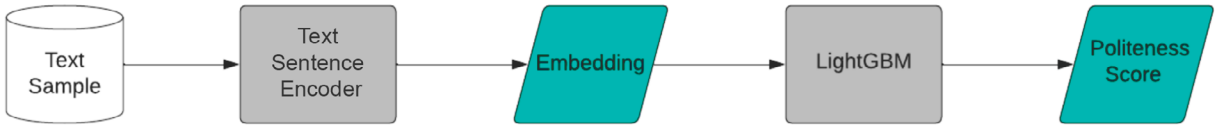


Figure 1. Overview of the Embeddings Classifier. Grey boxes indicate processing stages and teal boxes indicate generated data.

We utilized OpenAI's text-embedding model ADA-002 ('text-embedding-ada-002') [7]. The process begins by employing this model to generate embedding vectors for whole sentences (samples) in our dataset. Sentence embeddings are a representation of the semantic content of a sentence, mapping it to a numerical vector that's unique for each sentence. As such, the dimensions of the latent space relate to semantic interpretations of the text.

Because these embedding vectors are high-dimensional and because of the non-linear relations between them, a direct interpretation of these dimensions is usually impossible. However, it's still possible to use machine learning methods to interpret these sentence embeddings. Therefore, we use these sentence embeddings as the input for a machine-learning model that classifies the samples.

In this work, we chose the gradient boosting method from the LightGBM framework [8]. It uses decision trees as the base models for the boosting strategy. This means that LightGBM will combine multiple decision trees — weak prediction models that may not be very accurate on their own — into a larger model that benefits from their individual strengths. With this strategy, the final ensemble model produces classifications of higher quality and higher precision [9]. This strategy is also known as gradient-boosting decision trees (GBDT).

It's interesting to note that GBDTs can produce either categorical or numeric predictions (by using regression trees). Since the conversational attributes we're interested in can be evaluated in an ordered linear scale (e.g., negative, neutral, positive), generating a numeric prediction is more advantageous since it allows a more nuanced evaluation of the samples. Therefore, we set regression as the training objective for our model. When evaluating this model output, note the prediction rounds to conform to the ground truth labels.

2.2. LLM Prompt Classifier

For our next strategy, we employed an in-context learning classifier. We instructed a foundation LLM to function as a politeness classification assistant, as illustrated in Figure 2 below.

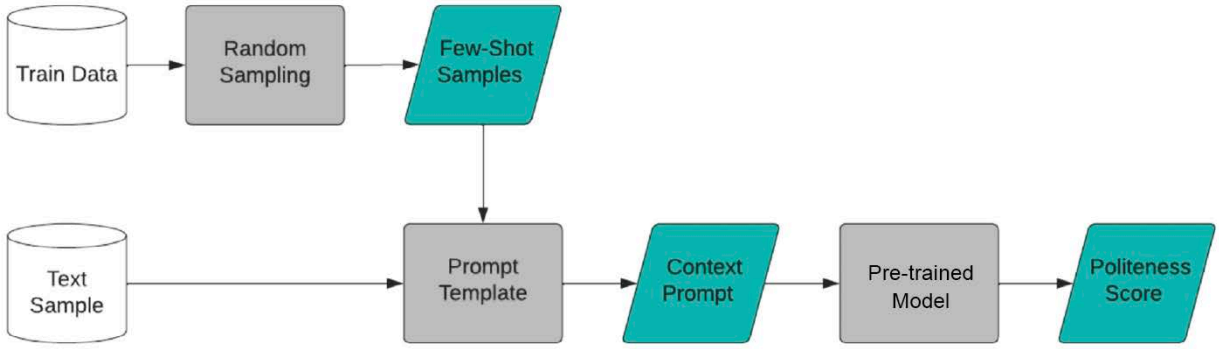


Figure 2. Overview of the in-context learning few-shot approach. Grey boxes indicate processing stages and teal boxes indicate generated data.

In-context learning allows us to leverage prior knowledge and comprehension that the models learned about general linguistics [10]. With SOTA models being huge and having access to massive amounts of training [10] data, the expectation is that no retraining or fine-tuning would be necessary to execute new tasks.

This strategy is comparable to specialist models — and sometimes outperforms them — by providing only a well-designed set of instructions in the prompt context. Unlike the previous classifier approach, this strategy gears towards multiclass classification. The models handle politeness levels outlined in the prompts because foundation models are not well suited for continuous regression analysis.

This approach presented two main challenges. The first one was to generate accurate predictions using the model, as with the previous approach. To achieve this, we described the five different politeness levels in the system prompt. The aim here was to instruct the language model about the scale of politeness explicitly it should be able to recognize and classify.

Therefore, we used the same guidelines provided for human reviewers when constructing the test dataset (see Section 3.3.1). In the case of our politeness classifier experiments, we use the following system prompt:

```
You are a Politeness Classification assistant that classifies text into
single categories.
You'll be given only JSON objects with two keys: "text" and "category".
The 'text' key's value is some sentence found in online forums.
The 'category' key's value is a string 'MASK'.
Your task is to respond concisely with another JSON, substituting 'MASK'
by the classification of the 'text' key's value.

The category options go from 1 to 5, having the following meaning:
1: Very impolite - Rude and offensive language or behavior. Disregards
social norms and respect for others.
2: Somewhat impolite - Lack of consideration for others, with occasional
usage of more offensive language or behavior.
3: Neutral - A standard level of politeness, adhering to basic social
norms and showing basic respect.
4: Somewhat polite - Demonstrates extra consideration for others, using
polite language and gestures more consistently.
5: Very polite - Exceptionally considerate and respectful, going above
and beyond to make others feel comfortable and valued. Consistently uses
polite language and mannerisms.
```

The second challenge was ensuring the model output was automatically parseable. We desired the model output to be a JSON object with a few specific fields. We included both an output description in the prompt as well as a set of N examples of input/output pairs from the training data in the following format:

```
USER:  { "text": "{{sentence_1}}", "category": MASK }
ASSISTANT: { "text": "{{sentence_1}}", "category": {{label_1}} }

USER:  { "text": "{{sentence_2}}", "category": MASK }
ASSISTANT: { "text": "{{sentence_2}}", "category": {{label_1}} }

...

USER:  { "text": "{{sentence_N}}", "category": MASK }
ASSISTANT: { "text": "{{sentence_N}}", "category": {{label_1}} }

USER:  { "text": "{{test_sentence}}", "category": MASK }
```

Note {{sentence_i}} is the i-th few-shot example, and {{label_i}} is its ground truth. These prompts simulate an ongoing conversation between the user and the chat model (the "assistant" role) using the desired JSON format, which helps the LLM to reproduce the desired format. Following the simulated exchange, the final user message contains the actual conversation text to be classified (the {{test_sentence}}). Therefore, the final text generation request uses the following format:

```
response = client.chat.completions.create(
    model=deployment,
    messages=[
        {"role": "system",       "content": "{{system_prompt}}"},
        {"role": "user",         "content": "{{example_query_1}}"},
        {"role": "assistant",    "content": "{{example_output_1}}"},
        {"role": "user",         "content": "{{example_query_2}}"},
        {"role": "assistant",    "content": "{{example_output_2}}"},
       ...
        {"role": "user",         "content": "{{example_query_n}}"},
        {"role": "assistant",    "content": "{{example_output_n}}"},
        {"role": "user",         "content": "{\"text\": \"{{test_
sentence}}\", \"category\": MASK}","},
    ],
    temperature=0,
)
```

These few-shot examples were selected randomly from the training data during the prompt optimization. We iterated multiple random combinations to search for the subset with the best average classification performance. Once we found the best few-shot approach, we kept the same set of examples for all future evaluations.

During this optimization phase, the pre-trained LLM temperature is set to zero to reduce the response randomness, increasing the predictions' consistency. Increased consistency speeds up the prompt optimization since there are fewer combinations of prompt and hyperparameters to evaluate.

2.3. Fine-Tuned LLM Classifier

The third and final classifier strategy we explored was fine-tuning pre-trained LLMs specifically for conversation attribute classification, as shown in Figure 3 below. This Fine-Tuned LLM Classifier approach relies on using new training data from a specific domain to fit the parameters of pre-trained models to work more efficiently and effectively on particular tasks, thus enhancing their predictive capability.

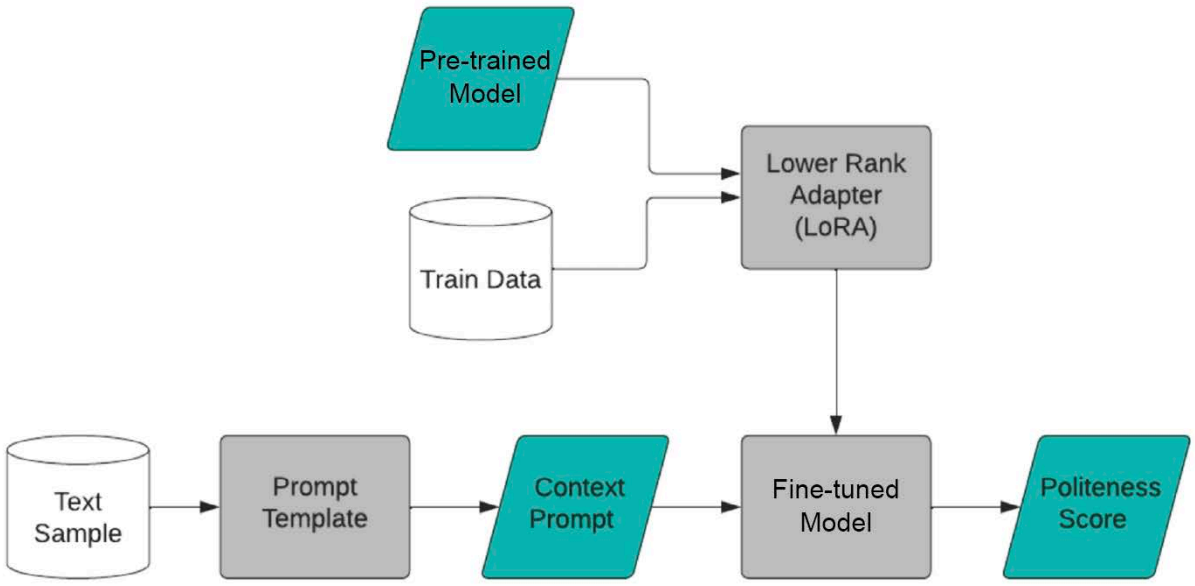


Figure 3. Overview of the in-context learning fine-tuned approach. Grey boxes indicate processing stages, and teal boxes indicate generated data.

In terms of implementation, we employed Low-Rank Adaptation (LoRA) [11], a technique for fine-tuning transformers that 1) freezes all parameters in a pre-trained neural network and 2) adds a small number of extra parameters to a few specific layers. Only these new parameters are fine-tuned to the new data. This enables faster (and often better) transfer learning when compared to traditional fine-tuning, where all weights are adjusted. Also, this approach is constructive because it allows us to repurpose models for specific tasks with a much smaller training corpus and computational resources than training from scratch or fine-tuning the whole model.

Like the previous approach, we prompt the language model to classify a target sentence on a scale from 1 to 5, with some guidance on what's expected from each attribute level. For our experiments on measuring conversational politeness, we used the following prompt:

```
### Task:
Please rate the politeness of the following sentence on a scale of 1 to
5, having the following meaning:
1: Very impolite - Rude and offensive language or behavior. Disregards
social norms and respect for others.
2: Somewhat impolite - Lack of consideration for others, with occasional
usage of more offensive language or behavior.
3: Neutral - A standard level of politeness, adhering to basic social
norms and showing basic respect.
4: Somewhat polite - Demonstrates extra consideration for others, using
polite language and gestures more consistently.
5: Very polite - Exceptionally considerate and respectful, going above
and beyond to make others feel comfortable and valued. Consistently uses
polite language and mannerisms.

Return only a number between 1 and 5, without any additional text.

### Sentence: "{{sentence}}"

### Classification: {{classification}}
```

Note {{sentence}} is the conversation text to be classified, and {{classification}} is a number from 1 to 5, as defined in the politeness classification scale. We didn't use few-shot examples here because the fine-tuned model generated responses in the desired format.

