Introduction to Large Language Models

STOR 566 Guest Lecture

Guanting Chen

Assistant Professor

Department of Statistics and Operations Research

University of North Carolina at Chapel Hill

Introduction to Large Language Models

In this course, we will talk about

- Architectures of large language models (LLMs) and their differences
- Evolution of the LLMs: Pre-training, fine-tuning, RLHF, and Reasoning

A Simplified Intro to Transformer and LLMs

We will review (optional)

- Attention-based encoder and decoder

- Transformers

We will and then discuss

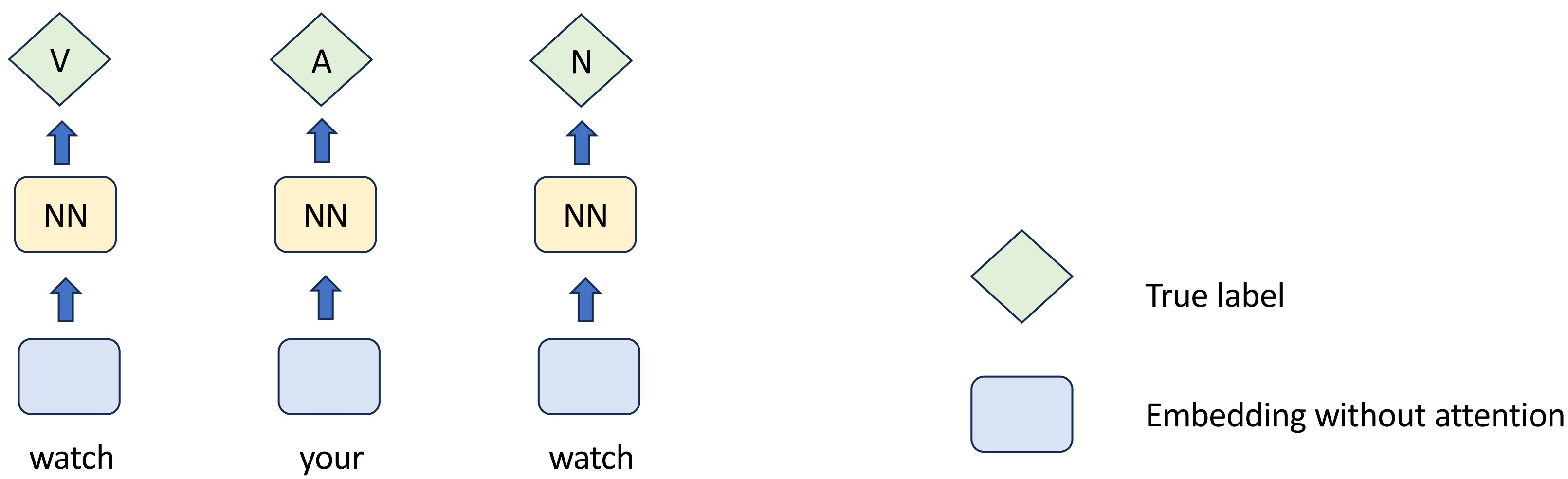
- Bert (optinal) and GPT

Embedding

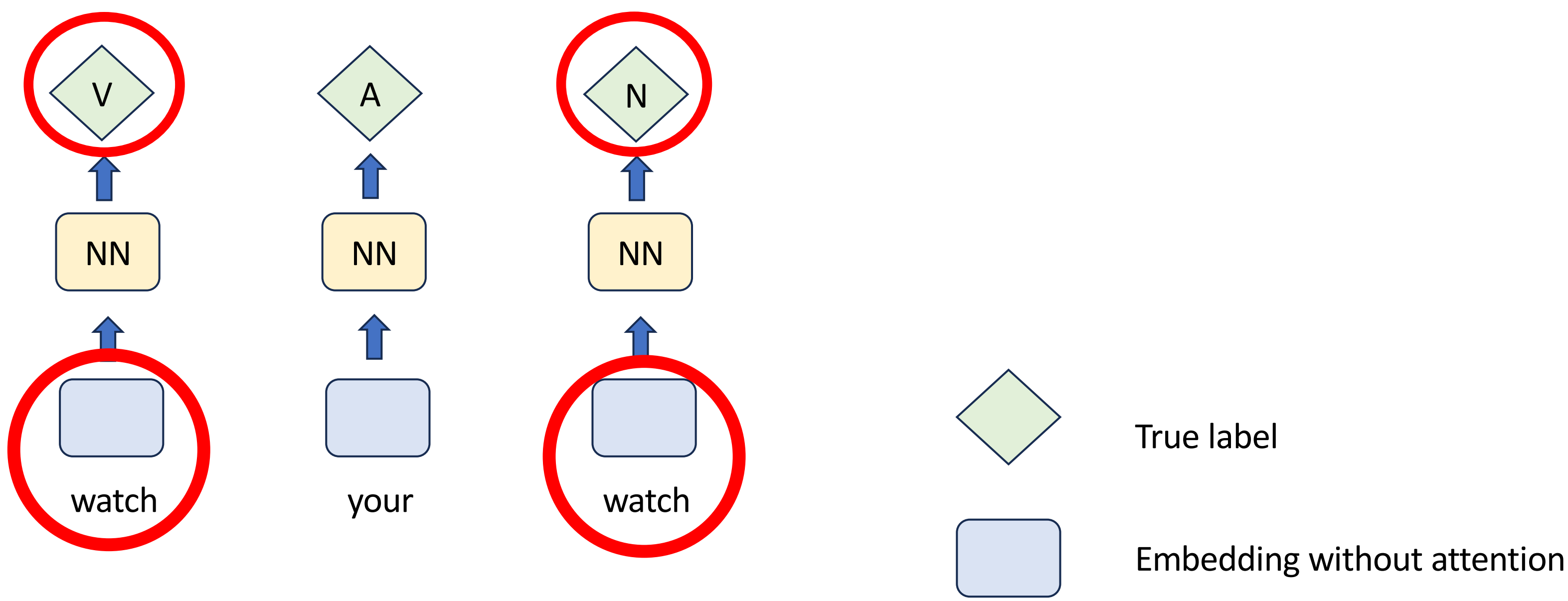
- How to make the inputs of words as vectors of numbers?
- See the blackboard for an illustration (optional)

First task: Parts-Of-Speech (POS) classification

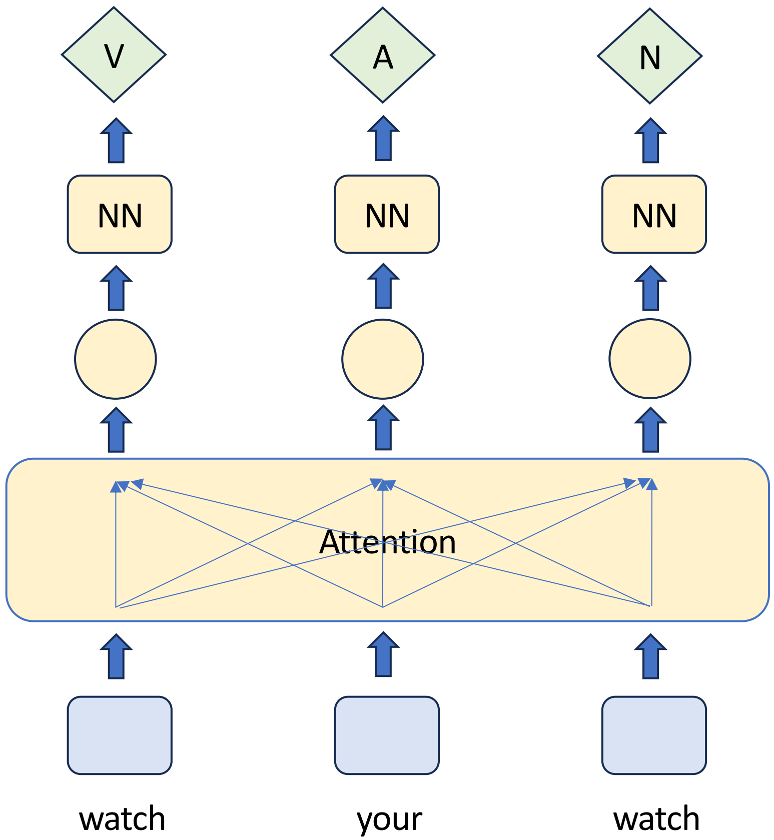
- A POS is a grammatical classification that commonly includes verbs, adjectives, adverbs, nouns, etc.

