Understanding the Figurative Language Capabilities of Large Language Models

Alex Rashduni

May 8, 2023

Abstract

Large language models have made significant strides in natural language processing, but they still face difficulties in understanding figurative language. This essay explores the importance of scoreboarding and post-training techniques to improve the figurative language capabilities of large language models. We discuss the use of probability distributions and pseudo-labeling to develop common ground between language models over the scoreboard of a conversation, and propose a solution that involves training the model on a diverse dataset of figurative language, using post-training techniques to fine-tune the model, and evaluating its performance using scoreboarding. We also suggest several datasets that can be used to train and evaluate the model's performance in recognizing and generating figurative language in different contexts. Improving the figurative language capabilities of large language models has the potential to improve their communication abilities and lead to more advanced natural language processing systems.

1 Introduction

Figurative language is an essential part of human communication that allows us to express complex ideas, emotions, and experiences in creative ways. However, the ability to understand and use figurative language poses significant challenges for large language models (LLMs) such as GPT-4.

One of the primary difficulties that LLMs face in understanding figurative language is that it is often ambiguous and context-dependent. Figurative language relies heavily on the connotations of words and the cultural and social contexts in which they are used. For example, the phrase "it's raining cats and dogs" is a common English idiom that means it is raining heavily. However, a literal interpretation of this phrase would lead to confusion for LLMs, as it makes no logical sense.

Another challenge is that figurative language often involves the use of metaphor, simile, and analogy, which require a deep understanding of the relationships between different concepts.

For instance, the metaphor "life is a journey" is a common comparison that highlights the similarities between two seemingly unrelated concepts. However, LLMs may struggle to recognize the underlying connection and interpret the phrase literally.

Furthermore, figurative language often relies on cultural and historical references that may be unfamiliar to LLMs. For example, the expression "to be a Trojan horse" refers to the ancient Greek story in which Greek soldiers infiltrated the city of Troy by hiding inside a giant wooden horse. Without prior knowledge of this historical event, LLMs would not be able to understand the metaphorical meaning of the phrase.

Additionally, figurative language can be highly subjective and dependent on the individual's experiences and perspectives. For instance, the phrase "my heart is singing" is a common metaphor for feeling joyful and content. However, the emotions associated with the metaphor may differ from person to person, making it difficult for LLMs to interpret the phrase accurately.

Despite the current challenges, ongoing research aims to improve LLMs' ability to recognize and understand figurative language, which will enable them to better grasp human language and communication. In this essay, we discuss the potential applications of conversational scoreboarding and co-training in the development of an evaluation benchmark for understanding the figurative language capabilities of large language models.

2 A Game-Theoretic Approach

Game theory is a mathematical framework that is used to analyze decision-making in situations where multiple individuals or entities, called players, interact with each other. In game theory, the behavior of each player is modeled as a set of strategies that they can choose from, and the outcome of the game depends on the strategies chosen by all players.

Typically, a game is represented as a set of states and actions. The state of the game represents the current situation or condition, and the actions available to the players represent the choices they can make. The actions taken by the players result in a transition from one state to another.

Upon making a particular action, in games, a player is granted a certain reward or score. These are determined by the reward function, a mathematical function that maps states and actions to a numerical value, which represents the immediate feedback that the agent receives for its actions. The goal of the agent is to leverage an optimal policy, which is a mapping from states to actions, that maximizes the expected cumulative reward over time.

In some cases, the reward function can be explicitly defined by the designer of the game or the agent, but in other cases, it may be implicitly defined by the structure of the game. For example, in a two-player game such as chess, the reward function may be defined as a win/loss/draw outcome at the end of the game. In the situation of communicative games, the reward of the communication game is a measure of the shared interpretation between speakers and listeners. To analyze communication in a game-theoretic manner, we can

use a scoreboard to measure the past understanding and common ground of the current conversation.

