ECE 1786 Lecture #7

Work-in-Flight:

- Project approval-in-principle - due Oct 24 at this form: https://forms.office.com/r/gHVaKngvds
- Proposal Document and Slides both due November 4 @9pm
- Will announce scheduling of presentation & your peer review after all AIP
- Note: In the October 22nd lecture period all 5 course TAs will be in the lecture room to help you with project ideation and scoping; you can also reach out to them by email
- Assignment 4 - Decoding for Generation, Prompt Engineering for Different Tasks, Agentic Approaches; not yet released, due Wednesday November 13

Last Day: 1. Language Generation & Decoding using Transformers
2. Project Ideation and Proposal

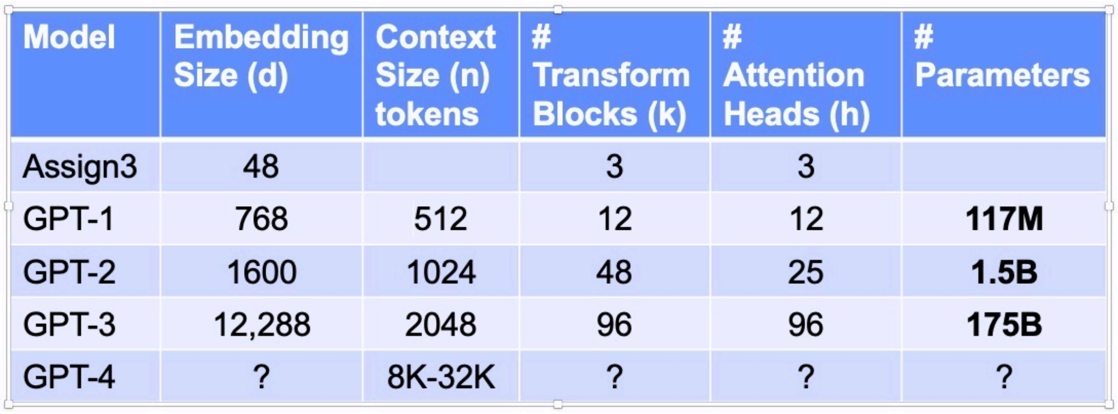
Today:

- LLM Scaling
- Why are the big LLMs so smart? Why do they do what you ask?
- Zero-Shot Prompting;
- Prompt Engineering - for Classification and Generation
- Chain of Thought Prompting;
- System Message
- API use of OpenAI; Class 2 Projects and Software Frameworks
- Retrieval Augmented Generation
- Tokenization

LLM Scaling

- The remarkable success and capabilities of the big models - GPT3.5 and 4 and later, and its cousins from Google, Anthropic, Meta, Cohere, Pi and now many more clearly comes from scaling up the size of the models, the amount of data and hence the amount of training.

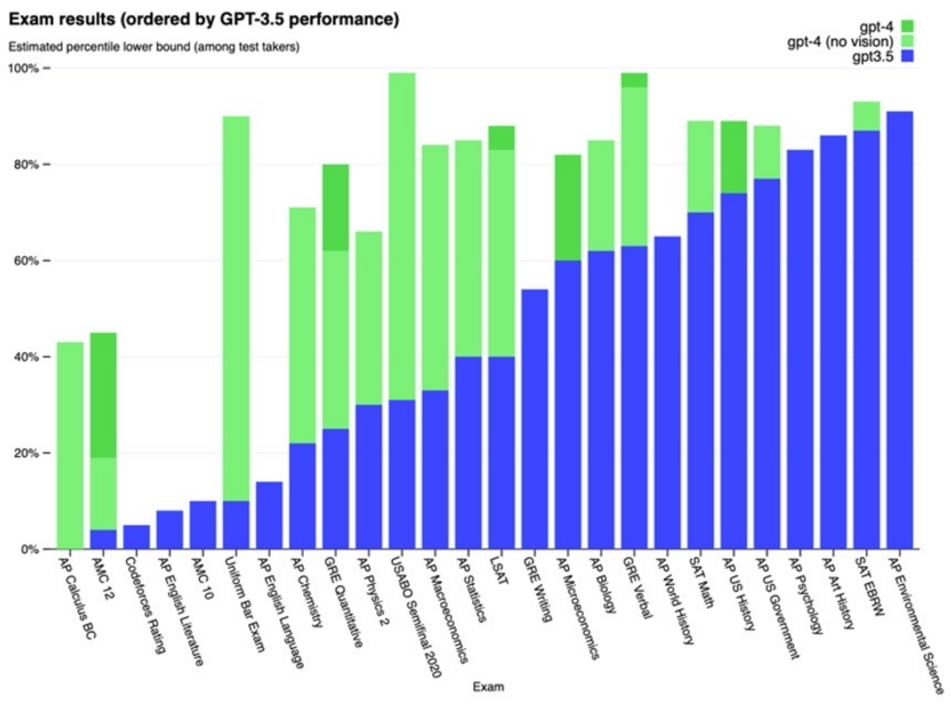
The scaling looks like this:



Model	Embedding Size (d)	Context Size (n) tokens	# Transform Blocks (k)	# Attention Heads (h)	# Parameters
Assign3	48		3	3	
GPT-1	768	512	12	12	**117M**
GPT-2	1600	1024	48	25	**1.5B**
GPT-3	12,288	2048	96	96	**175B**
GPT-4	?	8K-32K	?	?	?

- Anecdotal evidence for this:
  - 2018: GPT-2 spoke coherently, for the first time
  - 2020: GPT-3 began to do what it was asked/shown
  - 2022: GPT-3.5 did what we asked
  - 2023: GPT-4 did what we asked really really well!
  - 2024: GPT-4o ++ multi-modal, cheaper, faster

Some concrete evidence: the improvement from GPT3.5 to GPT4 at passing the U.S. Law Bar exam, and many other AP courses from high school - History, Government, Psychology:



Now, let's bring some intuition behind all those numbers, by asking two questions:

1. Why is it so smart? How does it know so much?
    - on the one hand, predicting the next word given just a few words might be easy:
        - "The door was __?__" (open, closed, shut, ajar)
    - on the other hand, predicting the next word from many - such as a complex surgery performed by a dental surgeon:
    - "... Once the metal implant post is placed in your jawbone, osseointegration begins. During this process, the jawbone grows into and __?__"
    - Right answer: "unites with the surface of the dental implant."
    - So, a key observation here is that you have to know a lot to predict the next word across all those different, highly complex disciplines
    - If you watch this: https://www.youtube.com/watch?v=0GKou6lSfi0 you'll see Illya Sutskever, formerly from OpenAI, talking about the higher-level 'world model' that the large-scale training creates, with not just lucky next-word prediction, deeper comprehension of the world
        - also, points out that this has happened across many different disciplines, much more than one person could ever do

2.Why does chatGPT/GPT-4 do what you ask it to do?

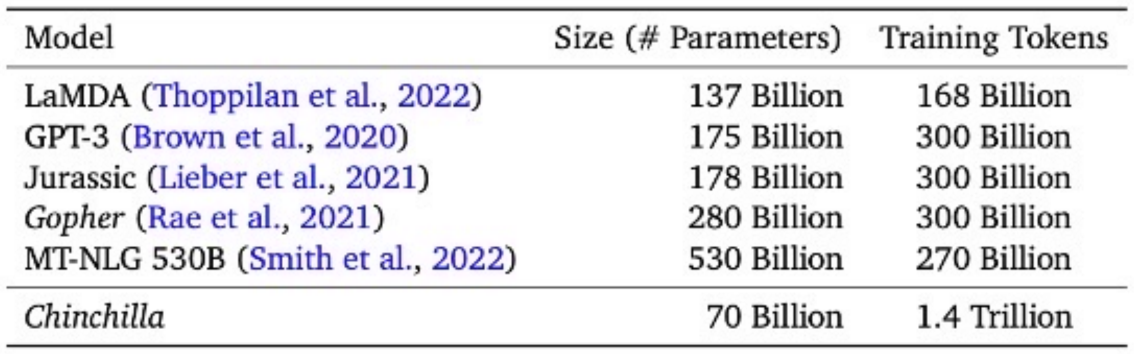
My Answer: if the model is very good at predicting the next word, based on the previous words, (which include what you asked), then the right next words, predicted auto-regressively, is to do what was asked. (Within the abilities of the models).

