# Unraveling MBTI Types from Online Posts

**Daneshvar Amrollahi**
daneshvar@cs.stanford.edu

**Salma Zainana**
szainana@stanford.edu

## 1 Introduction

Understanding human personality traits, particularly through the Myers-Briggs Type Indicator (MBTI), is increasingly important for applications like personalized content recommendation and targeted marketing. The MBTI classifies individuals into one of 16 personality types based on four binary attributes. Accurately identifying an individual's MBTI type from their digital activities can greatly enhance personalized user experiences. Our algorithm processes online forum posts by an individual as its input. We use Naive Bayes and GPT-4 models to predict the individual's overall MBTI personality type from the input. Additionally, we employ Neural Networks to predict each attribute within the individual's MBTI type.

## 2 Related Work

Numerous studies have investigated the relationship between personality analysis and social media data, with a special emphasis on leveraging machine learning and natural language processing technologies Basto (2021); Datta et al. (2023); Kadambi (2021); Keh and Cheng (2019). Both Kadambi (2021) and Keh and Cheng (2019) have refined the BERT model specifically for the Myers-Briggs Type Indicator (MBTI) classification task. Notably, Kadambi (2021) examines a range of feature encoding methods. A distinctive feature of Datta et al. (2023) is its use of an extensive dataset of MBTI types, consisting of 52 million samples. The study by Basto (2021) is centered on Long Short-Term Memory (LSTM) networks, and although it employs the BERT model, it does so without any model refinement. To our knowledge, there has been no previous effort to apply the GPT model for MBTI classification.

## 3 Dataset and Features

### 3.1 Dataset

Our dataset, stored in a CSV file, consists of 8675 entries across two columns: one for the MBTI personality type and another for 50 concatenated forum posts related to that type, separated by triple vertical bars ('|||'). Although the dataset is available on Kaggle, it is not part of a contest, and therefore, there is no baseline model provided for comparison of our results.

### 3.2 Data Preprocessing

In preparing our raw data for analysis, we implement several key cleaning steps: First, we **remove irrelevant characters** such as special characters and punctuation to ensure consistency and reduce text dimensionality. Next, we **eliminate stop-words**—common words lacking meaningful context—to focus on significant terms, thus lowering computational complexity. We also apply **lemmatization**, an NLP technique that simplifies words to their base form, streamlining the dataset and reducing feature space. Finally, we address class imbalance in our dataset (See Figure 1) using the **Synthetic Minority Over-sampling Technique (SMOTE)**, which creates synthetic instances of the minority class to improve dataset balance and enhance model performance.

### 3.3 Feature Extraction

**TF-IDF (Term Frequency-Inverse Document Frequency)** TF-IDF, a statistic that highlights a word's relevance in a document within a corpus, complements Naive Bayes classifiers well. It provides a sparse representation of documents as vectors in a high-dimensional space, with dimensions weighted by TF-IDF scores, fitting Naive Bayes' assumption of feature independence. This combination has been effective in text classification tasks like spam detection, sentiment analysis, and topic classification, benefiting from the interpretability and statistical strengths of both methods.

**Word2Vec** Word2Vec, a notable technique in NLP for generating word embeddings, is particularly effective when paired with Recurrent Neural Networks (RNNs) for several reasons. Firstly, it supports semantic representation and generalization by capturing the semantic relationships between words, aiding RNNs in understanding and learning word associations within sequences, thus improving generalization from training data. Additionally, Word2Vec offers dimensionality reduction, producing dense embeddings that streamline RNN training compared to the sparse representations of one-hot encoding. Finally, the density of Word2Vec embeddings contributes to more stable gradient flow in RNNs, mitigating issues related to exploding and vanishing gradients and enhancing model learning efficiency.