2.1 Using Scoreboards to Measure Understanding

Conversational scoreboards, introduced by David Lewis in [1], are a tool used in conversation analysis to track and analyze the progress of a conversation. From a game-theoretic perspective, conversational scoreboards can be viewed as a way of measuring the interpretations of the participants in a conversation. One example set of parameters for a conversational game is listed below:

Players. The players of a game-theoretic conversation primarily consist of all participants in the conversation that provide and/or interpret utterances. These participants in a real-world setting can vary from a give-and-take conversational setting to a speaker-audience setting.

Actions. The actions taken are strings of utterances made by any player within the game.

Reward function. The reward function may include conveying information, expressing opinions, establishing social rapport, or achieving other objectives. In this particular example, we focus most on mutual agreement on interpretation between players.

From a game-theoretic perspective, conversational scoreboards can also be used to evaluate the outcomes of the conversation and identify optimal speech strategies for the participants. For example, a conversational scoreboard might reveal that in a professor-student "lecture hall" dynamic, asking more questions leads to better outcomes, or that longer turns are more effective at conveying information. By identifying these patterns, chatbot and language model developers can develop better models of conversational behavior.

2.2 Applications to Large Language Models

Conversational scoreboards can be used to evaluate the figurative language capabilities of large language models by analyzing their responses to various types of figurative language in a conversation. For example, under a conversational task with many metaphors and idioms, a conversational scoreboard can be used to track the performance of the language model in understanding and generating appropriate responses.

By analyzing the conversational scoreboard data, we can gain insights into the strengths and weaknesses of the language model in handling figurative language. They can also use the data to identify areas where tested language models could be improved, such as by incorporating more cultural or contextual knowledge.

Algorithm 1 Co-training algorithm

input $\mathcal{U} = \{x_n\}_{n=1}^{U}$ unlabeled examples
input $\{(x_j, y_j)\}_{j=1}^{k}$ labeled examples (optional)
input initial coverage β, coverage increase β'
  $h_0 \leftarrow \texttt{InitClassifier}(\phi_0)$
  for t in $\{0, \ldots, T-1\}$ **do**
    $\tilde{\beta} \leftarrow \beta + t\beta'$
    // `GetConfData`* defined in Algorithms 2, 3
    $L_0^t \leftarrow \texttt{GetConfData}^* \left(\mathcal{U};\ h_0, \phi_0, \tilde{\beta}\right)$
    $h_1 \leftarrow \texttt{Train}(\phi_1, L_0^t)$
    $L_1^t \leftarrow \texttt{GetConfData}^* \left(\mathcal{U};\ h_1, \phi_1, \tilde{\beta}\right)$
    $h_0 \leftarrow \texttt{Train}(\phi_0, L_1^t)$
  end for
  return (h_0, h_1)

Figure 1: The co-training algorithm, as shown in [2].

3 Machine Learning Principles

In this section, we lay out the machine learning principles and applications used to develop the experiment laid out in Section 4.

3.1 Pseudo-Labeled Data & Co-Training

Pseudo-labeling is a semi-supervised machine learning technique that involves using the predictions of an existing model to generate estimated labels for a set of unlabeled data. The approach is based on the assumption that the model can make reasonably accurate predictions on unlabeled data, and those predictions can be used to create new labeled data. This process's applications in co-trained large language modeling was introduced in by Lang et al in [2], with the general co-training algorithm shown in Figure 1.

The process typically involves first training a model on a small labeled dataset. The trained model is then used to generate predictions for a larger set of unlabeled data. The most confident predictions are selected as new labels for the corresponding data points, shown in Figure 2. These newly labeled data points are then added to the original labeled dataset, and the model is retrained on the combined labeled data.

This process is iterative, with the newly labeled data points being added to the training set in each iteration. The central idea is that by incorporating the new labeled data into the training set, the model's accuracy on both the labeled and unlabeled data may improve.