3. How much training data is actually needed to achieve these things?

- in this course so far we have described the Transformer model in detail, and provided some sense of the specific computations

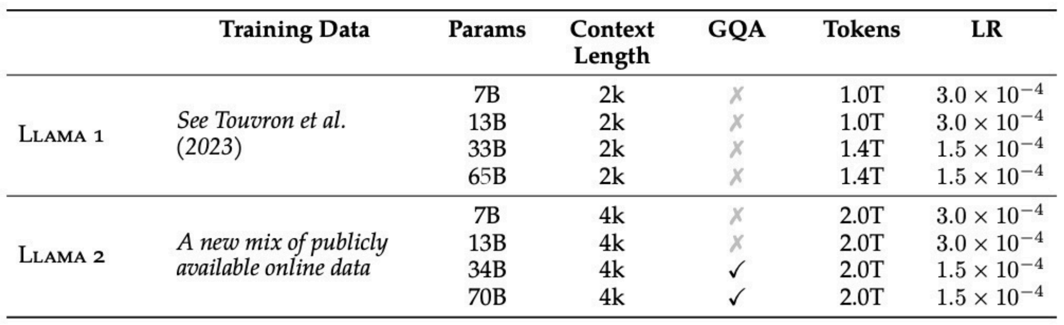
- However, OpenAI, in the GPT-3 time frame, got the ratio of amount of training data: model size wrong.

- The paper: "Training Compute-Optimal Large Language Models" Hoffmann et. al, March 2022 did experimental work, and showed that more data was needed for the size of model, per this table:



Model	Size (# Parameters)	Training Tokens
LaMDA (Thoppilan et al., 2022)	137 Billion	168 Billion
GPT-3 (Brown et al., 2020)	175 Billion	300 Billion
Jurassic (Lieber et al., 2021)	178 Billion	300 Billion
Gopher (Rae et al., 2021)	280 Billion	300 Billion
MT-NLG 530B (Smith et al., 2022)	530 Billion	270 Billion
Chinchilla	70 Billion	1.4 Trillion

- More recently, Meta has released the open source Llama, Llama2 and Llama3 models, with similar (to Chinchilla) ratio of tokens:model size, in the paper "Llama 2: Open Foundation and Fine-Tuned Chat Models," Touvron et. al:



	Training Data	Params	Context Length	GQA	Tokens	LR
Llama 1	*See Touvron et al. (2023)*	7B	2k	✗	1.0T	3.0×10^{-4}
		13B	2k	✗	1.0T	3.0×10^{-4}
		33B	2k	✗	1.4T	1.5×10^{-4}
		65B	2k	✗	1.4T	1.5×10^{-4}
Llama 2	*A new mix of publicly available online data*	7B	4k	✗	2.0T	3.0×10^{-4}
		13B	4k	✗	2.0T	3.0×10^{-4}
		34B	4k	✓	2.0T	1.5×10^{-4}
		70B	4k	✓	2.0T	1.5×10^{-4}

Table 1: Llama 2 family of models. Token counts refer to pretraining data only. All models are trained with a global batch-size of 4M tokens. Bigger models — 34B and 70B — use Grouped-Query Attention (GQA) for improved inference scalability.

- This evidence also suggests that the earlier pre-trained models that are much smaller, trained with less data, "know" far less. For example BERT and RoBERTa, widely used for classification and generation tasks are orders of magnitude smaller on both fronts - e.g. RoBERTa is 125M parameters, trained on 160B tokens

7.5

- BERT and RoBERTa are trained in a different manner: not to predict the next token, but to predict missing tokens within a few sentences - called 'Masked Language Modeling' because missing tokens are masked with a special token
    - I don't know if you need to know more, less or the same to predict missing tokens rather than the next token
    - Do suspect that size of model and amount of training trumps this question
    - => There is lots of code using BERT & RoBERTa to do things, but I think you'll be better off (and strongly recommend) using a pre-trained GPT-2 (or Llama2 if you're up for that) & put use a classification or generation head on it - they know a lot more!!
    - => DO NOT, in your project, as I've already mentioned, under any circumstances, use an RNN (like LSTM or GRU) and train it from scratch, in your project, without speaking with me. (Doing so means you haven't been paying attention to the lectures)