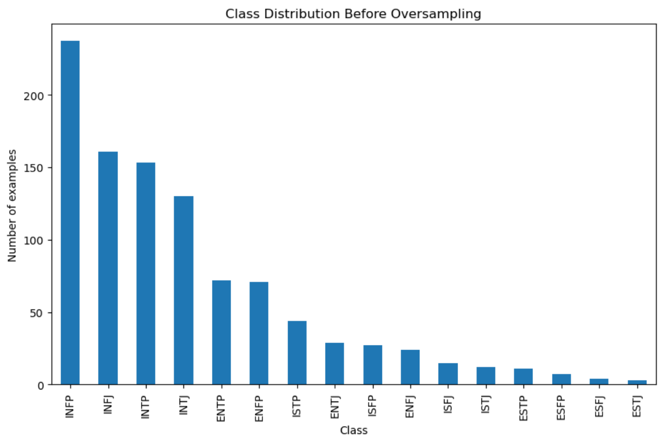


Figure 1: Data distribution before oversampling

## 4 Methods

### 4.1 Naive Bayes

The Naive Bayes algorithm is a simple yet effective classification method based on Bayes' Theorem. It assumes all features are independent of each other within each class, simplifying calculations. This algorithm calculates the probability of each class given a set of features and assigns the data point to the class with the highest probability. Despite its simplicity, Naive Bayes performs well in many scenarios, especially in text classification tasks like spam detection and sentiment analysis.

### 4.2 Neural Networks

Neural networks use layers of artificial neurons to process text data for MBTI classification by first converting words into numerical vectors through techniques like word embeddings. These vectors capture semantic relationships and are fed into the network, which may consist of recurrent layers (like LSTM or GRU) to handle sequential data inherent in text. The network learns by adjusting weights through backpropagation, optimizing a loss function to distinguish between the 16 MBTI types based on the patterns found in the training data. Essentially, it identifies and leverages linguistic and contextual features to predict personality classifications.

#### Architecture

- **Embedding Layer:** Converts integer-encoded words into dense vectors of a fixed size, facilitating semantic understanding of words through high-dimensional representation.
- **Layer Types:**
  - *SimpleRNN Layer:* Sequentially processes text, maintaining input 'memory' for short sequences but struggles with longer ones due to vanishing gradients.
  - *Multiple LSTM Layers:* Excels in learning long-term dependencies, with stacked layers improving recognition of complex patterns related to linguistic traits.
  - *Bidirectional LSTM Model:* Analyzes text in both directions to fully capture context, crucial for interpreting sequences reflective of personality types.
  - *Conv1D Layers:* Applies convolution to 1D text data to identify local patterns and syntactic features associated with specific communication styles.
- **Dense Layer:** Serves as the output layer, with further details provided in Section 5.1, designed according to the task requirements.

### 4.3 GPT

GPT-4 OpenAI et al. (2024), a state-of-the-art large language model (LLM), processes MBTI classification by understanding and generating human-like text. It's pretrained on a vast corpus of text data, enabling it to grasp intricate language patterns, context, and semantics. For MBTI classification, GPT-4 receives text inputs—such as personal statements or writings indicative of personality traits—and uses its deep understanding of language to predict the writer's MBTI type. It achieves this through a mechanism called "attention," which allows the model to weigh the importance of different words and phrases in the context of the entire text input, making informed predictions about personality types.

## 5 Experiments

In this section, we detail the experimental setup and present the outcomes of applying our machine learning models to the task of classifying MBTI personality types. We have made the code for our experiments available to the public at our GitHub repository[1].

### 5.1 Naive Bayes

First, without upsampling the minority types, we picked 5000 rows of our data and used the 80/20 split rule for train and test data. We have trained models by considering uni-grams, bi-grams, and tri-grams for different values of the `max_features` parameters. The results are shown in Figure 2

The model accuracy is peaking at `max_features = 500` and decreasing afterward. The contributing reasons are:

1. **Overfitting**: Adding too many features can cause the model to learn from noise in the training data rather than actual patterns, leading to poor generalization.
2. **Feature Redundancy**: Beyond a certain point, new features may not add useful information and could introduce noise, reducing model accuracy.
3. **Increased Complexity and Noise**: More features increase data complexity and the likelihood of including non-informative noise, detracting from model performance.
4. **Sparse Data**: Higher dimensionality with many features results in sparse data, which dilutes the influence of important features and complicates accurate prediction.