Pseudo-labeling is particularly useful in situations where labeled data is scarce, and acquiring additional labeled data is expensive or time-consuming. By leveraging the predictions of an existing model to generate new labeled data, it can help to improve the accuracy of machine

Algorithm 2 `GetConfDataMC`

input $\{x_n\}_{n=1}^U$ unlabeled examples
input model h_i, view ϕ_i
input coverage fraction $\tilde{\beta}$
input minimum class percentage γ
  // compute pseudolabel and score for each example
  for n in $\{1, \ldots, U\}$ **do**
    $o_n = h_i(\phi_i(x_n))$ (note $o_n \in \mathbb{R}^{numlabels}$)
    $\hat{y}_n \leftarrow \text{argmax}_l\, o_{nl}$
    $s_n \leftarrow \max_l o_{nl}$
  end for
  $L \leftarrow \emptyset$

  // first, select top γ% for each class by score
  $ms \leftarrow \lfloor \gamma \tilde{\beta} U \rfloor$ // min num points to select
  for l in $\{1, \ldots, numlabels\}$ **do**
    $\mathcal{I}_l = \{(x_n, \hat{y}_n, s_n) : \hat{y}_n = l\}$
    // sort $\mathcal{I}_l$ by score s_n (ascending)
    $S_l \leftarrow$ `Sort`($\mathcal{I}_l$, key=lambda q: q[2])
    // add top α% to L (read off end of S_l)
    $L \leftarrow L \cup S_l[-ms:]$
  end for

  // now select the rest of the points
  $rs \leftarrow \lceil \tilde{\beta} U \rceil - |L|$ // num remaining points to select
  $\mathcal{I} = \{(x_n, \hat{y}_n, s_n)\}_{n=1}^U$
  $\mathcal{I} \leftarrow \mathcal{I} \setminus L$ // don't select twice
  $S \leftarrow$ `Sort`($\mathcal{I}$, key=lambda q: q[2])
  $L \leftarrow L \cup S[-rs:]$

  // chop off score and return point + pseudolabel
  $L \leftarrow \{(x_n, \hat{y}_n) | \exists s_n : (x_n, \hat{y}_n, s_n) \in L\}$
  return L

Algorithm 3 `GetConfDataCS`

input $\{x_n\}_{n=1}^U$ unlabeled examples
input model h_i, view ϕ_i
input coverage fraction $\tilde{\beta}$
  // compute pseudolabel and repr. for each example
  for n in $\{1, \ldots, U\}$ **do**
    $o_n = h_i(\phi_i(x_n))$ (note $o_n \in \mathbb{R}^{numlabels}$)
    $\hat{y}_n \leftarrow \text{argmax}_l\, o_{nl}$
  end for
  $L \leftarrow \emptyset$
  for y in $\{1, \ldots, numlabels\}$ **do**
    $\hat{P}_y = |\{n : \hat{y}_n = y\}|/U$
  end for

  // compute 20 nearest neighbors for each ex.
  for u in $\{1, \ldots, U\}$ **do**
    $N(u) = \text{NN}_{20}(\phi_i(x_u), \{\phi_i(x_v)\}_{v=1}^U)$
    for v in $N(u)$ **do**
      $w_{uv} = 1/(1 + \|\phi(x_u) - \phi(x_v)\|_2)$
      $I_{uv} = \mathbb{I}[\hat{y}_u \neq \hat{y}_v]$
    end for
    // now compute cut statistic
    $J_u = \sum_{v \in N(u)} w_{uv} I_{uv}$
    $\mu_u = (1 - \hat{P}_{\hat{y}_u}) \sum_{v \in N(u)} w_{uv}$
    $\sigma^2 = \hat{P}_{\hat{y}_u}(1 - \hat{P}_{\hat{y}_u}) \sum_{v \in N(u)} w_{uv}^2$
    $s_u = \frac{J_u - \mu_u}{\sigma}$
  end for
  // now sort by statistic and return top data
  $\mathcal{I} \leftarrow \{(x_n, \hat{y}_n, s_n)\}_{n=1}^U$
  $S \leftarrow$ `Sort`($\mathcal{I}$, key=lambda q: q[2])
  $ns = \lfloor \tilde{\beta} U \rfloor$
  $L \leftarrow S[:ns]$
  $L \leftarrow \{(x_n, \hat{y}_n) | \exists s_n : (x_n, \hat{y}_n, s_n) \in L\}$
  return L