3. Experiments and Results

3.1. Training the Attribute Classifiers

Having established our general methodology for training the attribute classifiers, we applied it to the conversation attribute of politeness. The dataset used to train the classifiers was Stanford's politeness annotated dataset [5]. It contains two subsets of forum messages: one from Wikipedia and one from Stack Exchange, both annotated

for politeness by humans. In this work, we used the Wikipedia subset, dividing it into three parts:

- training (3,046 samples)
- validation (654 samples)
- testing (350 samples)

We trained the Embeddings Classifier using the embedded training set as inputs to a LightGBM model. We also used the embedded validation set as early stopping criteria. The training objective was set as regression. Predictions converted to integers in the 1 to 5 range. Our final model was trained using the following hyperparameters optimized through grid search:

- learning rate: 0.1
- number of estimators: 150
- number of leaves: 10

We tested multiple prompt variations for our prompt engineering approach (see the previous section for the details of which performed best). The few-shot examples came from the training set, and the temperature stayed constant at zero. Tests with higher temperature values were unsuccessful because the model tended to either classify sentences out of the defined scale or change the desired format of the output. The models used in this step were GPT-4 (version 0613) and GPT-3.5 Turbo (version 0613).

The last approach we tackled was fine-tuning smaller LLMs to act as classifiers for politeness. For this step, we used the training and validation datasets, formatted using the prompt described in the previous section. We used the transformers library and the LoRA approach for fine-tuning with the following training hyperparameters: LoRA rank=16, α=64. The model was trained for 150 steps with a batch size of 32 and learning rate of 2.5e-05.

We fine-tuned all available linear layers in both Mistral-7b-Instruct and Llama2-7b-Chat. After training, we generated the classification using the text generation pipeline from the transformers library, allowing only one token to generate and using this single token as the classification.

To test the performance of the implemented politeness classifiers, we used two different approaches: in-domain tests and out-of-domain-tests.

3.2. In-Domain Test

Our first approach, in-domain testing, refers to tests performed using data from the same domain the evaluators were trained on. A language domain can be defined as a set of specialized text that includes terms and syntactic constructions commonly used in that specialization field. Hence, we refer to this approach as "in-domain" tests.

Since the evaluators were trained in the Wikipedia subset from Stanford's politeness dataset [1], the dataset used in our in-domain tests contained sentences from its Wikipedia Forum subset (i.e., its terms and syntax structures were not that different from those in the training dataset). Thus, it's expected the evaluators would perform satisfactorily in the same-domain tests.

At first, we evaluated models based on three key performance metrics: Precision, Recall, and F1 (see Table 1 below). The values reported are averages of metrics for the five classes. The model gpt4-5shot shows the highest Precision of 0.484 while the highest Recall of 0.44 belongs to mistral-7b-instruct-finetuned. The fine-tuned Mistral also offers a strong balance between Precision and Recall, scoring 0.423 – the highest F1 score.

Since these are classification metrics, they consider only if the exact class was hit or missed, not by how far it was from the correct answer given the ordinal labels.

Model	Precision	Recall	F1
mistral-7b-instruct-finetuned	**0.454**	**0.440**	**0.423**
gpt4-10shot	0.439	0.351	**0.336**
llama2-7b-chat-finetuned	0.340	**0.400**	0.324
gpt4-5shot	**0.484**	0.320	0.312
lgbm_with_embeddings	0.427	0.306	0.294
gpt-3.5-turbo-10shot	0.323	0.317	0.282
gpt4-0shot	0.285	0.214	0.187

Table 1. Classification metrics for in-domain tests. All metrics are averages of the five classes. **Teal** *and* **Bolded** *values are best and second best, respectively.*

In evaluating politeness classifiers, we defined a metric called "Adjacent Accuracy." This metric is conceptualized as a variant of traditional accuracy, allowing for a prediction to be considered correct if it falls within a window of distance one from the true label. This approach acknowledges the inherent ordered nature of our classification categories despite the methodological treatment of the problem as a classification task.

The justification for introducing Adjacent Accuracy lies in recognizing the politeness spectrum: **A misclassification neighboring the true category is a more favorable outcome compared to a prediction far from the target, especially considering the nuanced gradation of politeness**. This metric, therefore, affords additional insights into the performance of our models, highlighting those that more closely approximate the correct politeness measure, even if not pinpoint accurate.

Considering this new metric, our experiments revealed notable performances by the Embeddings Classifier and the fine-tuned Mistral-7b-Instruct. Both exhibited results close to the GPT-4 few-shot prompting specifically for the Wikipedia corpus dataset. As illustrated in Figure 4's bar graph below, Mistral-7b-Instruct's Adjacent Accuracy was the highest among the fine-tuned models tested with a score of 0.806, closely following LGBM with 0.820.

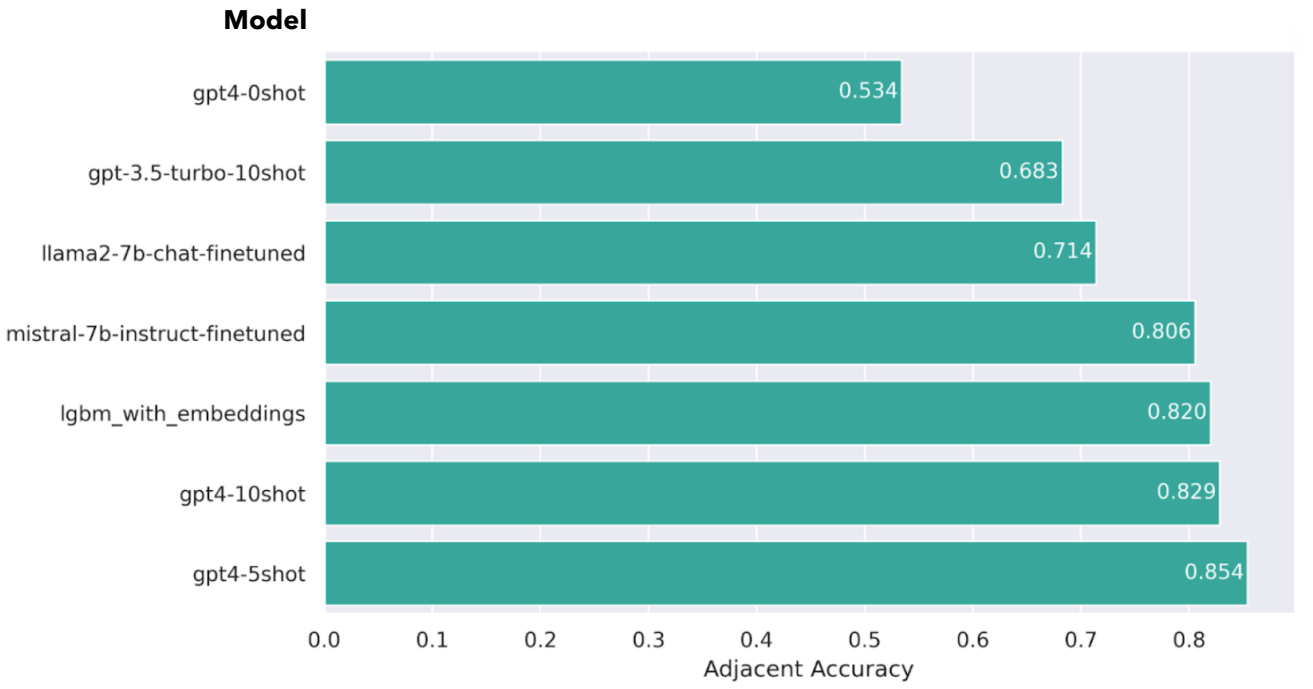


Figure 4. Adjacent Accuracy for models in the in-domain dataset.

The graph demonstrates a clear performance gradient with the trained models surpassing some of the prompt classifiers, including the base GPT-4 with zero-shot prompting, which achieved 0.534 for its Adjacent Accuracy.

However, including additional examples through few-shot prompting enhances GPT-4's performance significantly, conforming to the expected output format as evidenced by the increased Adjacent Accuracy scores for both the 5-shot and 10-shot scenarios.

3.3. Out-of-Domain Test

The second approach to testing the classifiers was based on using data from language domains different from those of the training dataset. In this document paper, this specific approach will henceforward be referenced as "out-of-domain" tests. To do so, we gathered selected data from different sources into a single dataset with selection criteria further defined.

Testing the evaluators on domains different from what they trained in meant we could 1) validate their robustness and 2) measure their general understanding of politeness. This way, we could measure the results of their usability as conversation supervisors in different contexts.

To do this, we used publicly available data on instruction-following chatbot assistants available through the Hugging Face Hub datasets platform:

- Meta-Learning Wizard of Oz (MetaLWOz) [12]
- AlpacaFarm Evaluation (AlpacaFarm) [13]
- Human ChatGPT Comparison Corpus (HC3) [14]

We imposed three main requirements to select the data from MetaLWoz and AlpacaFarm. The first requirement, "assistant utterance," relates to the assistant-like role feature the data had to contain. Hence, the "assistant utterance" requirement discarded sentences like "Hello, I need some help" – which mimics a user introducing a conversation with an assistant rather than the latter responding to the former.

The second requirement, called "instruction-following," imposes that the utterances were responses to instructions (i.e., the "assistant utterance"-like data had to look like a chatbot assistant responding to a user looking for guidance). For instance, sentences like "Hello, how can I help you?" were discarded, while others like "Sure, I can definitely help you with that. To do so, simply [INSTRUCTIONS]" were approved. Combining both criteria, "assistant utterance" and "instruction-following," ensured that the out-of-domain tests used data from which the results would be insightful – such data represented chatbot assistant response to users seeking help.

We imposed the third and final requirement at the dataset level rather than single

conversation samples. After selecting utterances that looked like chatbot assistant responses, we filtered the dataset to locally maximize its **lexical diversity** as a means to approximate its global maximum. Such a metric is defined below. This provision guaranteed that the out-of-domain dataset used to test the evaluators contained a few similar utterances pairs like "Sure! I can definitely help you with your appointments" and "Sure! I can definitely help you with some appointments." **Avoiding such pairs when testing the evaluators reduces the biases in their performance results because it's a way to increase the internal relative representativeness of the data in the dataset. We call this criterion "lexical diversity."**

3.3.1. Lexical Diversity

$$\text{Lexical Diversity (Sentences Dataset)} = \frac{\text{Number of Unique Words (Sentences Dataset)}}{\text{Total Number of Words (Sentences Dataset)}}$$

MetaLWoz contains conversations between two humans, one of whom pretends to be a chatbot assistant responding to some user. This is the "Wizard of Oz" reference — using people behind the curtain who act as the machine. The samples are divided into domains that map each conversation to whatever topic the user's requests relate to (e.g., banks, apartment finding, etc). This way, selecting utterances that looked like a chatbot responding to user requests from MetaLWoz was straightforward because conversation samples were indexed by alternating user and assistant turns — the process to do so was simply filtering the data using the assistant utterances indexes. AlpacaFarm's evaluation split supports the development of instruction following models, and it contains sentences that look like assistant responses to user instructions. No sampling was needed to get assistant responses only.

As previously mentioned, the MetaLWoz corpus contains conversations covering different domains, and the user utterances are not necessarily requests. Ergo, the chatbot utterances are not necessarily instruction-following ones. Every chatbot utterance after the first was disrated because in a human-assistant alternating conversation that the assistant begins, the first message is usually a greeting. On the other hand, in domains containing user requests, the first user utterance usually already contains a request. So, the assistant's response will likely be an instruction-following one.

We needed to ensure the data from MetaLWoz was instruction-following-like, so we selected user requests and then calculated the term frequency-inverse document (TF-IDF) statistic for each domain. This measure helps us understand the terms/words/expressions that are the most informative/frequent in identifying the domain/category of a document made of strings.

In our case, such documents are each a set of sentence samples categorized by their domains. With that value in hand, the goal was to filter out domains/sets of documents from which there were no "instruction-like" terms among the top 10 most informative ones. We used the **Scikit-Learn's Feature Extraction module**'s [15] [16] implementation of TF-IDF in the `TfidfVectorizer` class.