After upsampling using SMOTE, we see a 15% increase in the accuracy reaching to 60% which is a high number for such a simple model compared to the other work mentioned in the introduction. The results after upsampling are shown in Figure 3.

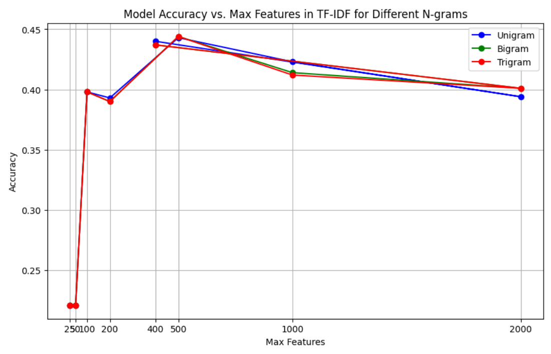


Figure 2: Naive Bayes without SMOTE

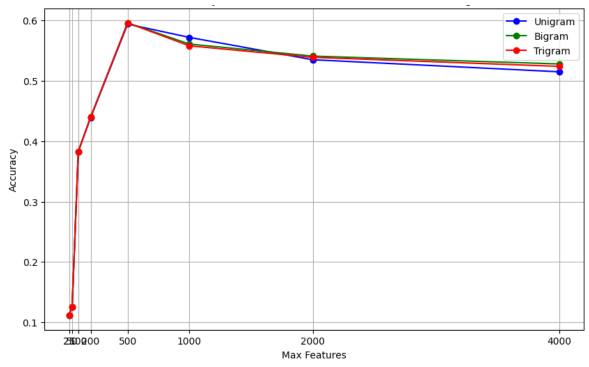


Figure 3: Naive Bayes with SMOTE

### 5.2 Neural Network

**Hyperparameters** To optimize our model, we carefully selected a set of hyperparameters. Below is a detailed explanation of each:

- ***vocab_size***: The total number of unique words within our dataset, set to 135,303. This parameter defines the size of our vocabulary and influences the complexity of the embedding layer.
- ***embedding_size***: The dimensionality of the word vectors, set to 300. A higher dimension allows for capturing more nuanced semantic relationships between words.
- ***max_seq_length***: The maximum length of input sequences, capped at 957. This value ensures uniformity in input size, crucial for processing sequences in batches.
- ***kernel_size***: The size of the convolutional kernel, set to 3. This affects the window size for capturing local features within the sequences.
- ***filters***: The number of output filters in the convolutional layers, set to 64. It determines the depth of the feature maps.
- ***hidden_dims***: The size of the hidden layer, set to 250. This parameter impacts the model's ability to learn complex patterns.
- ***num_epochs***: The total number of training cycles through the entire dataset, set to 30. This influences how well the model learns from the data.
- **batch_size**: The number of samples processed before the model is updated, set to 128. It affects the speed and stability of learning.

Each hyperparameter was selected based on preliminary experiments and literature reviews to balance between learning efficiency and computational resource usage.

[1]https://github.com/daneshvar-amrollahi/mbti-cs229

**Additional pre-processing** In the development of our NN model, we loaded a pre-trained word embeddings model from the Google News Word2Vec model, using Gensim's KeyedVectors API for efficiently handling the embeddings. This model comprises 300-dimensional vectors trained on a vast corpus of Google News articles, offers a comprehensive semantic representation of words based on their contextual usage.

To tailor these pre-trained embeddings for our specific application, we constructed an embedding matrix of dimensions (*vocab_size, embedding_size*). This matrix was initially filled with zeros. Subsequently, for each word present in our tokenizer's vocabulary and also found in the Word2Vec model, we extracted its 300-dimensional vector from the Word2Vec model and assigned it to the corresponding row in our embedding matrix, indexed by the word's integer encoding in our tokenizer.