Figure 2: The "Get Confident Data" Algorithm, referred to in Figure 1, as shown in [2]. the "MC" version on the left gets confident data using model confidence as a measure, whereas the "CS" version uses cut-statistics. For the sake of this essay, we focus on the model confidence variation.

learning models while reducing the cost and effort required to obtain labeled data.

This process is improved by the use of co-training, in which multiple models are initialized and trained on the labeled data. From there, the confidence in pseudo-labeled data is generated through the consensus of the models. This approach is based on the assumption that different features can provide complementary information about the data, and by training multiple models on different sets of features, the accuracy of the model can be improved.

The co-training process typically involves dividing the labeled dataset into two or more subsets, each containing different features. Each model is trained on one of these subsets of features, and then the models exchange their predictions on the unlabeled data. The most confident predictions from each model are used to create new pseudo-labeled data points, which are then added to the original labeled dataset. The process is iterated with the newly labeled data points being used to retrain the models.

3.2 Common Ground: Similarity Metrics

In the co-training process, model probability distribution convergence can be used to assess the similarity between the probability distributions of the predictions made by different models. The convergence of probability distributions is an important metric because it indicates whether the models are learning complementary information or if they are making redundant predictions.

Convergence metrics can be used to assess the similarity of the probability distributions generated by different models. These metrics can be used to compare the distributions of the same model trained on different subsets of features or to compare the distributions generated by different models.

Examples of convergence metrics include:

Kullback-Leibler (KL) Divergence. KL divergence measures the amount of information lost when using one probability distribution to approximate the other. It is defined as the sum of the probability of each event multiplied by the logarithm of the ratio of the probabilities of that event under the two distributions. KL divergence is asymmetric, meaning the result will be different depending on which distribution is used as the reference. The KL divergence value can be interpreted as a measure of the distance between the two distributions, where smaller values indicate greater similarity.

$$D_{\mathrm{KL}}(P||Q) = \sum_{x \in X} P(x) \log \left(\frac{P(x)}{Q(x)} \right)$$

Jensen-Shannon (JS) Divergence. JS divergence is a symmetric version of KL divergence. It measures the average divergence between two probability distributions and is defined as half the KL divergence between the two distributions, plus half the KL divergence between the two distributions' average. The JS divergence value ranges from 0 (identical distributions) to 1 (completely different distributions).

$$D_{\mathrm{JS}}(P||Q) = \frac{1}{2}D_{\mathrm{KL}}(P||Q) + \frac{1}{2}D_{\mathrm{KL}}(Q||P) = D_{\mathrm{JS}}(Q||P)$$

Total Variation Distance (TVD). TVD is a measure of the distance between two probability distributions based on the difference in probability assigned to each event. It is defined as the sum of the absolute differences between the probabilities assigned to each event. TVD ranges from 0 (identical distributions) to 1 (completely different distributions).

$$\delta(P, Q) = \sum_{x \in X} |P(x) - Q(x)|$$

By monitoring the convergence of the probability distributions generated by different models, co-training can be used to assess whether the models are learning complementary information. If the distributions are similar, it may be an indication that the models are making redundant predictions, and more robust feature representation may be needed. If the distributions are dissimilar, it may be an indication that the models are learning complementary information, and the co-trained models can be ensembled to improve model accuracy.

3.3 Applications to Conversational Data

In the scoreboard approach over conversational data, we can use pseudo-labeling to iteratively update the probability distributions representing the state of the conversation for each large language model. This can help bring the probability distributions closer together and improve their similarity, indicating a greater level of common ground between the models. To implement this, we can apply the co-training algorithm discussed in Figure 1.