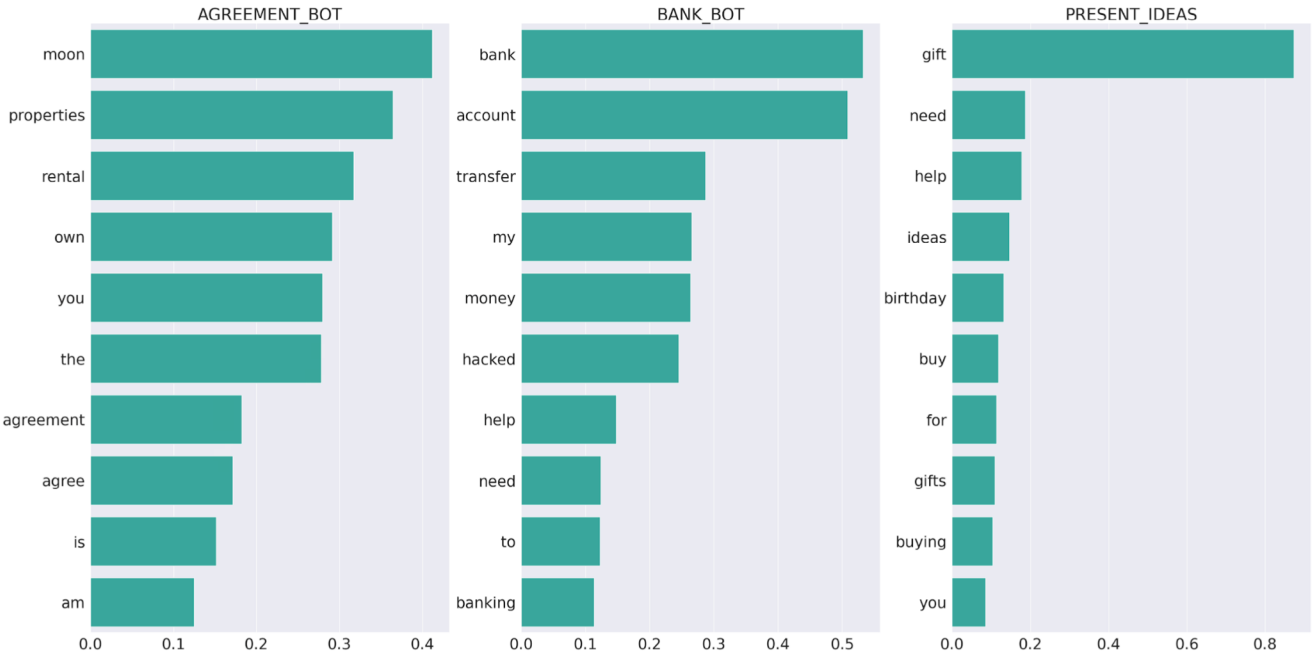


Figure 5. Top 10 TF-IDFs for some MetaLWoz domains. The plot titles represent the domain names, the y-axis the terms, and the x-axis the frequency of the terms.

By analyzing plots like the one in Figure 5 above for each domain, it was possible to tell that only the MetaLWoz' AGREEMENT_BOT domain's samples were not instruction-like. Thus, all of the assistant responses were filtered out. As observed above, this domain's most relevant terms don't look like instruction ones as expected — in an "AGREEMENT_BOT" domain, a user usually expects the assistant's agreement and usually doesn't request any instruction. This filtering eliminated 37.4% of MetaLWoz samples. The other domains remained because instruction-like words were relevant to them. Some

examples are terms like “can,” “you,” “help,” and “need.” As previously described, AlpacaFarm’s evaluation set was designed to contain instructions. Therefore, its samples contain instruction-following examples, and no further filtering was necessary.

We observed that the lexical diversity of the assistant’s utterances in both MetaLWoz and AlpacaFarm were small – 13% and 36%, respectively. This indicates that such utterances were similar in each of these datasets. To avoid this problem, we kept only a subset of sentences in each domain that were internally lexically diverse – the top 100 and top 50 sentences regarding their internal lexical diversity were kept for MetaLWoz and AlpacaFarm, respectively. We made this decision because we were interested in a set of sentences that were lexically diverse as a whole. So, a simple way to do that was to ensure that none of the sentences repeated too many of the words internally.

This greedy technique resulted in MetaLWoz’s and AlpacaFarm’s instruction-following-like subsets achieving lexical diversities of 31% and 57%, respectively. Still not ideal, but they improved in the further human review of the final out-of-domain test dataset (see Figure 6 below).

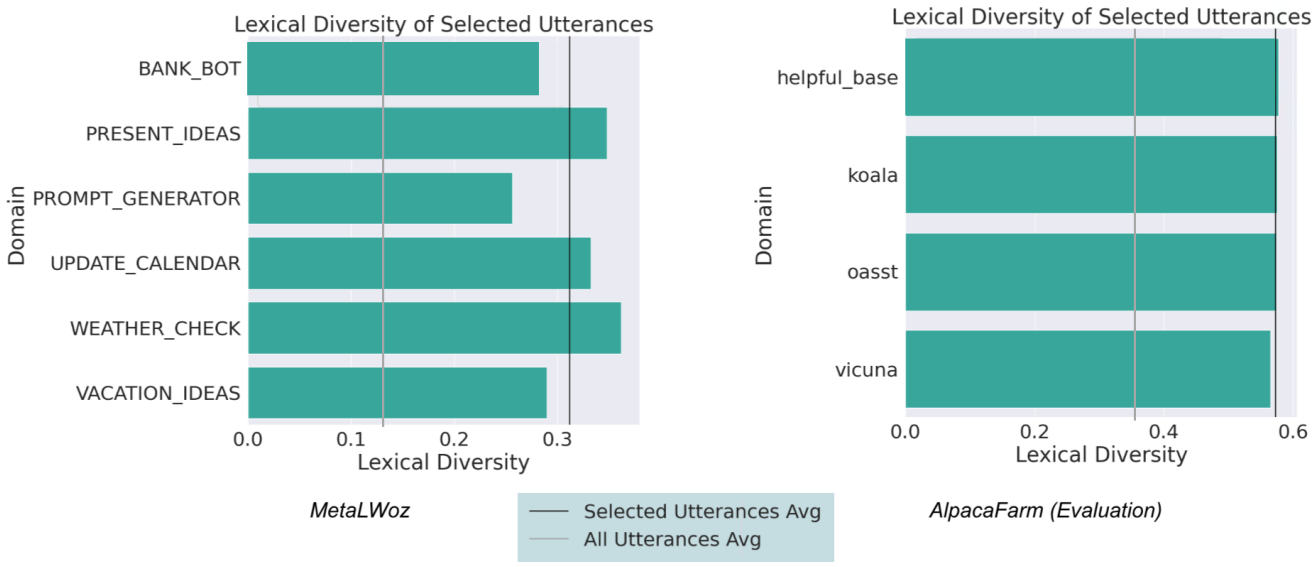


Figure 6. Lexical diversity for some domains of MetaLWoz and AlpacaFarm (Evaluation).

We used two criteria to select the samples from HC3 for the out-of-domain test dataset. The first relates to the length of the ChatGPT and human answers strings, which needed at least five words but less than three hundred. When selecting the samples that met such criteria, we randomly selected 20 questions and their answers (i.e., AI assistant and

human) from each source. This process resulted in samples equally distributed among HC3's sources/domains (e.g., finance, medicine, question-answering, reddit explain-like-I'm-five questions, computer science).

The resulting instructions dataset annotated for politeness had 1077 instruction-following-like chatbot assistant utterances selected from MetaLWoz, AlpacaFarm (Evaluation), and HC3 datasets. This data was then fed to a GPT-4 zero-shot politeness classifier to be annotated for politeness. (Note this is not the LLM Prompt Classifier previously described in **section 2.2.**) The system prompt was:

```
"You are a Politeness Classification assistant that classifies sentences
according to its politeness. You'll be given JSON-like strings with two
keys: 'text' and 'category'. The 'text' key's value is the sentence you
must classify correctly and carefully. The 'category' key's value is the
string 'MASK'. You must respond concisely with another JSON-like string
with the exact same keys and only them, but you must replace the value
of the 'category' key with a name that tells the politeness category of
the text' key's value. You must user a linear politeness scale and the
category options are 1, 2, 3, 4 and 5, where 1 is very impolite and 5 is
very polite"
```

The resulting out-of-domain dataset's annotations were then sent to human reviewers, and two humans reviewed each sentence. Their task was to validate the zero-shot classifier's annotations and discard irrelevant and repetitive samples, resulting in 431 samples.

If the reviewers disagreed on the politeness score for some sample, we sent it to a third reviewer. The resulting politeness classification score averaged the three reviews.

The resulting human-reviewed dataset was imbalanced, which we solved by using GPT-4 as a zero-shot data augmenter fed with the politeness-neutral samples. Below is the system prompt used in this process:

```
"Your task is to generate different versions of the same sentence. The
original sentence is an answer to a question, you will be provided with
the question for context.
The original sentence is neutral in politeness, follow the definition of
politeness levels below to generate different versions of the original
sentence.
```

```
Impolite - Rude and offensive language or behavior. Disregards social
norms and respect for others.
Slightly Impolite - Lack of consideration for others, with occasional
usage of more offensive language or behavior.
Neutral - A standard level of politeness, adhering to basic social norms
and showing basic respect.
Slightly Polite - Demonstrates extra consideration for others, using
polite language and gestures more consistently.
Polite - Exceptionally considerate and respectful, going above and beyond
to make others feel comfortable and valued. Consistently uses polite
language and mannerisms.

Return a JSON object with the keys 'impolite', 'slightly_impolite',
'slightly_polite' and 'polite'".
```

The final balanced for politeness scores out-of-domain dataset was made up of 431 and 593 original and synthetic sentences, respectively, resulting in 1,024 out of domain samples that were further used to compare the performance of the politeness classifiers.

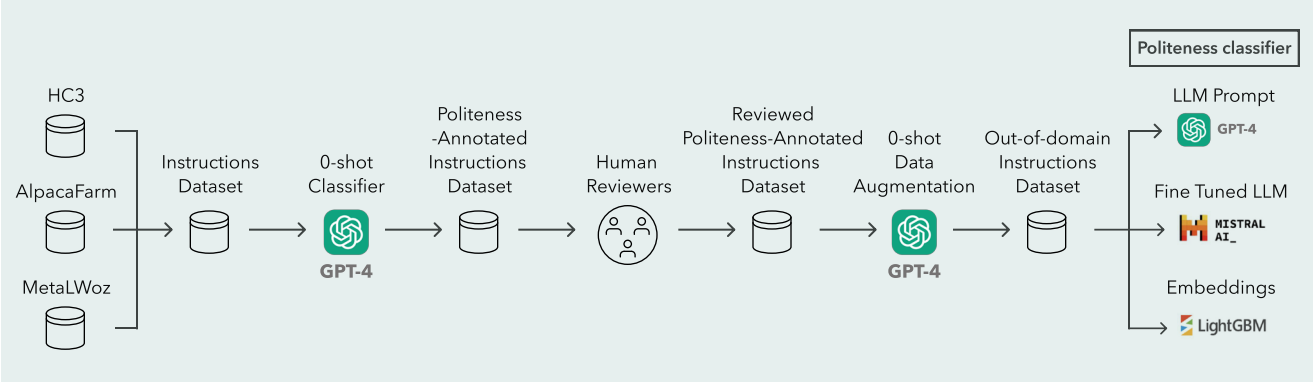


Figure 7. Out-of-domain tests pipeline.

3.3.2. Results

The second round of testing for the candidate classifiers used the out-of-domain dataset described in the previous section. Comparing the models using Precision, Recall, and F1 scores, the GPT-4 models with different shots (five-shot, 10-shot, and zero-shot) generally performed better than other models. The GPT-4 five-shot model showed the highest Precision, Recall, and F1 score, highlighting its superior performance in identifying relevant

instances (Precision) and retrieving a high proportion of actual instances (Recall), illustrated in Table 2 below.