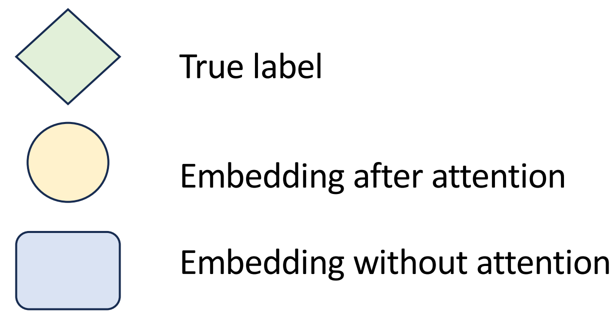


We need to efficiently exploit previous contexts

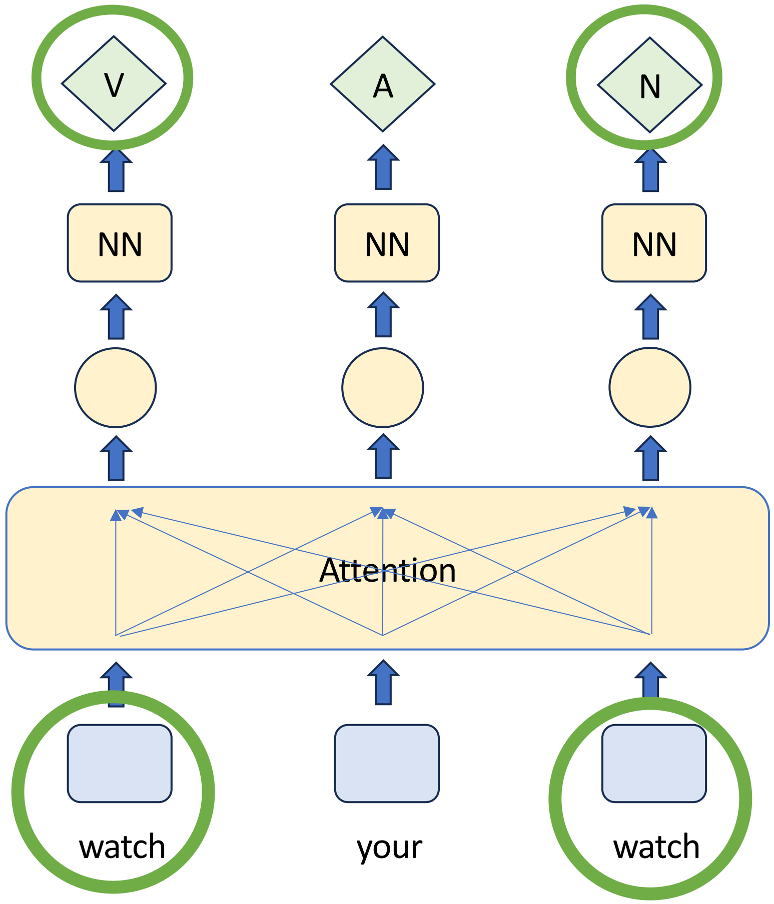


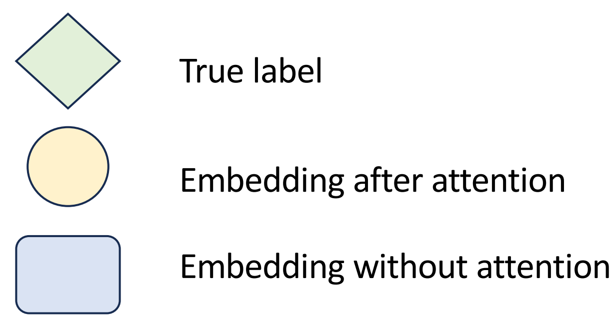
Self-attention can effectively leverage context information