This approach allowed us to leverage the rich semantic information encoded in the Word2Vec embeddings, directly applying this knowledge to enhance the learning process of our neural network

### 5.2.1 Results

Different model architectures were utilized and combined for the prediction of MBTI types, including SimpleRNN and LSTM (incorporating Bidirectional LSTM), as well as LSTM with Conv1D, followed by Conv1D models on their own. Our best model follows the below architecture:

```
model = Sequential()
model.add(Embedding(vocab_size, embedding_size,
    weights=[embedding_matrix], input_length=MAX_SEQ_LENGTH))
model.add(Conv1D(filters, kernel_size, padding='valid', activation='relu'))
model.add(MaxPooling1D())
model.add(Flatten())
model.add(Dense(hidden_dims, activation='relu'))
model.add(Dense(4, activation='sigmoid'))
model.compile(loss='binary_crossentropy',
    optimizer='adam', metrics=['accuracy'])
```

The use of an embedding layer with a pre-trained embedding matrix allows for the incorporation of rich pre-existing linguistic knowledge. The Conv1D layer excels in extracting local and positionally invariant features, which is particularly advantageous in text analysis, while MaxPooling reduces the computational load and prevents overfitting. The dense layers, with a high number of dimensions, allow for learning complex patterns, which is essential for distinguishing between the subtle linguistic cues of different personality types. The sequential architecture, culminating with a sigmoid-activated output layer, makes the model well-suited for handling the binary classification inherent in the one-hot encoded MBTI dichotomies.

The optimized performance metrics, as indicated in the Table 1, underscore the efficacy of this approach in balancing precision and recall despite the challenges posed by the imbalanced dataset.

Table 1: Summary of Performance Metrics for MBTI Dichotomies

| MBTI Dichotomy | ROC-AUC Score | Average Precision-Recall Score | Geometric Mean Score with ERR |
|---|---|---|---|
| Extrovert vs Introvert | 0.68 | 0.67 | 0.51 |
| Intuition vs Sensing | 0.69 | 0.67 | 0.39 |
| Thinking vs Feeling | 0.87 | 0.67 | 0.79 |
| Judging vs Perceiving | 0.68 | 0.67 | 0.61 |

Since the primary objective is to rank predictions rather than classify them outright, AUC seemed as an appropriate metric since it assesses ranking performance. The evaluation focuses on the AUC (Area Under the Curve) metric derived from ROC (Receiver Operating Characteristic) curves for each class, indicating the models' ability to distinguish between the classes. Figure 4 shows that the performance of the model for each task indicating the model's ability to learn from the training data effectively, with values hovering above 0.5, higher than a random guess.

Table 2 presents the performance metrics for several classification tasks, focusing on the Geometric Mean Score with the best threashold and detailed metrics from the classification report imbalanced with binary prediction.

Table 2: Consolidated Classification Metrics for MBTI Dichotomies

| MBTI Dichotomy | Best Threshold | Accuracy | ROC-AUC Score | Avg Precision-Recall | G-Mean Score | F1-Score (avg/total) |
|---|---|---|---|---|---|---|
| Extrovert vs Introvert | 0.93 | 0.7804 | 0.68 | 0.67 | 0.51 | 0.72 |
| Intuition vs Sensing | 0.33 | 0.8599 | 0.69 | 0.67 | 0.39 | 0.80 |
| Thinking vs Feeling | 0.59 | 0.7925 | 0.87 | 0.67 | 0.79 | 0.79 |
| Judging vs Perceiving | 0.68 | 0.6594 | 0.68 | 0.67 | 0.61 | 0.64 |

The analysis of the model's performance across four tasks shows varied effectiveness, highlighted by geometric mean scores ranging from 0.37 to 0.79. The model excels in classification task S vs F, showing a good balance between identifying true