To apply pseudo-labeling, we need a way to represent the problem of understanding figurative language. One possible approach is to use one of the language models as an annotator and generate labels for the other model's generated text. For example, we can use one model to generate a response to a prompt and ask the other model to annotate the response with the intent of the response or the topic discussed. The annotated data points can then be used as labeled data for pseudo-labeling.

Pseudo-labeling can help improve the performance of large language models and bring them closer together in terms of their understanding of the conversation. By using one model to annotate the generated text of the other model, we can help develop a shared understanding of the context and improve the models' ability to generate coherent and relevant responses.

4 Proposed Experiment

In this section, we bring together the theory of machine learning and conversational scoreboarding to develop an experiment to determine the figurative language capabilities of large

language models:

1. Collect a large dataset of texts that contain various types of figurative language, such as idioms, metaphors, similes, and sarcasm. This dataset should cover a wide range of topics and be representative of the language and styles used in everyday communication. Refer to Section 4.1 for the specifics of acquiring such a dataset.

2. Train a large language model on this dataset using post-training techniques, such as contrastive learning or language modeling objectives. This will help the model learn to recognize and generate various types of figurative language and improve its overall language understanding capabilities. Although any pretrained language model can be used in theory, a more recent and advanced one such as GPT-4 should provide the most cutting-edge results.

3. Use scoreboarding to evaluate the performance of the model on a set of prompts that contain various types of figurative language. This evaluation should focus on the model's ability to recognize and interpret the figurative language in the prompt and generate a coherent and relevant response.

4. Analyze the results of the scoreboarding evaluation to identify areas where the model is struggling to understand or generate figurative language. Use this information to guide further post-training, either by fine-tuning the model on specific types of figurative language or by incorporating additional data to address specific weaknesses.

5. Repeat steps 3-4 until model performance converges.

By incorporating scoreboarding into the post-training process, we can ensure that the model is not only improving its ability to recognize and generate figurative language but also its ability to do so in a way that is relevant and coherent within the context of a conversation. This can help address some of the challenges associated with evaluating the performance of large language models on figurative language, such as the lack of objective metrics or the subjectivity of human evaluation.

4.1 Data Sources

Naturally, to determine the figurative language capabilities of language models, a massive corpus of labeled data is necessary. Despite that, the vast majority of natural language data that exists is unlabeled. Therefore, we propose developing large language models that are pretrained on existing figurative language datasets, and then using co-training to develop a pseudo-labeled corpus over Common Crawl, a massive dataset of web pages. The Common Crawl dataset naturally contains many different forms of figurative language, as it is scraped from the Internet, and thus should suffice in understanding language models' figurative language capabilities.

5 Conclusion

In this essay, we discussed the challenges that large language models face in understanding figurative language, as well as the importance of scoreboarding and post-training techniques to improve their figurative language capabilities. We also explored the use of co-training and pseudo-labeling to develop common ground between language models over the scoreboard of a conversation.

To improve the figurative language capabilities of large language models, we proposed a solution that involves training the model on a diverse dataset of figurative language, using post-training techniques to fine-tune the model, and evaluating its performance using scoreboarding. We also discussed several datasets that can be used for this experiment, including Common Crawl, the Figurative Language Dataset, Sarcasm Detection Dataset, Sentiment Analysis Datasets, and Creative Writing Prompts.

Improving the figurative language capabilities of large language models is an important area of research that has the potential to improve their language understanding and communication capabilities. By incorporating scoreboarding and post-training techniques into the training process and using diverse datasets to evaluate the model's performance, we can help address the challenges associated with understanding figurative language and improve the overall performance of large language models.

References

[1] Lewis, D. 1979. Scorekeeping in a Language Game. In 1979, *Journal of Philosophical Logic*, 8: 339-359.

[2] Lang, H., et al, 2022. Co-training Improves Prompt-based Learning for Large Language Models. 2202.00828 (arxiv.org).