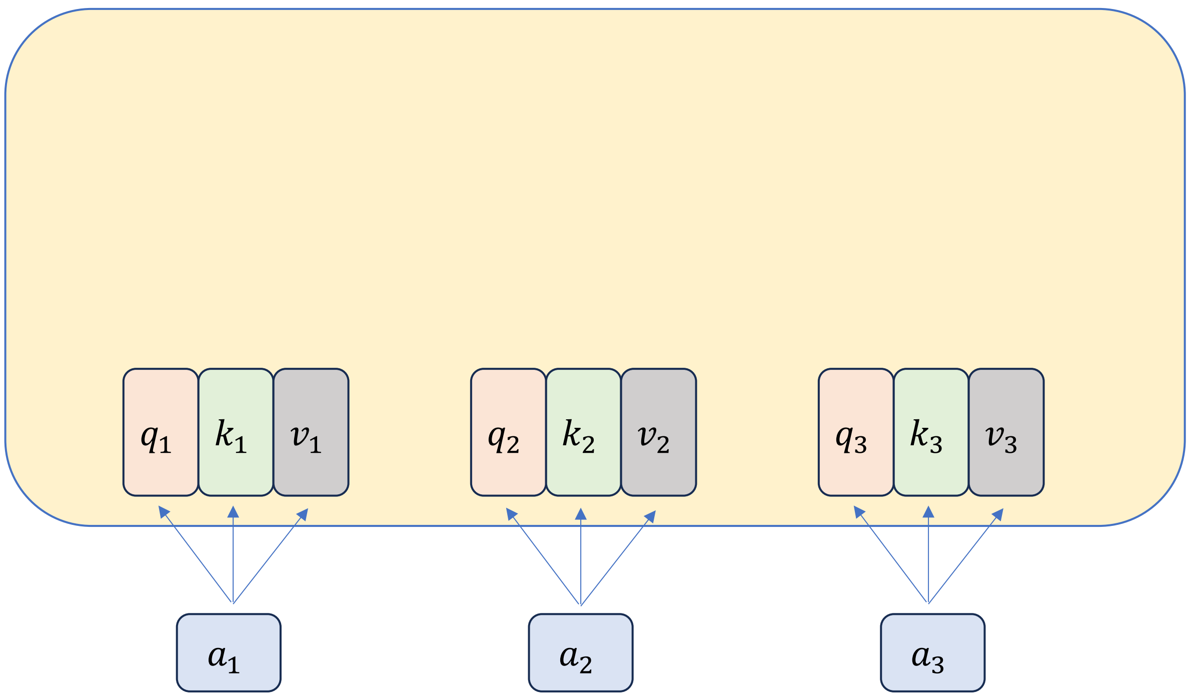


Self-attention can effectively leverage context information





Review of the Self-attention mechanism

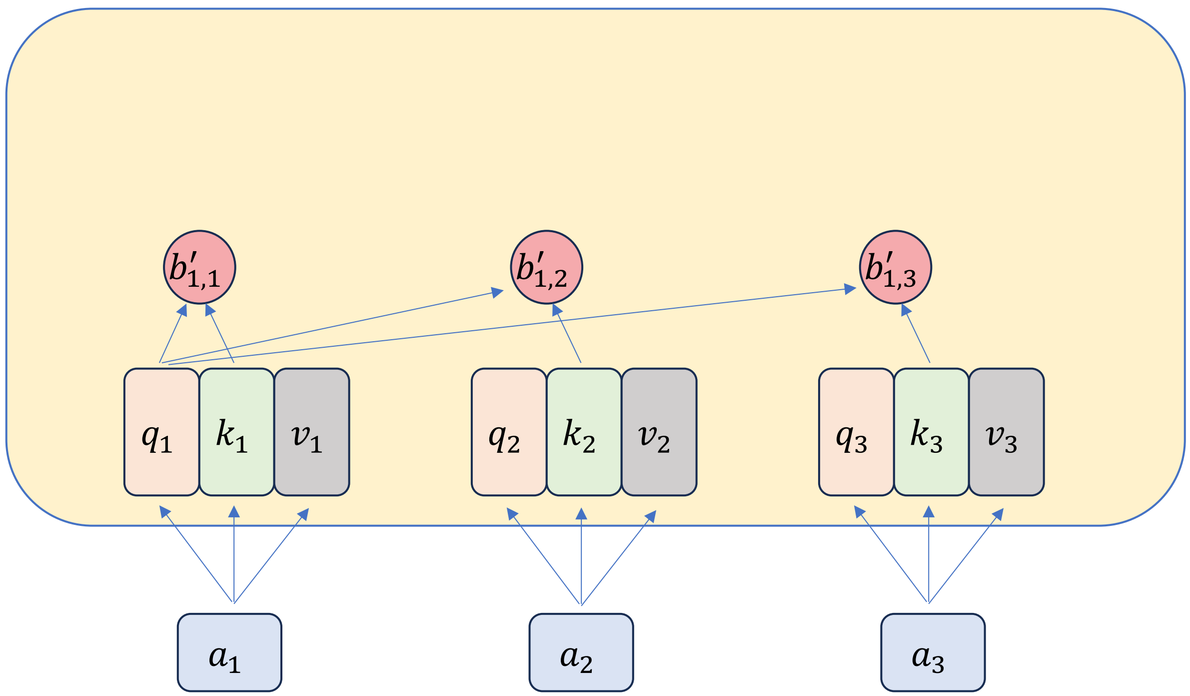


$q_1 = W^{(q)} \cdot a_1 \quad \mathrm{k}_1 = W^{(k)} \cdot a_1 \quad v_1 = W^{(v)} \cdot a_1$