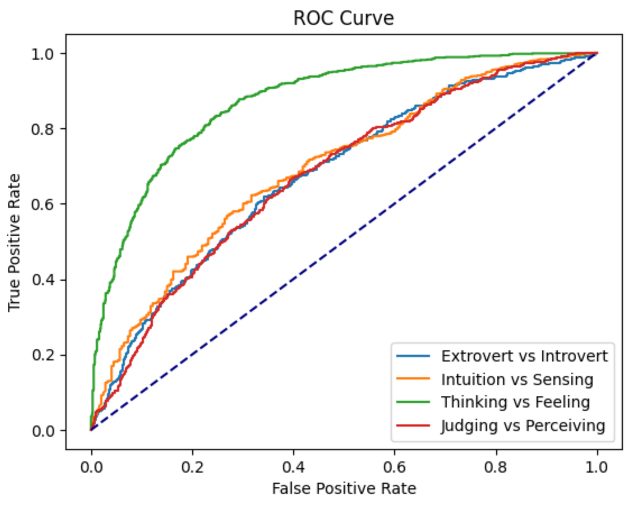
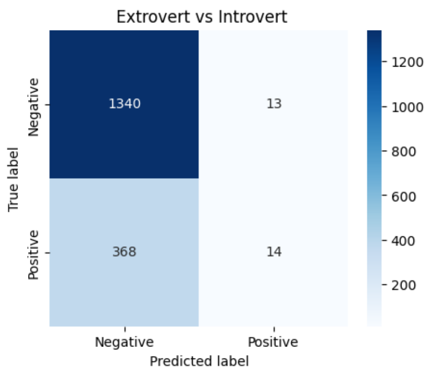


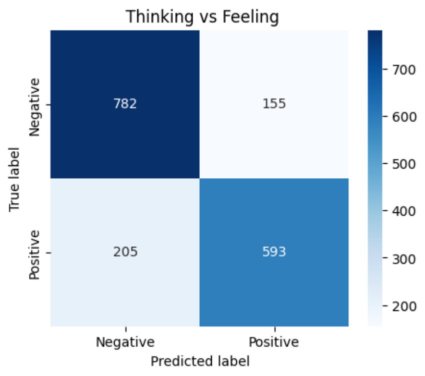
Figure 4: Area under the curve for binary classification on four dimensions

positives and negatives. However, Tasks E vs I and N vs S reveal a struggle to accurately identify positive instances, evident from high recall for the negative class but low recall for the positive class. Enhancing model sensitivity and considering adjustments in feature selection or modeling techniques may help achieve more balanced outcomes across all tasks.

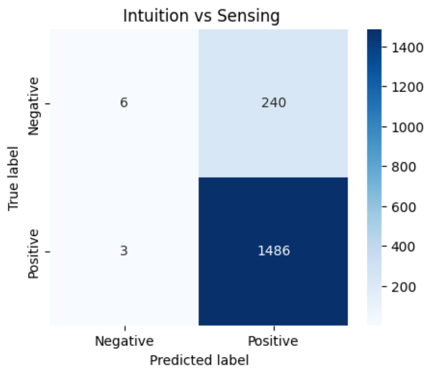
Confusion matrices are crucial for evaluating our neural network's performance on the MBTI dichotomies, revealing both the model's strengths and areas for improvement in classification tasks.



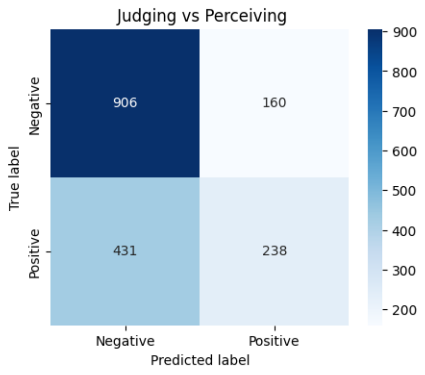
(a) E vs. I



(b) T vs. F



(c) N vs. S



(d) J vs. P

Figure 5: Confusion matrices for MBTI Dichotomy Classification Tasks

In Figure 5, the Extrovert vs Introvert confusion matrix indicates a strong model bias towards introvert classification. Meanwhile, the Judging vs Perceiving matrix shows a more balanced classification capability. The Intuition vs Sensing matrix demonstrates a high success rate in sensing classification, although with a tendency to overlook intuition. Lastly, the Thinking vs Feeling matrix suggests a fair level of discernment between thinking and feeling preferences, with relatively equal misclassification rates for both traits.