Model	Precision	Recall	F1
gpt4-5shot	**0.695**	**0.634**	**0.637**
gpt4-10shot	**0.661**	**0.603**	**0.578**
gpt4-0shot	0.617	0.442	0.421
mistral-7b-instruct-finetuned	0.407	0.437	0.356
lgbm_with_embeddings	0.433	0.331	0.313
gpt-3.5-turbo-10shot	0.468	0.381	0.297
llama2-7b-chat-finetuned	0.320	0.390	0.273

Table 2. Classification metrics for out-of-domain tests. All metrics are averages of the five classes. **Teal** *and* **Bolded** *values are best and second best, respectively.*

In this testing stage, model performance ranking according to Adjacent Accuracy remained almost identical to the previous tests, with only the zero-shot GPT-4 variant surpassing the fine-tuned Llama-2-7b-Chat. LightGBM with embeddings also kept its position as the best-trained alternative, with the fine-tuned Mistral closely behind it.

Considering results from both the in-domain and out-of-domain tests, we can see that the GPT-4 classifier with five-shot prompting was the best classifier, followed closely by GPT-4 with 10-shot prompting. Notice the bottom two bars in Figures 8 and 9 below.

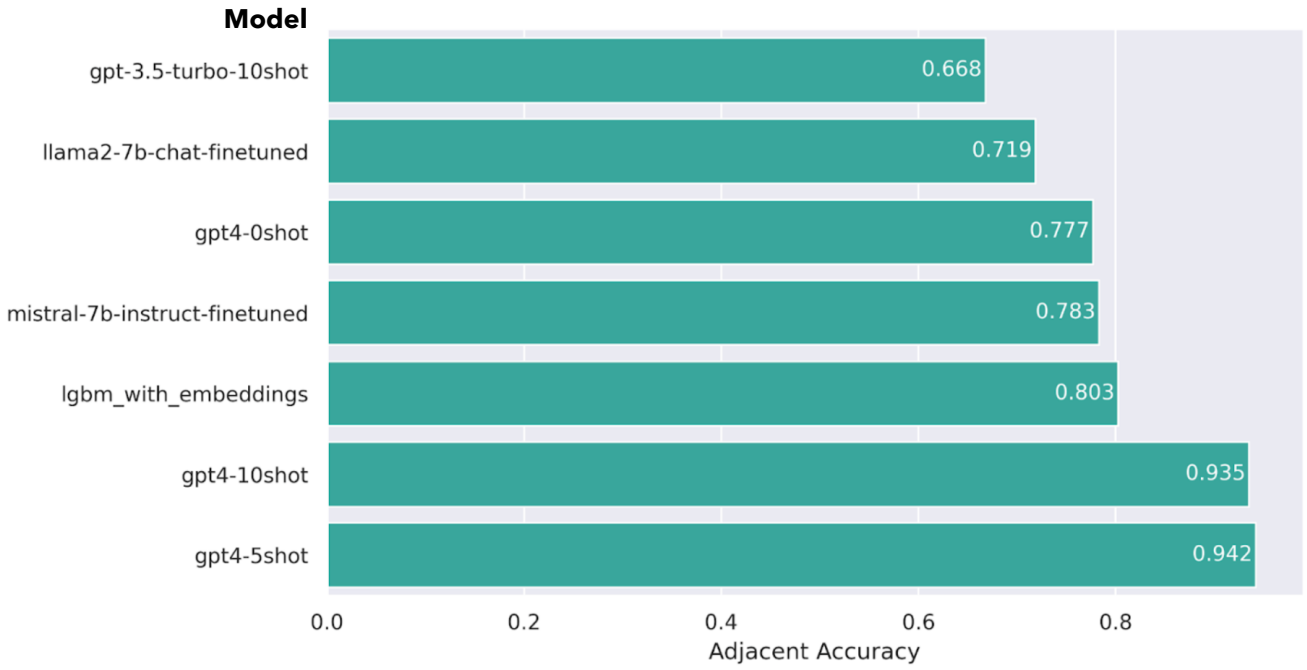


Figure 8. Out-of-domain Adjacent Accuracy for each classifier.

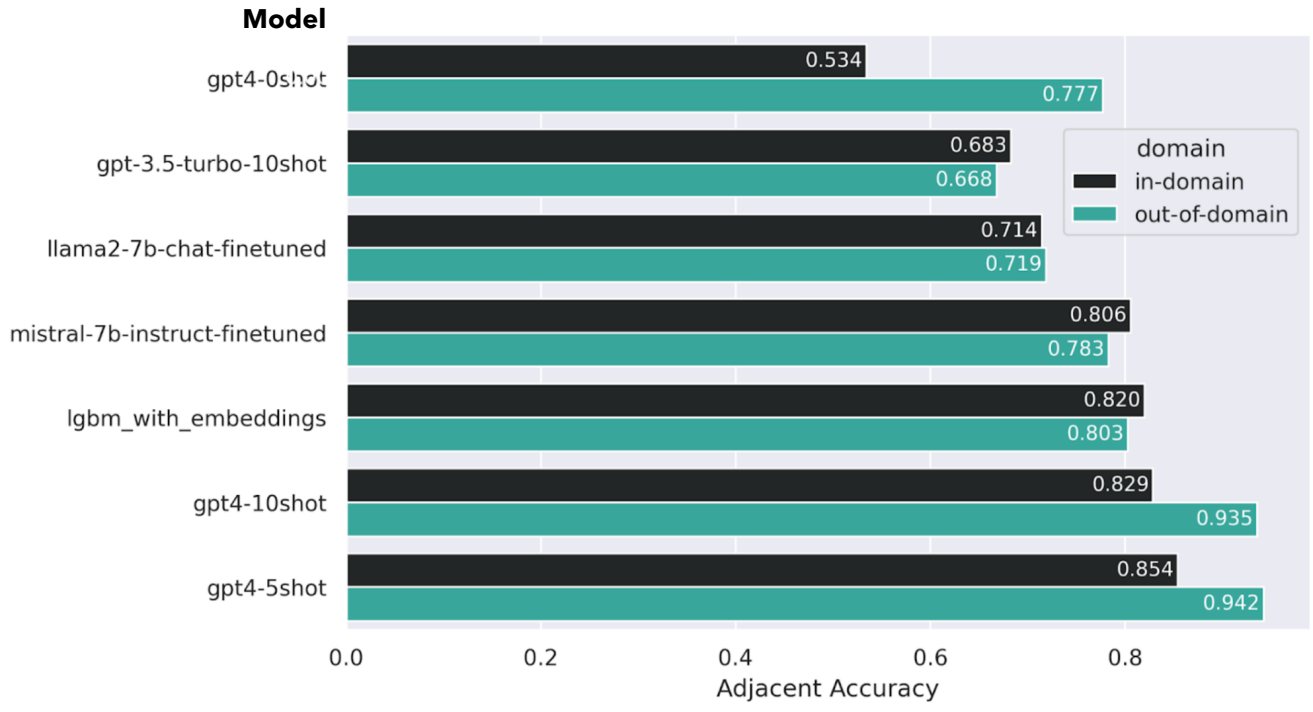


Figure 9. In- and out-of-domain Adjacent Accuracies.

These results illustrate the potential of fine-tuning and few-shot prompting techniques in improving model performance within politeness classification, endorsing their application in future research endeavors.

3.4. Investigating Classifiers' Potential Biases

To investigate the biases our politeness classifiers might have, we performed a syntax analysis on the samples classified at each politeness level by each classifier. According to Dimitrova-Galaczi [6], politeness is a linguistic phenomenon with many facets manifested in different levels, such as lexical, syntactic, pragmatic, sociocultural, non-verbal, and kinesthetic. Hence, the presented investigation is based on the syntactic aspect of politeness.

We applied a TF-IDF approach to extract insights on the terms/words/expressions that were the most relevant for each politeness level predicted by some evaluator. **With such a table, it was possible to compare the evaluators regarding the terms they consider most when giving a specific politeness score, possibly detecting biases**.

N-grams of sizes three and four were used when building the TF-IDF table because of the multi-word nature of polite/impolite expressions rather than unigrams. For instance, the unigram "help" may be scored as relevant for detecting polite samples. But it's more informative if the score is accompanied by the subject, verb, and object "I," "can," and "you," building up the four-gram expression "I can help you."

We analyzed the 10 most relevant N-grams for each evaluator in each politeness level to answer the following questions:

1. **Do the relevant N-grams make sense** (i.e., are the applicable terms and expressions for each politeness level predicted relevant to what humans would consider)?
2. **How do the different evaluators compare to humans** regarding the relevance given for each N-gram in each politeness level?

The first question was answered qualitatively, while the second one required using the cosine similarity between the TF-IDF scores for each politeness level predicted and the TF-IDF scores calculated using the human-reviewed annotations.

To illustrate this, consider the following N-grams set (*"I can help you," "please provide more information," "would you like to"*) with the following TF-IDF score vector (s_1, s_2, s_3) for evaluators' predictions and (h_1, h_2, h_3) for the TF-IDF scores extracted from the human-reviewed real labels.

The cosine similarity is a normalized metric by definition that scores the similarity between vectors with very similar ones getting scores close to one. Hence, comparing the vector of TF-IDFs scores for a predictor with that of the human-reviewed labels is equivalent to telling how similar the predictor's scores are to a human's (i.e., how do the models compare to humans regarding the importance they give to the terms and expressions present in a sentence when classifying its politeness since that's the interpretation of the TF-IDF vector).

The 10 most relevant N-grams for each evaluator in each politeness level appear in Figures 11, 12, and 13 below. The 10 most relevant N-grams for the human-reviewed labels are in Figure 10. The cosine similarity values are in Figure 14, also below.

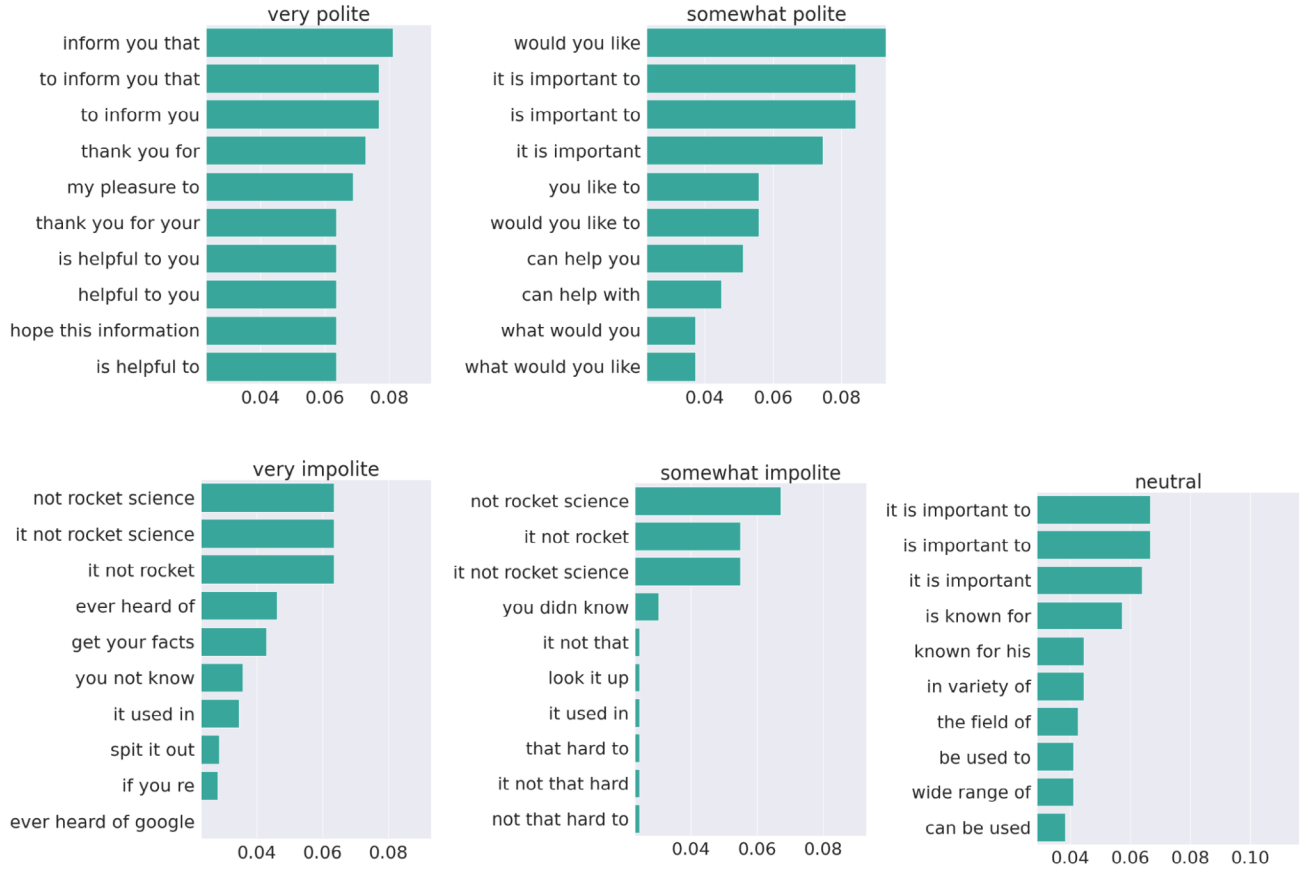


Figure 10. 10 most relevant 4-grams for the human reviewed true labels. The x-axis show the TF-IDFS scores for the 3 or 4-grams