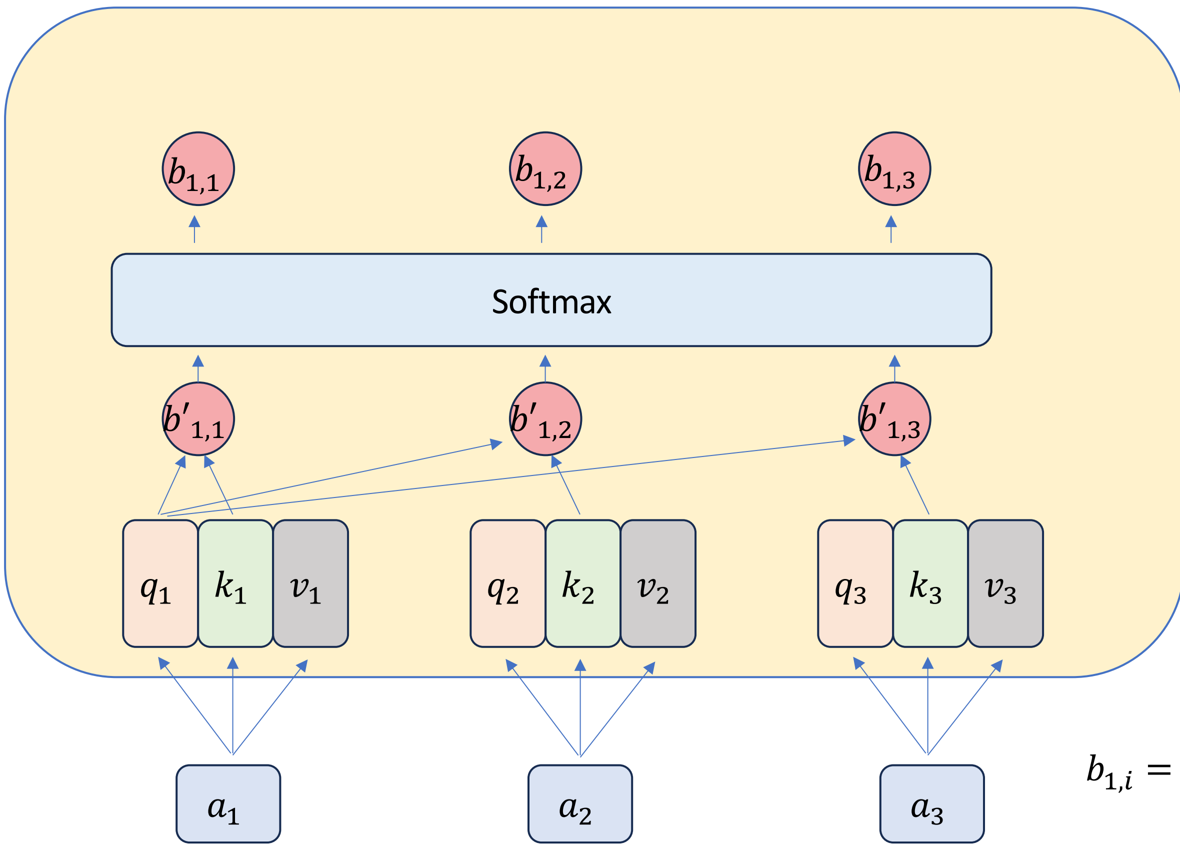
Each token generates a
- Query Q
- Key K
- Value V

Each token will be mapped to their embedding space, i.e. if the input is "How are you", "How" will be mapped to a_1.