- We will now discuss the differences between the 1) old way of using these smaller models to do this - called 'Class 1' in the project discussion/documents and 2) the newer way, using prompting of the very large models called 'Class 2'

- Consider the problem of "question answering", here is an example, from A4:

Context: Age, diameter, height, radial growth, geographical location, site and growing conditions, silvicultural treatment, and seed source, all to some degree influence wood density. Variation is to be expected. Within an individual tree, the variation in wood density is often as great as or even greater than that between different trees (Timell 1986). Variation of specific gravity within the bole of a tree can occur in either the horizontal or vertical direction

Question: Which part of a tree can have vertical or horizontal variation in its specific gravity?

Answer: The bole.

This and questions like this are part of many datasets that have long been in the NLP arena. The goal is to make models that get the correct answers.

Previously it was common to train a BERT/RoBERTA to do this, and they were surprisingly successful; in Assignment 4 we'll do it with a "Zero-Shot" Prompt to GPT-4 and see that it works without any training at all.

This is now possible because, as discussed, these big models will attempt to do anything you ask them to do, and they can do many things based on text.

Indeed an interesting questions is what can't they do? It has become know that it does matter how you ask the model what you want. This has become known as the field of Prompt Engineering. It is the skill/knowledge of writing clearly what you want the model to do, but also comes with some basic insights that I will now try to convey.

Example 1: Make a Classifer

Prompt: Determine if the following statement is objective or subjective. If it is subjective output "SUB". If it is objective output "OBJ".

the script is a tired one , with few moments of joy rising above the stale material .

Output: SUB

DEMO more from Assignment 2 on this prompt; show different prompt styles (remove SUB/OBJ), try o1

Example 2: . Recall previous descriptions of my research, in which we think about reflective listening, and the therapeutic act of making a reflection.

Generative Prompt: Prompt for generating reflections, which are therapeutic statements that a therapist might give in a conversation about a smoking habit (M. Abdelwahab):

The following is an interaction between you and a user. You are a therapist and the user is someone having smoking issues. Give a SHORT reflection to the user's response. The reflection must be a plausible guess or assumption about the user's underlying emotions, values, or chain of thought. The reflection can relate the response to one of the user's good attributes. The reflection must not just be a rephrasing of the user's response. Be creative with your use of prefaces in the reflection, don't always use "it sounds like" or "it seems like" or "you". The reflection must be very short.

This prompt would come first, and then the actual 'input' to the system comes next, as follows:

therapist: To start, what is the thing you like most about smoking?

client: the heavy taste of smoking and makes me feel calmer.

This is the 'completion' of the above prompt (all the green text):

therapist: You may find comfort and a sense of relaxation in the strong sensory experience smoking provides.

A second example input and output from the same prompt (green in, red out)

therapist: Now, what is the thing you like least about smoking?
client: leaves scent on hair and clothes.
therapist: It seems that you value cleanliness and may feel frustrated by the lingering odor of smoke.

Here is a method for prompt engineering: it is both iterative and experimental.

1. Write down your criterion for what makes the output acceptable, in English. This may involve looking up the definition of one or more words of the goal, or exploring other online resources so that you clearly understand it. This applies to both generation and classification.

2. Draft a Prompt that uses that criterion to direct the model to generate what you want.

3. Using just one input example (which in classical machine learning training would be called a training example) run the prompt and the example to see how well it works, in the OpenAI playground, on the model.

4. Keep evolving the prompt, using English, to make it work perfectly. This means changing the prompt to correct anything that is wrong in the output. However, do not use language that is specific to the input data, as that won't generalize. Notice that the concept of generalization will In the prompt engineering world, it will be helpful to cultivate an ability to write clearly, with good use of language and meaning of words.

5. Once you've made it work for one input example, make it work for two, using the same method - using a prompt with general words, not specific to example, but correcting any issues seen on the second example. See the above examples for some insights.

6. Then, try the prompt on five more training input examples all at once, and label the outputs as good/not good with respect to your criterion. Evolve the prompt to succeed on all five.

7. Test. Run the prompt on your 20 hold-out test set. Measure success rate yourself, i.e. with human labelling. Report your success rate.