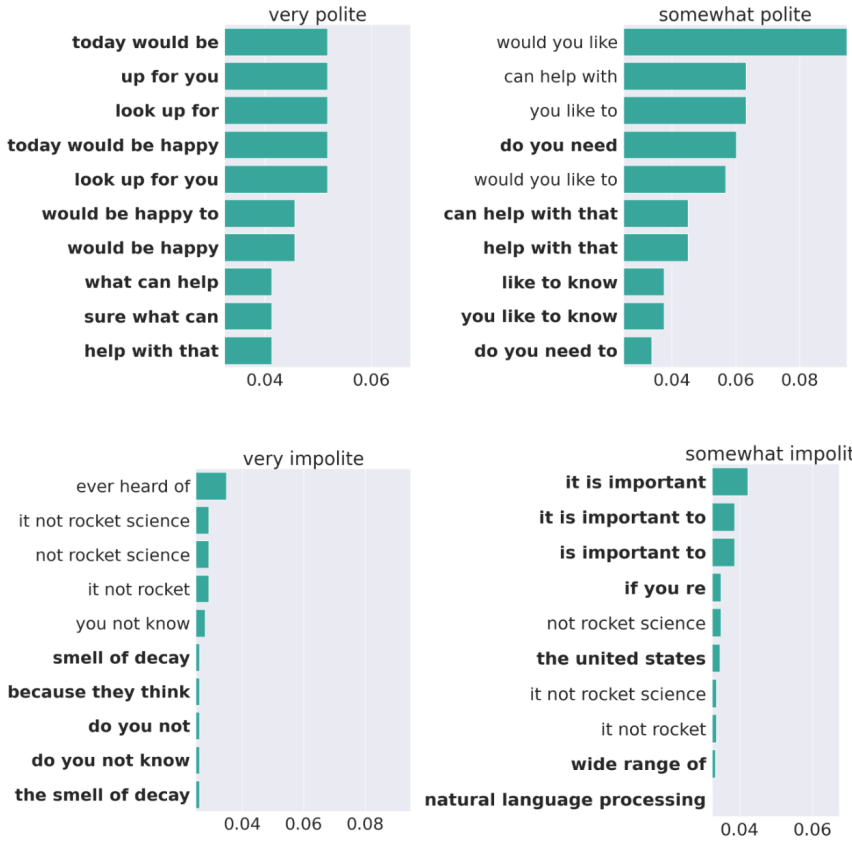


Figure 11. The 10 most relevant four-grams for the Embedding Classifier's predictions. The x-axis shows the TF-IDFs scores for the three- or four-grams.

The **bold** texts indicate the specified N-gram was not among the top 10 most relevant ones for that politeness level in the human-reviewed labels.

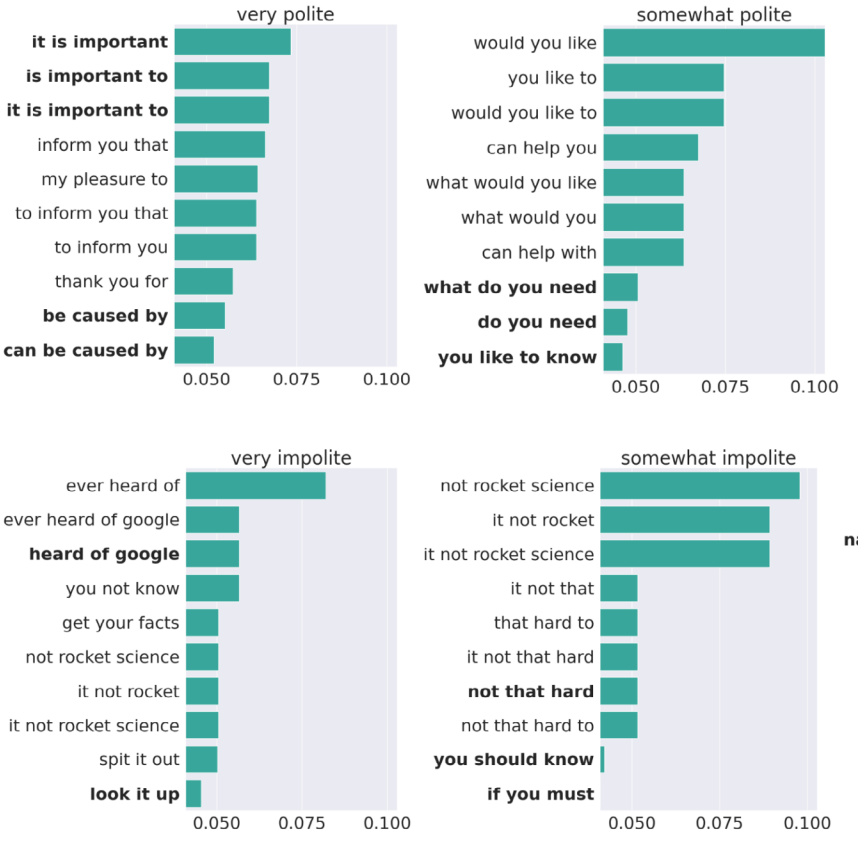


Figure 12. The 10 most relevant four-grams for the LLM Prompt Classifier's predictions. The x-axis shows the TF-IDFS scores for the three- or four-grams.

The **bold** texts indicate the specified N-gram was not among the top 10 most relevant ones for that politeness level in the human-reviewed labels.

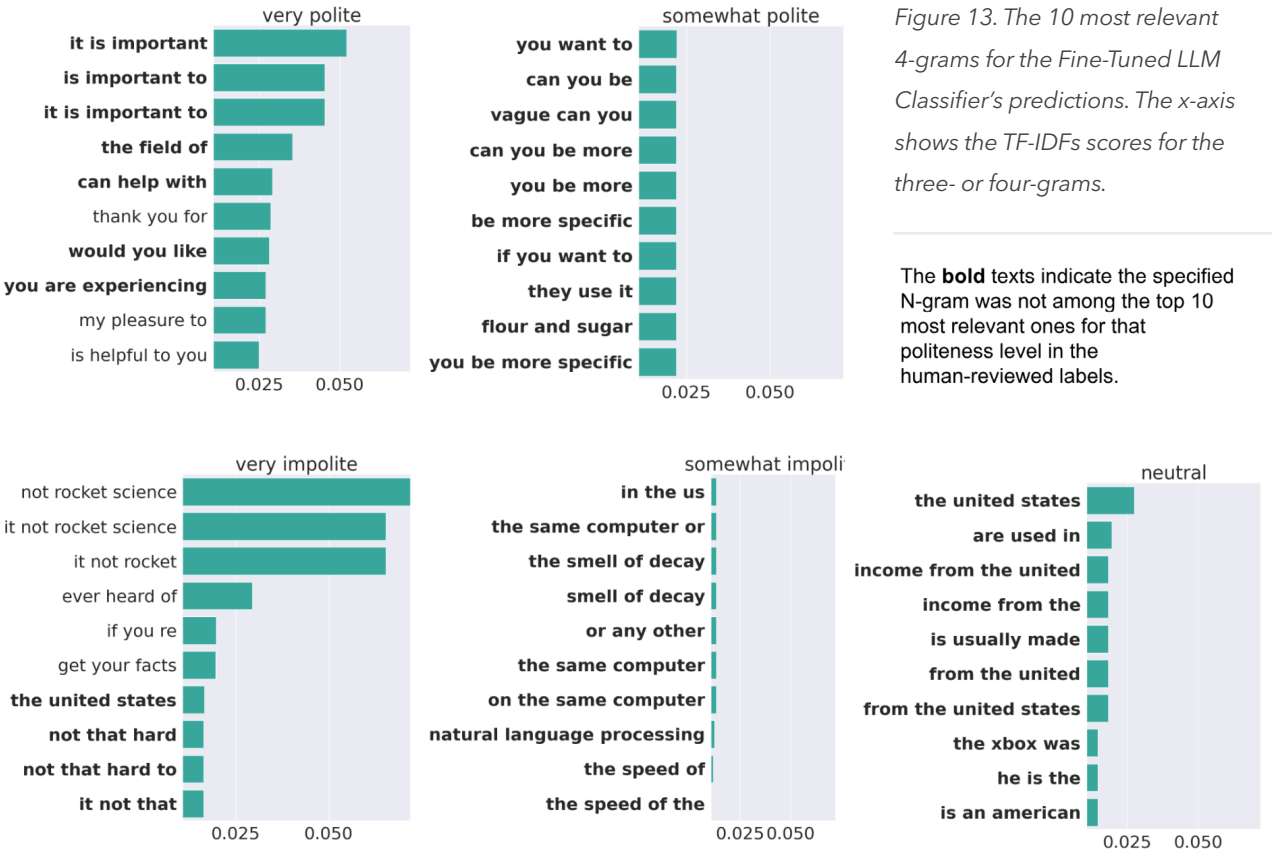


Figure 13. The 10 most relevant 4-grams for the Fine-Tuned LLM Classifier's predictions. The x-axis shows the TF-IDFs scores for the three- or four-grams.

The **bold** texts indicate the specified N-gram was not among the top 10 most relevant ones for that politeness level in the human-reviewed labels.