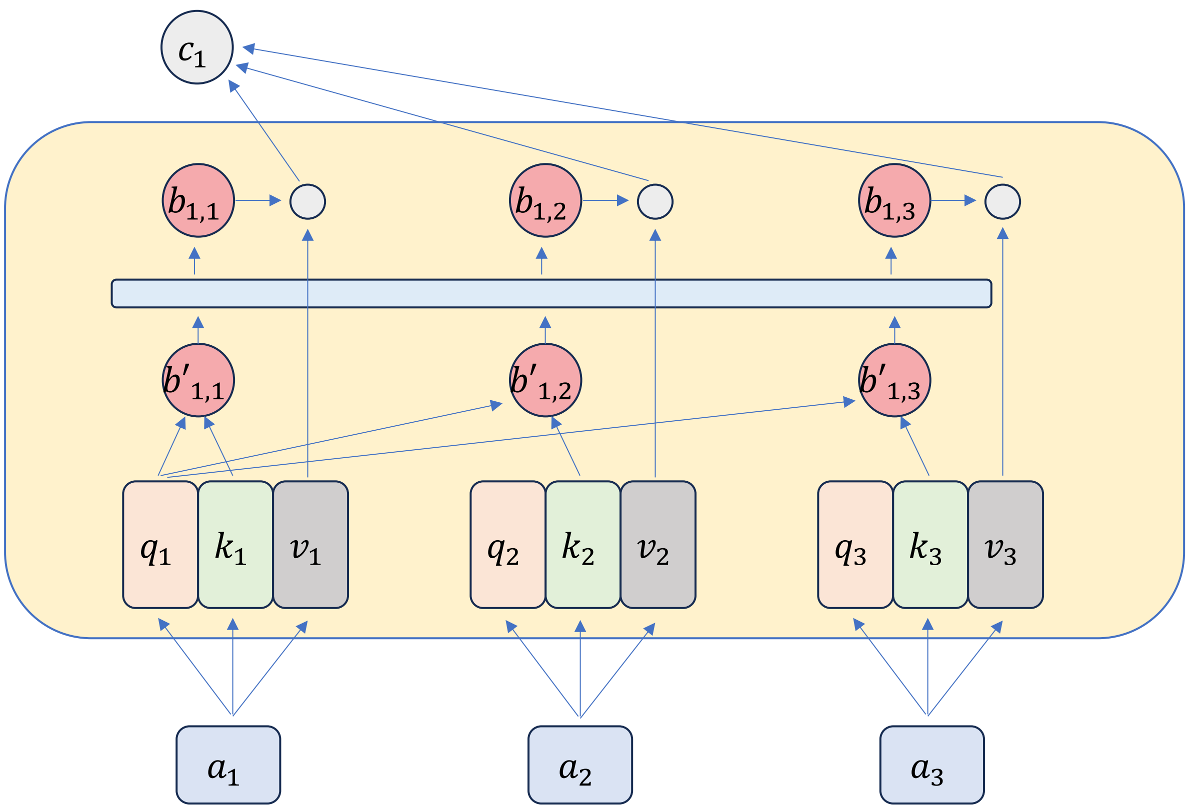
To compute the corresponding query, key, and value, the attention module maintains matrices $W^{(q)}, W^{(k)}, W^{(v)}$ as parameters of the model.