8. If the success rate is below 100% you could choose to iterate once again, adjusting the prompt to correct the non-successful outcomes.

7-9

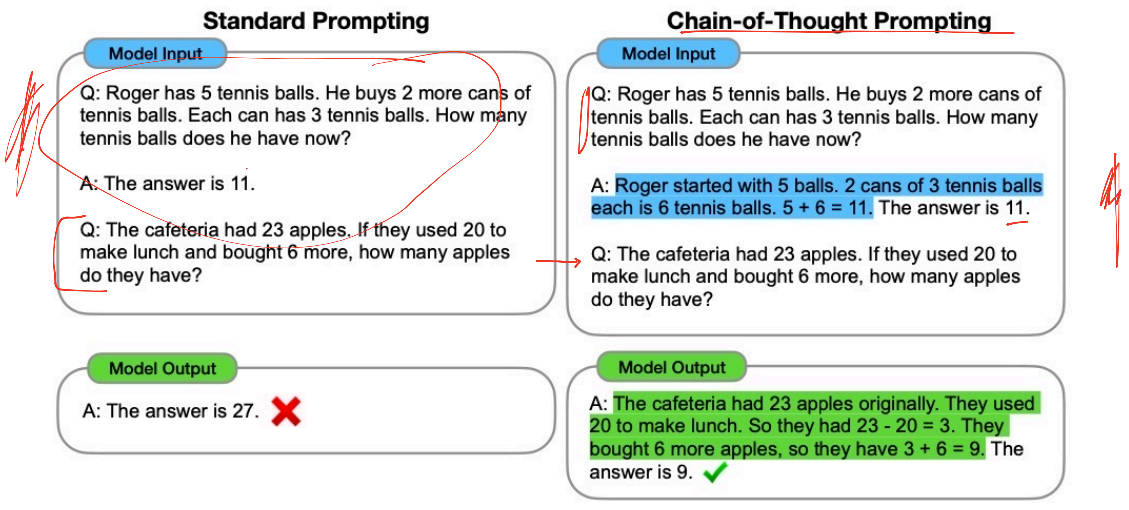
In our work above, our human labeling achieve 98% successful with respect to adherent to the therapeutic method (Motivational Interviewing). [although more subletly as to whether it was complex - 87% vs. Simple)]

Other sources of insight on Prompt Engineering

Read this Medium post: https://www.dropbox.com/scl/fi/niszm85qnsaf7wtwe5iys/PromptEngineering_Medium.pdf?rlkey=shsfwv64na1w11hhs6oru7vdp&dl=0

Zero-Shot vs. Few-Shot Prompting

- You are familiar with 'zero-shot' prompting — you just tell the model what you want to do with direct instructions
- Few-Shot prompting means you give instructions or a question, but also show an example of how to do it
- Example on left side below



Standard Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ✗

Chain-of-Thought Prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had 23 - 20 = 3. They bought 6 more apples, so they have 3 + 6 = 9. The answer is 9. ✓

Chain of Thought Prompting

- One of the more important observations from the literature is the idea that asking the model for its "Chain of Thought" reasoning for giving an answer makes it give a better answer: https://arxiv.org/abs/2201.11903

- Above example on the right side shows an example **few shot way** to evoke a model to show its steps

- There is another more direct way: the zero-shot instruction simply says:
→Show your steps towards an answer, or think step-by-step:

e.g. If John has 5 pears, then eats 2, and buys 5 more, then gives 3 to his friend, how many pears does he have? (Add: Let's think step-by-step.); Note that GPT-4 doesn't seem to need this, nor does GPT-4o; GPT-4o1 goes even farther, and automatically cycles through the output and tries to improve it, in a built-in agentic approach.

- Apparently, so does 'take a deep breath'

Discuss/Demo System Message on chat interface of GPT Playground.
  - For the biggest models on OpenAI, the command you want to be operative throughout, should now be placed separately in the 'System Message'
  - In our research this has been shown to work better than putting in the regular stream

API Access to GPT-4

- In assignment 4, you'll be using GPT-4o to do much of it, to get a handle on zero-shot prompting engineering.
- You'll also be asked to use the API to access GPT-4, as this may well be something you'll be doing in your project - show code as given
- This should make it clear that GPT-4 is very powerful, and I think you should try to use that power to do something, in your project, that makes use of it
- At the same, time, what was once difficult - requiring lots of training and data and effort, is no longer hard.