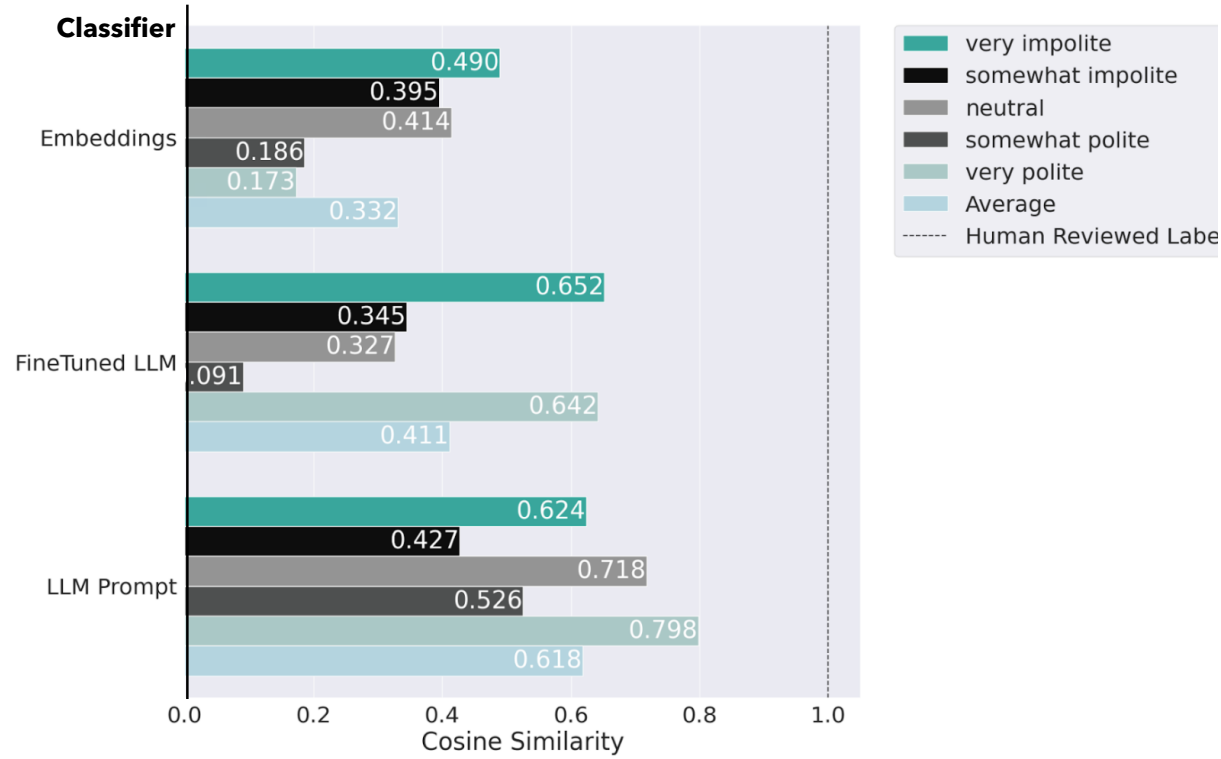


Figure 14. Cosine similarity between the TF-IDF vector from the human-reviewed labels and the evaluator's predictions.

Figures 11, 12, and 13 show that each classifier's relevance attribution to the terms and expressions when labeling the sentences from the human-reviewed out-of-domain test dataset is coherent with what one expects to be relevant for the specified politeness levels in most cases.

On the other hand, the Embeddings and Fine-Tuned LLM Classifiers' relevance attributions for their so-labeled impolite samples were very different from those of the LLM Prompt Classifier and the human-reviewed true labels. In particular, the Embeddings Classifier's TF-IDF values reveal that its top three impolite-relevant expressions were relevant for what humans considered very polite samples.

In the case of the Fine-Tuned LLM Classifier's relevance attributions, besides the impolite-relevant expressions not looking like what humans would consider impolite, they also seem related to few domains. That might indicate a syntactic bias when detecting impolite sentences in such a classifier. However, we need further research on this and other models' feature relevance attribution to establish a better understanding of their behavior.

Furthermore, as previously mentioned, politeness is a multi-faceted phenomenon. Finding biases in other facets also helps us understand better how the evaluators detect those biases in the first place.

The results presented in Figure 14 show that, on average, the LLM Prompt Classifier's TF-IDF N-grams relevance attributions were significantly closer to the human-reviewed labeling. This is consistent with the accuracy results for all the classifiers. Moreover, it indicates that the LLM Prompt Classifier might identify syntactic features of politeness more similarly to how humans would.

On the other hand, regarding the cosine similarities broken down by politeness levels, the Fine-Tuned LLM Classifier was the only one to beat the LLM Prompt Classifier in one politeness score: the very impolite samples. This might indicate that the Fine-Tuned LLM Classifier's relevance attribution to expressions and terms in very impolite samples was closer to what humans would do than the LLM Prompt Classifier.

The fact that the LLM Prompt Classifier attributes syntactic relevance to some expressions and terms more similarly to what humans would do indicates that using GPT-4 as a few-shot classifier for politeness is the best option to mimic the human behavior in such attribution for politeness.

It is important to emphasize that since politeness comprehension has many dimensions rather than only syntactic ones, we need more work to understand better how GPT-4 and other models compare to humans when detecting politeness.

However, the approach presented here might be enough for less complex conversation attributes that are more syntactic-prone when evaluating OpenAI's and other models' natural language understanding capabilities.

4. Conversation Evaluation Methodology

This section discusses our methodology for evaluating conversations with the attribute classifiers we developed: Embeddings Classifier, LLM Prompt Classifier, and Fine-Tuned LLM Classifier (see Section 2 to review each in detail). These attribute classifiers give us a reliable tool for gauging the level of politeness present in any conversation, whether it be between humans, machines, or humans and machines.

Using these attribute classifiers, we can analyze the responses generated by LLMs. This analysis enables us to understand the varying degrees of politeness exhibited by these generator models and how different factors influence them. For this analysis, we chose the classifier GPT-4 with five-shot prompting due to its better performance in both in-domain and out-of-domain tests. To maintain clarity, we refer to the attribute evaluators described in Section 2 as "attribute classifiers," while the models for which we measure politeness are referred to as "chat models."

To facilitate this analysis, we construct a prompt dataset encompassing various scenarios, such as open-domain question-and-answer exchanges, finance-related queries, medical inquiries, and more. This prompt dataset serves as input for the generator models, ultimately generating a list of responses. These responses can then be comparer using the proposed classifier model.

Furthermore, our objective is to establish a set of tests and comparisons that can be easily adapted to assess different attributes. This adaptable suite of tests will enable the generation of insightful reports, even when the target attribute and specific domains within the prompt dataset change.

4.1. Prompt Dataset Construction

To benchmark the politeness of the SOTA chat models when answering user requests, we assembled a prompt dataset by gathering data from the same public datasets used in our out-of-domain test data (see Section 3.2). The data extracted from those were the user prompts rather than the assistant's responses. These prompts went to the different-sized LLMs, and their responses were collectefor further evaluationed by the politeness classifiers.

Similar to when we built the out-of-domain instructions dataset, we based the data selection process on some requirements that had to be met. For inclusion in the final dataset, the prompts must:

1. contain between 3 and 300 words
2. be request-like
3. be lexically diverse

The AlpacaFarm's samples were selected based on the presence of a question mark at their sentences' endings, ensuring the samples would be request-like. HC3 samples were assumed to be request-like due to the dataset's definition. To select the request-like prompts from MetaLWoz, we used the TF-IDF approach as described in the out-of-domain tests (Section 3.3)– only the domains in which the most relevant and frequent words were request-like were kept.

There were 269 request-like prompts from AlpacaFarm using the question-mark criteria, the fewest among the three (after applying requirement 2). The goal was to select samples from MetaLWoz and HC3, ensuring that all of them plus the 269 request-like prompts were lexically diverse. To achieve this, we used a different approach than the one previously described in the assembly of the out-of-domain dataset (see Section 3.3.1). It aimied for better efficiency in accomplishing the same lexical diversity goal, but without the need for any data augmentation or further human reviewing.

To select the sentences that made up the prompt dataset, the most frequent words in MetaLWoz' and HC3's domains were ranked so that at most one sentence containing each one of those words was selected inside each domain. This way, it was possible to avoid selecting many similar sentences with the same lexicons.

The final prompt dataset comprised 920 request-like prompts gathered from AlpacaFarm, MetaLWoz, and HC3 balanced among these datasets' domains. Moreover, 60 more samples came from web forums about tourism and traveling. Therefore, the resulting prompt dataset has 980 samples with 56% average lexical diversity around AlpacaFarm, MetaLWoz, HC3, and web forums' domains – the latter called the tourism domain.

4.2. Comparing the Politeness of Chat Models

Having the request-like prompt dataset assembled, we fed it to some of the SOTA chat models and collected their responses, which were further classified by the LLM Prompt Classifier. We didn't use a system prompt when sending the messages to the chat models (though it would be valuable to check how this would impact politeness scores in future research). As for the chat models themselves, we used the following with their temperatures set to zero:

- Claude 2.1 [17]
- Claude 3 Sonnet [18]
- Claude 3 Opus [18]
- Gemini 1.0 Pro [19]
- GPT-4 [20]
- Llama 2 70B Chat [21]
- Llama 2 13B Chat [21]
- Mixtral-8x7B Instruct 2 [22]

During the generation of responses, not all models successfully responded to all prompts. Some failed because they hit the API's moderation layers and were blocked. API failures caused other unsuccessful responses. If a prompt failed in at least one model, we discarded it from the analysis of all models. Due to this approach, the dataset ultimately totaled 888 prompts.

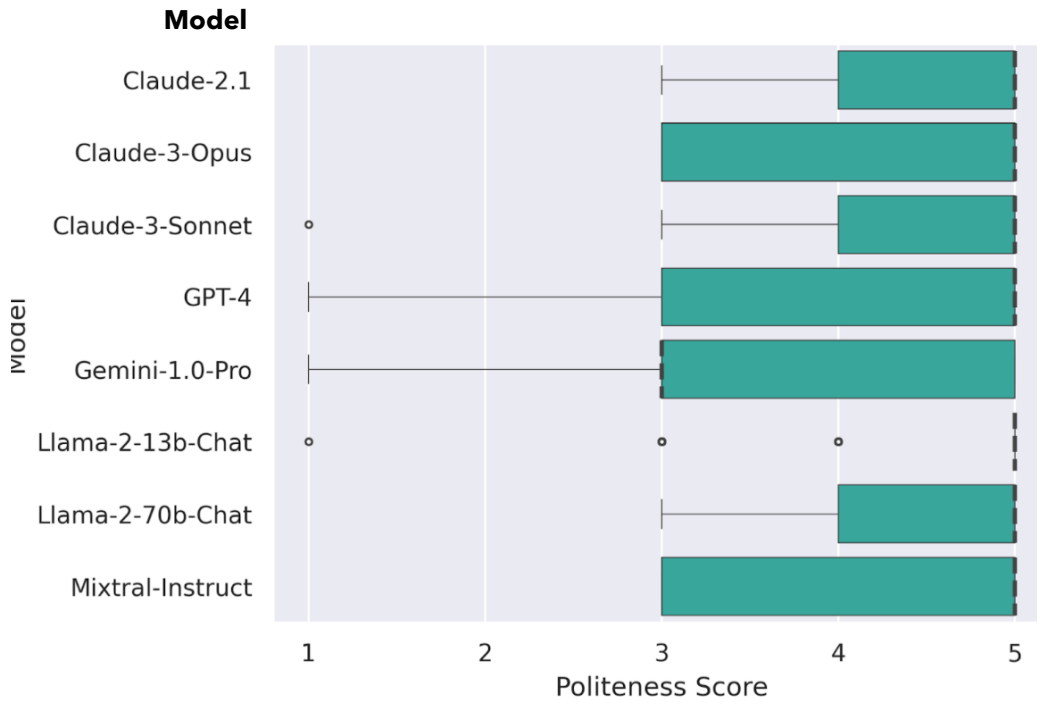


Figure 15. Politeness score boxplots by chat model. The dashed lines represent the median values.

To compare the outputs of the chat models to the same set of prompts, we performed an exploratory data analysis of their politeness. Our measured scale of politeness goes from level 1 to 5, which corresponds to the labels "Very Impolite," "Somewhat Impolite," "Neutral," "Somewhat Polite," and "Very Polite," respectively. To simplify the visualization in some of the graphs, we also used the following scale:

- Polite: scores 1 and 2 (Very Impolite or Somewhat Impolite)
- Neutral: score 3 (Neutral)
- Impolite scores 4 and 5 (Very Polite or Somewhat Polite)

Regarding the measured politeness scores of the chat models, all models generally tended to answer politely, with the median values falling into the most polite category (except for Gemini 1.0 Pro). Notably, no answer classified as 2, but a few classified as 1 on the scale. The smallest model used was Llama 2 13B Chat, and it turns out that it was the model that generated the most polite answers: 75% of utterances were rated as Very Polite, as seen in Figure 16 below.

Claude 2.1, Claude 2 Opus, and Mixtral did not respond with any utterance rated as impolite. Gemini 1.0 Pro and GPT-4 showed a broader range of politeness in their responses. Gemini 1 Pro is the least polite model, with more than half of its utterances falling in the neutral category. GPT-4 showed more nuance, having more answers score in category 4 than the other models.

Figure 16. Proportion of utterances in each category by model.