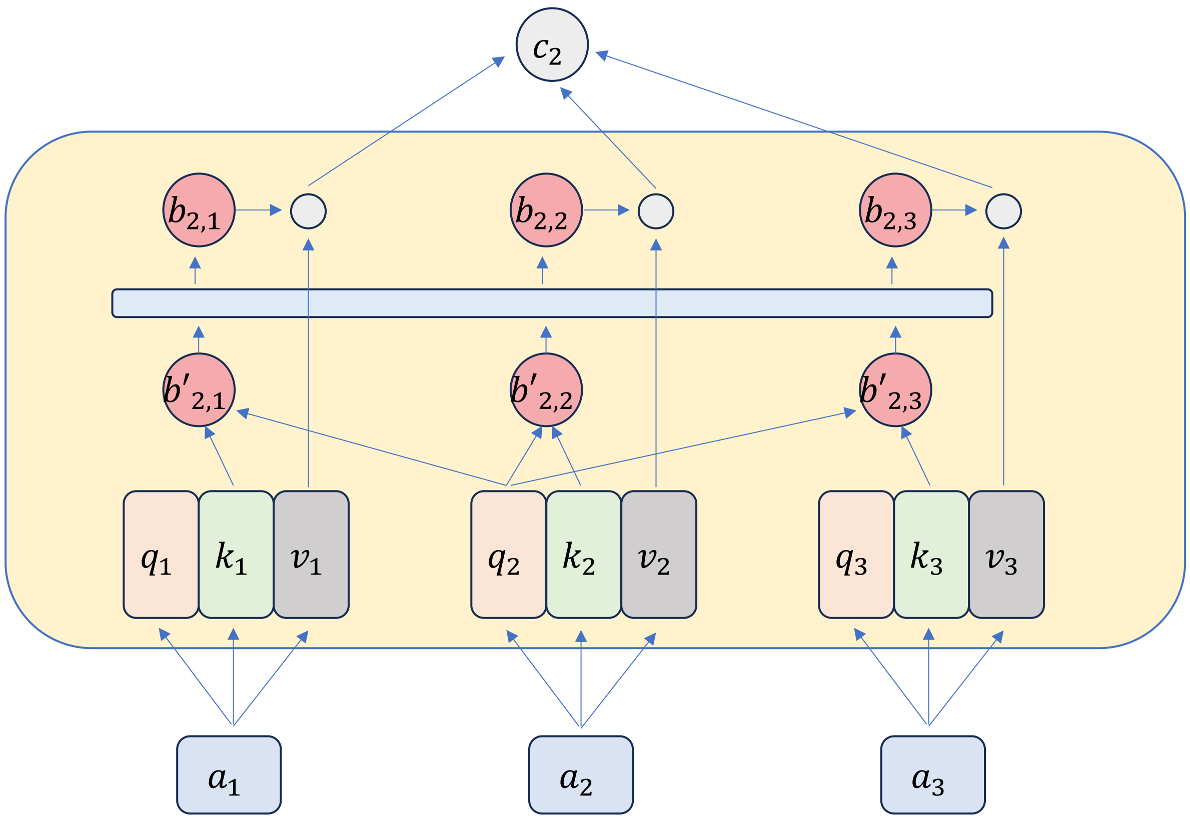
$$b'_{1,i} = k_i \cdot q_1$$