Retrieval Augmented Generation

- Idea here is two-fold: 1) That you put into the context (in input tokens to the model), all the information that might be what is the answer to a question, and After that you ask the question; These models can look for the answer in the context
- However, there might be too much information to draw from, so instead create a database that is searched to provide that first chunk of context.
    - Use neural techniques to store and search the database
    - Encode each entry in the data base using, say a transformer encoding of a sentence or multiple sentences - i.e. get an embedding of the sentences
    - Also encode the query/question with the same method
    - Use cosine distance between the query and the the data base to select the appropriate context
- Called 'Retreival Augmented Generation'

Tokenization

Tokenization is the process of breaking up in the input words in the input sentences into separate tokens. You've seen simple versions of this process in Assignments 1 and 2. I'm going to cover the main method used in Transformers of tokenization, because it lets me speak about how some of the knowledge of the models is in the embeddings, and some is in the model.

Determining the set of possible tokens is typically done on a specific Corpora, and once it is done it is fixed - the number of tokens gives the number of outputs of the transformer - those probabilities. It seems that the tokenizer for GPT-2 is the one that continues to be used, perhaps.

Because those tokens also include all symbols and letters, there is no input sequence of text words & symbols that cannot be reduced to a sequence of known tokens.

The set of tokens was determined using the Byte-Pair Encoding algorithm:
- See Jurafsky Section 2.5 (specifically Section 2.5.2) for description
- more recently, OpenAI has revised their tokenization (without disclosing it) and is said to be more efficient - using fewer tokens overall for the same corpus

Some of the chosen tokens are:
- Full words, or part of words
- Some tokens are meant to attached to other tokens to form full words, and some that are not

Consider some acronyms:
- lol (laugh out loud) appears to have its own token
- idk (i dont' know) does not, and is represented as i-d-k from the letter tokens

So, where does the knowledge about idk reside? It must be inside the model

What does the embedding of lol look like? I'd be curious as to what its closest words are.

——————

EXTRA: Prompt for Detection of Good Quality Reflections (J. Zhu)
- from my research; as an example of a more lengthy, thought-through prompt

Decide, in "True" or "False", whether the "reflection" sentence in the following smoking-related conversation is good.

Please refer to the following operational definition of a reflection in the context of Motivational Interviewing (MI):

Reflective listening statements are made by the clinician in response to client statements. A reflection may introduce new meaning or material, but it essentially captures and returns to clients something about what they have just said. Reflections are further categorized as simple or complex reflections.

Simple reflections typically convey understanding or facilitate client–clinician exchanges. These reflections add little or no meaning (or emphasis) to what clients have said. Simple reflections may mark very important or intense client emotions, but do not go far beyond the client's original intent in the statement.

Complex reflections typically add substantial meaning or emphasis to what the client has said. These reflections serve the purpose of conveying a deeper or more complex picture of what the client has said. Sometimes the clinician may choose to emphasize a particular part of what the client has said to make a point or take the conversation in a different direction. Clinicians may add subtle or very obvious content to the client's words, or they may combine statements from the client to form complex summaries.

Here are some additional hard constraints for a reflection to be good:

A reflection must be a statement rather than a question.
A reflection must not be MI-inconsistent in the following ways: Confronting the person by disagreeing, arguing, correcting, shaming, blaming, criticizing, labeling, ridiculing, or questioning the person's honesty, or directing the person by giving orders, commands, or imperatives, or otherwise challenging the person's autonomy.
A reflection must not move people to the wrong direction in terms of smoking cessation. If the client has expressed their will towards quitting smoking, do not overstate their statement; if the client has expressed their will against quitting smoking, do not understate their statement.
A reflection must not be factually wrong about smoking.
A reflection must be grammatically correct.
A reflection must be relevant to the conversation.
Given all the context above, please make an informed decision on whether or not the reflection is good. If the reflection is good, output "True". Otherwise, output "False", and output an explanation that includes which properties it has that makes it not good.