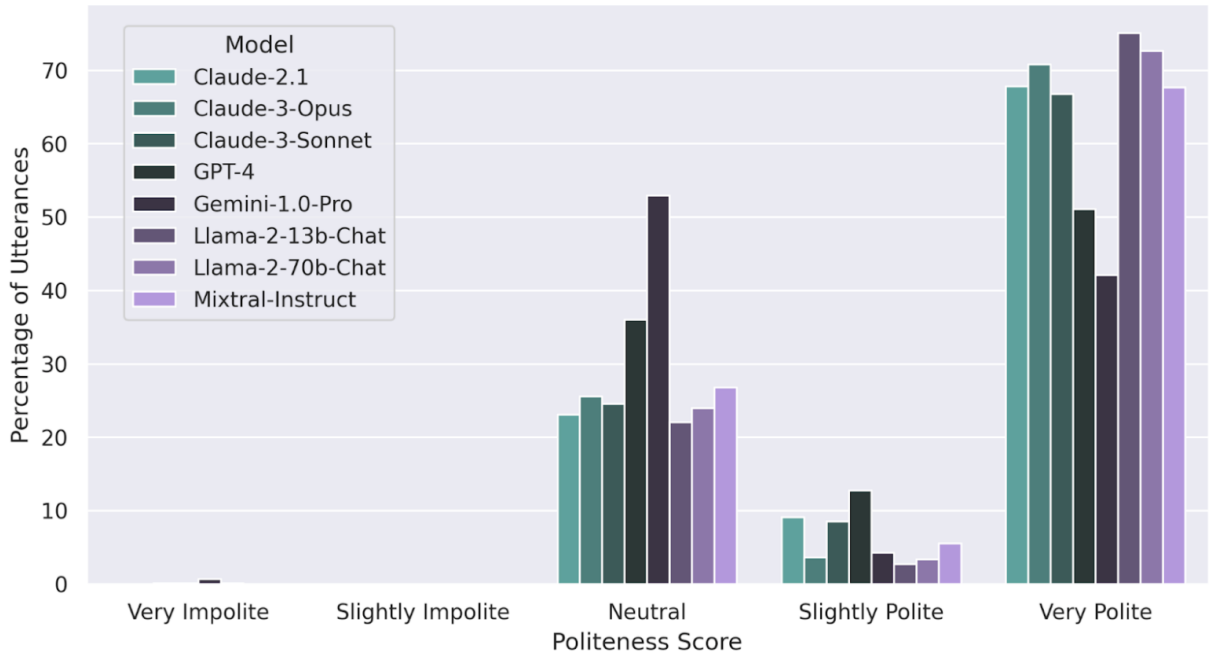


Figure 17. Percentage of utterances for each category by model.

The prompts dataset contains utterances from six different domains. We analyzed how the models' answers collectively differed from domain to domain, and how their answers varied inside the domains. The domain that got mostly neutral questions was the "wiki_csai" (which we expected since its contents are only straightforward questions about computer science and artificial intelligence concepts).

The domain that got the most polite answers was the "medicine" domain, which implies that the models can respond respectfully to vulnerable situations. For specific models, we can highlight that Gemini was the least polite model in all domains except for "wiki_csai."

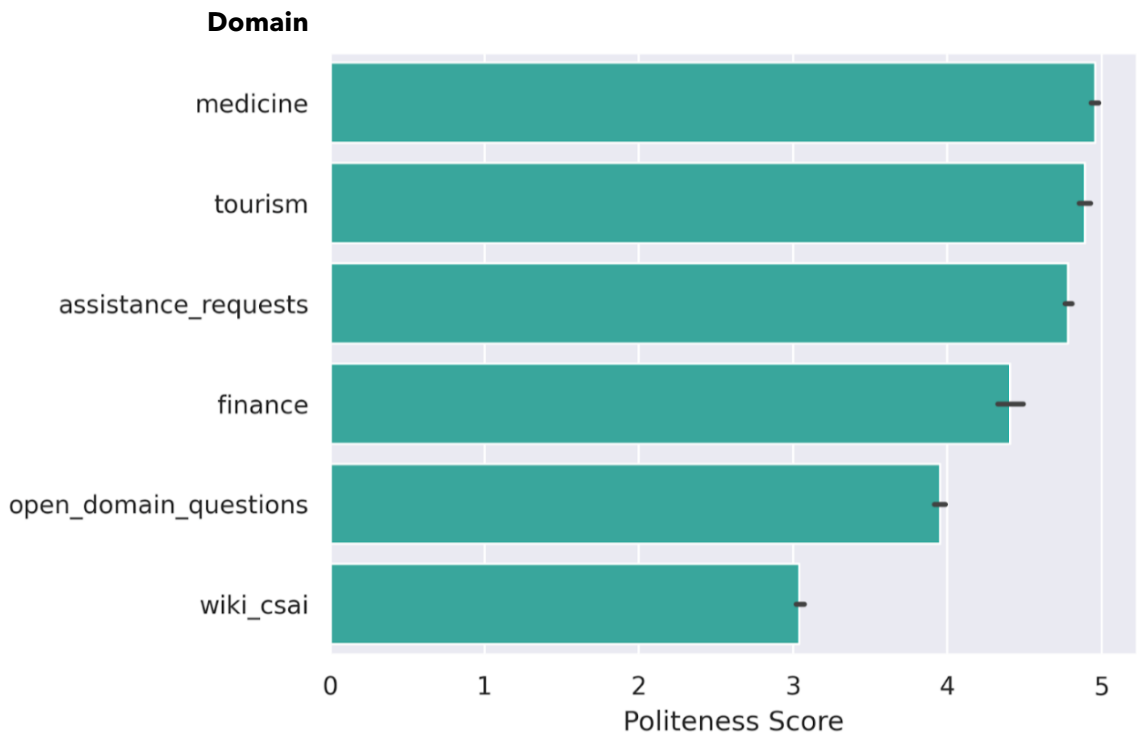


Figure 18. Average politeness score by domain, considering all models. Error bars represent the bootstrap confidence interval.

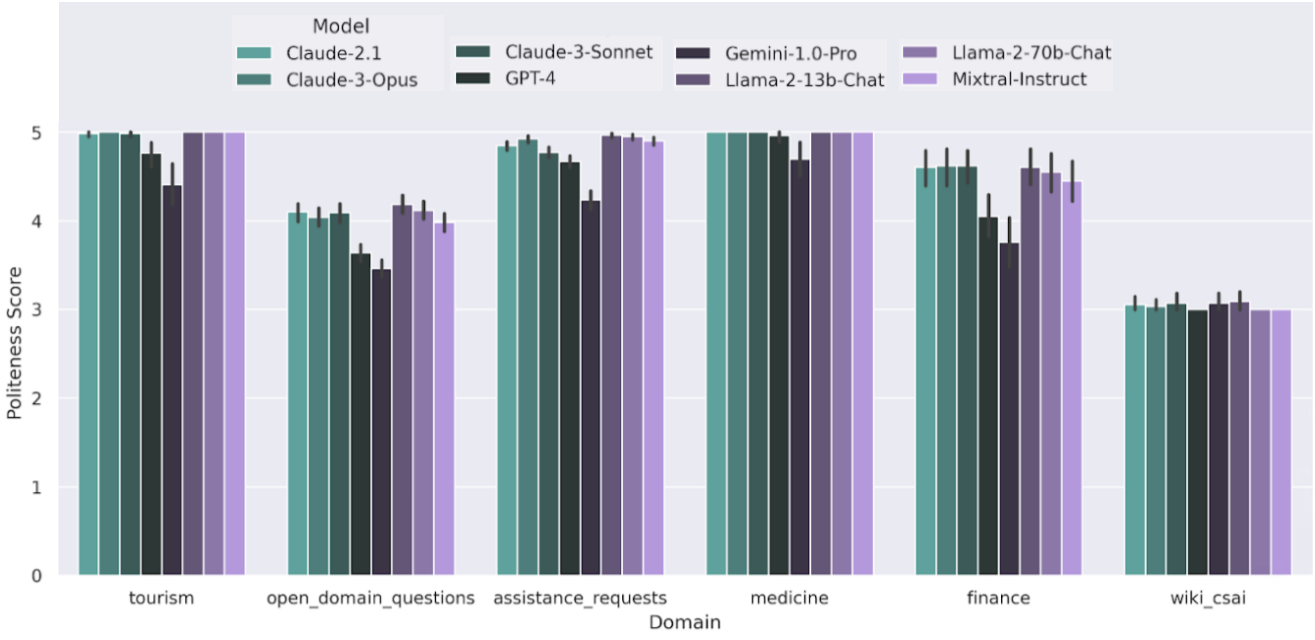


Figure 19. Average politeness scores by model and domain. Error bars represent the bootstrap confidence interval.

Besides measuring the politeness of chat models' utterances, we also evaluated the politeness of the prompts dataset. All the prompts were scored for politeness using the same classifier used to score the chat models' responses. Since the prompts dataset consists of utterances collected from diverse sources, not all prompts have the same politeness alignment, some being more harsh and direct than others. We investigated if and how this distinction would affect the models' responses.

In the heatmap below (Figure 20), we show how the models respond to prompts in each class of politeness. We can see that even for impolite prompts, all models responded politely. Neutral prompts tended to get more neutral responses than polite and impolite prompts. For polite prompts, more than 90% of the responses were also polite. Further below in Figure 21, we see that Gemini 1.0 Pro is the only model that responded with more neutral answers than polite answers for the neutral prompts category.

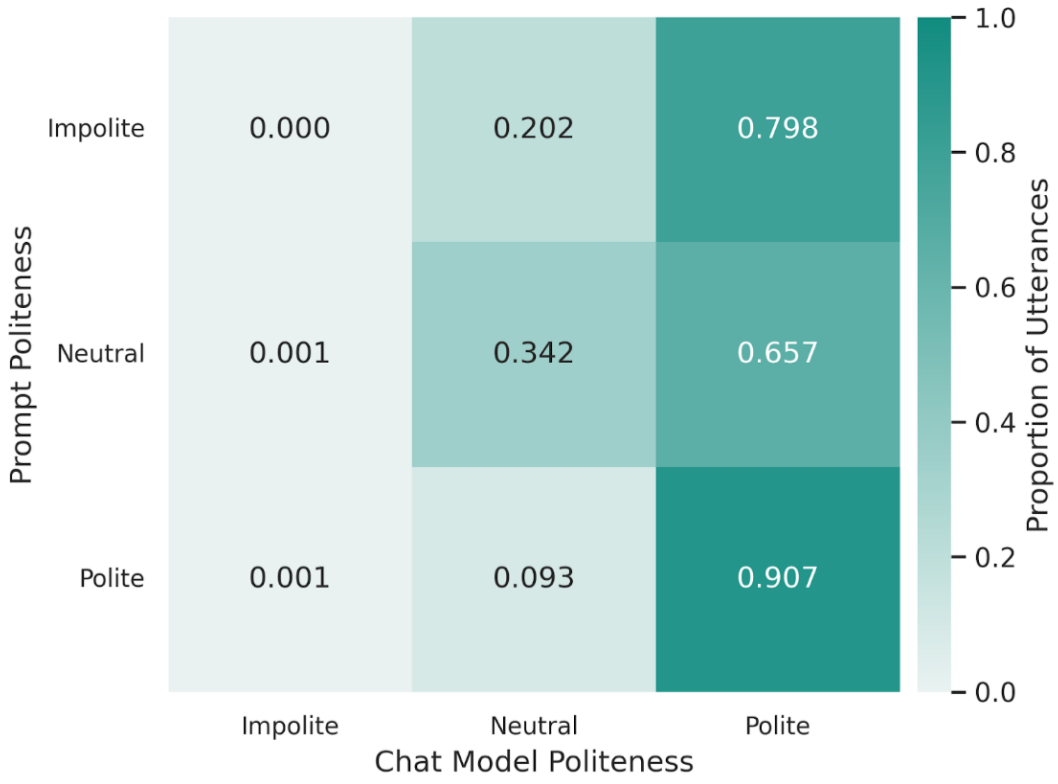


Figure 20. Politeness of the prompts versus politeness of the responses, considering all models.