Despite the sequential model's demonstrated proficiency, alternative architectures incorporating LSTM or Bidirectional LSTM were also evaluated. These models achieved commendable accuracy on the test set, with LSTM models attaining around 65% compared to the 59% of our preferred convolutional approach. However, when assessing the Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC), the performance of LSTM-based models hovered around 0.5 for each MBTI dichotomy task, indicating a lack of improvement over random guessing in distinguishing between the classes. This discrepancy between high accuracy and lower AUC may suggest that the LSTM models are overfitting to certain dominant features in the training data, which do not generalize well to the test data. It emphasizes the importance of considering multiple metrics beyond accuracy when evaluating model performance, especially in the presence of imbalanced datasets. The AUC is particularly telling as it provides a measure of the model's ability to rank predictions correctly, regardless of the classification threshold, thereby offering a more robust evaluation of model performance in classifying each of the MBTI dichotomies.

### 5.3 GPT-4

We utilize the OpenAI API to send queries to the GPT-4 engine. The prompt used is:

```
Predict the MBTI type (only output 1 word) of the author of the following forum posts:
<ForumPosts>
```

where `<ForumPosts>` represents the concatenated forum posts of a user after preprocessing. The experiment was conducted on $15 \times 16 = 240$ data points, with 15 data points for each personality type. Despite the absence of fine-tuning, the accuracy was impressively high at 71%. The confusion matrix is illustrated in Figure 6.

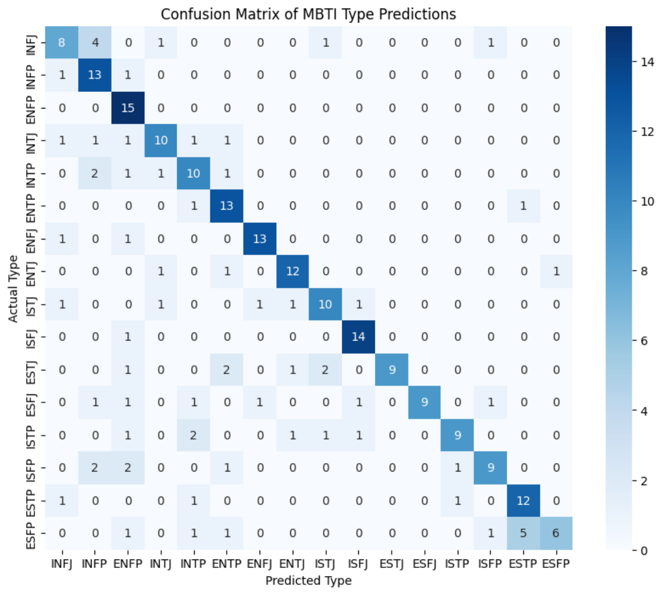


Figure 6: Confusion matrix of the GPT-4 model predictions

## 6 Conclusion

GPT offers high accuracy with minimal setup effort, suggesting that further hyperparameter tuning could enhance performance significantly. In contrast, neural networks demand more time for training and development. Meanwhile, Naive Bayes provides a quick and easy training process for decent prediction accuracy, though it appears to have a ceiling on its potential accuracy improvement.

## 7 Contributions

- Salma Zainana: The entire part on RNNs
- Daneshvar Amrollahi: The entire part on Naive Bayes and GPT

## References

Carlos Basto. 2021. Extending the abstraction of personality types based on mbti with machine learning and natural language processing.

Abhilash Datta, Souvic Chakraborty, and Animesh Mukherjee. 2023. Personality detection and analysis using twitter data.

Partha Kadambi. 2021. Exploring personality and online social engagement: An investigation of mbti users on twitter.

Sedrick Scott Keh and I-Tsun Cheng. 2019. Myers-briggs personality classification and personality-specific language generation using pre-trained language models.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. Gpt-4 technical report.