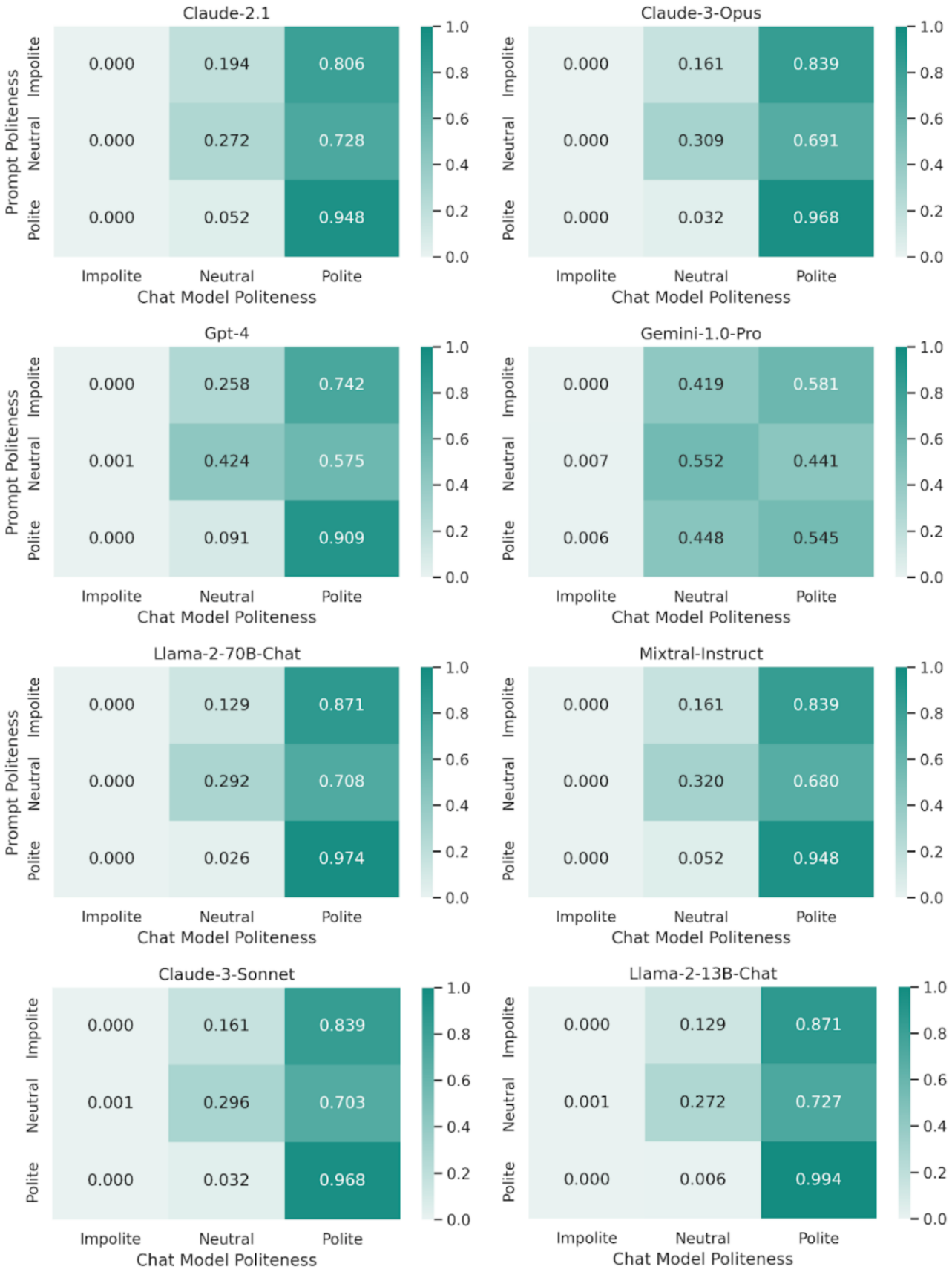


Figure 21. Politeness of prompt versus politeness of response for each model.

5. Conclusion: Testing Conversational AI Frameworks at Scale

This whitepaper shows that particular attributes of conversation — even inherently nuanced dimensions such as politeness — can be monitored for in generative AI systems. We also saw how current SOTA LLMs all skew strongly away from impolite answers by nature, as we expected, regardless of the politeness of the question.

In testing our framework, we saw the best results with our LLM Prompt Classifier (i.e., the few-shot in-context learning prompt classifier) regarding the specific attribute of politeness. From here, future research could reveal much more. Other valuable experiment variations that come to mind include:

- adding more context to classifier inputs (e.g., longer conversation history)
- testing with human conversations
- testing with more domains and biases
- evaluating conversation attributes over time vs. scoring everything at once
- using more augmentation models for out-of-domain dataset assembly
- using techniques other than TF-IDF to perform classifiers' interpretability (especially the ones that tailor more than syntactic and lexical features)
- investigating how the prompts impact the conversational attributes in the outputs

You can run these experiments to fine-tune the conversational abilities of your own generative AI systems. With an enterprise AI platform Fuel iX, you can:

- plug-and-play different LLMs to avoid vendor lock-in
- centralize control for visibility and good AI governance
- build workflows that make trustworthy AI your default

Learn more by visiting **Fuel iX**.

6. References

[1] R. Bowman et al., "Exploring how politeness impacts the user experience of chatbots for mental health support," Int. J. Hum.-Comput. Stud., vol. 184, p. 103181, Apr. 2024, doi: 10.1016/j.ijhcs.2023.103181.

[2] P.-N. Schwab, L. Rosier, and S. Rothenberger, "Politeness matters: The antecedents and consequences of politeness in a complaint handling setting," ULB –Universite Libre de Bruxelles, Working Papers CEB 15–011, Apr. 2015. [Online]. Available: https://ideas.repec.org/p/sol/wpaper/2013-197891.html

[3] S. J. S. Jungco, "The Use of Politeness Strategies and Respect Markers by Call Center Agents," Int. J. Soc. Sci. Hum. Res., vol. 04, no. 08, Aug. 2021, doi: 10.47191/ijsshr/v4-i8-04.

[4] A. Salinas and F. Morstatter, "The Butterfly Effect of Altering Prompts: How Small Changes and Jailbreaks Affect Large Language Model Performance." arXiv, Jan. 09, 2024. Accessed: Mar. 26, 2024. [Online]. Available: http://arxiv.org/abs/2401.03729

[5] C. Danescu-Niculescu-Mizil, M. Sudhof, D. Jurafsky, J. Leskovec, and C. Potts, "A Computational Approach to Politeness with Application to Social Factors." arXiv, Jun. 25, 2013. Accessed: Mar. 04, 2024. [Online]. Available: http://arxiv.org/abs/1306.6078

[6] E. Dimitrova-Galaczi, "Issues in the Definition and Conceptualization of Politeness," Stud. Appl. Linguist. TESOL, vol. 2, no. 1, Art. no. 1, May 2002, doi: 10.7916/salt.v2i1.1650.

[7] OpenAI, "GPT-3.5 Turbo API Documentation." [Online]. Available: https://platform.openai.com/docs/models/gpt-3-5-turbo

[8] G. Ke et al., "Lightgbm: A highly efficient gradient boosting decision tree," Adv. Neural Inf. Process. Syst., vol. 30, pp. 3146–3154, 2017.

[9] J. H. Friedman, "Greedy function approximation: A gradient boosting machine.," Ann. Stat., vol. 29, no. 5, pp. 1189–1232, 2001, doi: 10.1214/aos/1013203451.

[10] T. B. Brown et al., "Language Models are Few-Shot Learners." arXiv, Jul. 22, 2020.

Accessed: Mar. 26, 2024. [Online]. Available: http://arxiv.org/abs/2005.14165

[11] E. J. Hu et al., "LoRA: Low-Rank Adaptation of Large Language Models." arXiv, Oct. 16, 2021. Accessed: Mar. 12, 2024. [Online]. Available: http://arxiv.org/abs/2106.09685

[12] "MetaLWOz," Microsoft Research. Accessed: Mar. 26, 2024. [Online]. Available: https://www.microsoft.com/en-us/research/project/metalwoz/

[13]Y. Dubois et al., "AlpacaFarm: A Simulation Framework for Methods that Learn from Human Feedback." arXiv, Jan. 07, 2024. doi: 10.48550/arXiv.2305.14387.

[14] B. Guo et al., "How Close is ChatGPT to Human Experts? Comparison Corpus, Evaluation, and Detection." arXiv, Jan. 18, 2023. doi: 10.48550/arXiv.2301.07597.

[15] R. Baeza-Yates and B. Ribeiro-Neto (2011). Modern Information Retrieval. Addison Wesley, pp. 68-74.

[16] C.D. Manning, P. Raghavan and H. Schütze (2008). Introduction to Information Retrieval. Cambridge University Press, pp. 118-120.

[17] Anthropic, "Claude 2," Anthropic News. [Online]. Available: https://www.anthropic.com/news/claude-2

[18] Anthropic, "Introducing the next generation of Claude," Anthropic News. [Online]. Available: https://www.anthropic.com/news/claude-3-family

[19] S. Pichai and H. Demis, "Introducing Gemini: our largest and most capable AI model," Google The Keyword. [Online]. Available: https://blog.google/technology/ai/google-gemini-ai/#sundar-note

[20] OpenAI et al., "GPT-4 Technical Report." arXiv, Mar. 01, 2024. Accessed: Mar. 04, 2024. [Online]. Available: http://arxiv.org/abs/2303.08774

[21] H. Touvron et al., "Llama 2: Open Foundation and Fine-Tuned Chat Models." arXiv, Jul. 19, 2023. Accessed: Mar. 04, 2024. [Online]. Available: http://arxiv.org/abs/2307.09288

[22] A. Q. Jiang et al., "Mixtral of Experts," no. arXiv:2401.04088. arXiv, Jan. 08, 2024. Accessed: Jan. 18, 2024. [Online]. Available: http://arxiv.org/abs/2401.04088

7. Appendix

In this section, we present the running times and token usage of the presented study. We divided the reports into costs to train and test the conversation attributes classifiers (Table 3) and the ones to compare the chat models regarding their politeness (Table 4). The model completion calls to assemble the prompts dataset were made using their provider's APIs and the virtual machine used to train and test some of the classifiers was provisioned in Microsoft Azure (NC24ads A100 v4 model). The tests performed with the Embeddings Classifier were done locally in a Dell Precision 3571 Core i713th generation machine with 32GB of RAM, so no virtual machine was needed to test it.

Phase	Attribute Classifier	Service /Provider	#Input Tokens	#Output Tokens	Total (s) Runtime
Training	Embedding Classifier	Azure VMs*	109664	-	756
	FineTuned LLM Classifier	Azure VMs**	743232	-	1193
In-domain Tests	LLM Prompt Classifier	OpenAI API	227247	14797	4662
	Embedding Classifier	Local Machine*	11271	-	77
	FineTuned LLM Classifier	Azure VMs**	83156	83156	146
Out-of-domain Tests	LLM Prompt Classifier	OpenAI API	686677	65109	10485
	Embedding Classifier	Local Machine*	54600	-	225
	FineTuned LLM Classifier	Azure VMs**	268366	268366	426

*Table 3. Costs to train and test the classifiers (*Dell Precision 3571 / **NC24ads A100 v4). The LLM Prompt Classifier did not need training.*

		Completion			LLM Prompt Classifier Evaluation	
Chat Model	**Service/ Provider**	**#Input Tokens**	**#Output Tokens**	**Service/ Provider**	**#Input Tokens**	**#Output Tokens**
Claude 2.1	Anthropic API	23805	218449	OpenAI API	820392	225532
Claude 3 Opus		24069	237990		838877	244017
Claude 3 Sonnet		23684	238529		839507	244647
GPT-4-0613	Open AI API	23721	151592		762665	167805
Gemini 1.0	Google API	22180	335880		971042	376182
Llama 2 13B Chat	AWS Bedrock	27716	402815		959149	364289
Llama 2 70B Chat		27716	394626		952151	357291
Mixtral-8x7B	Mistral API	26778	289133		870767	275907

Table 4. Costs to perform and collect chat completion services using the prompts dataset.