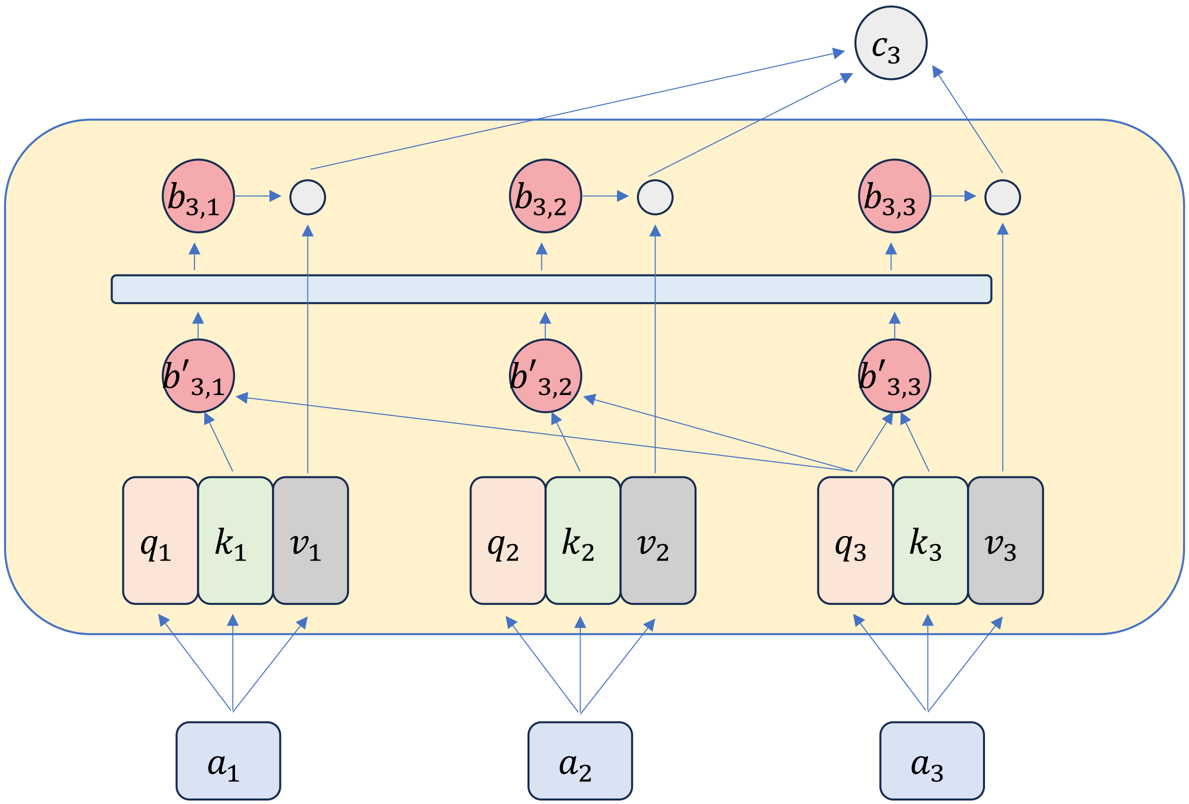
$$b_{1,i} = \exp\left(b'_{1,i}\right) / \sum_j \exp(b'_{1,j})$$



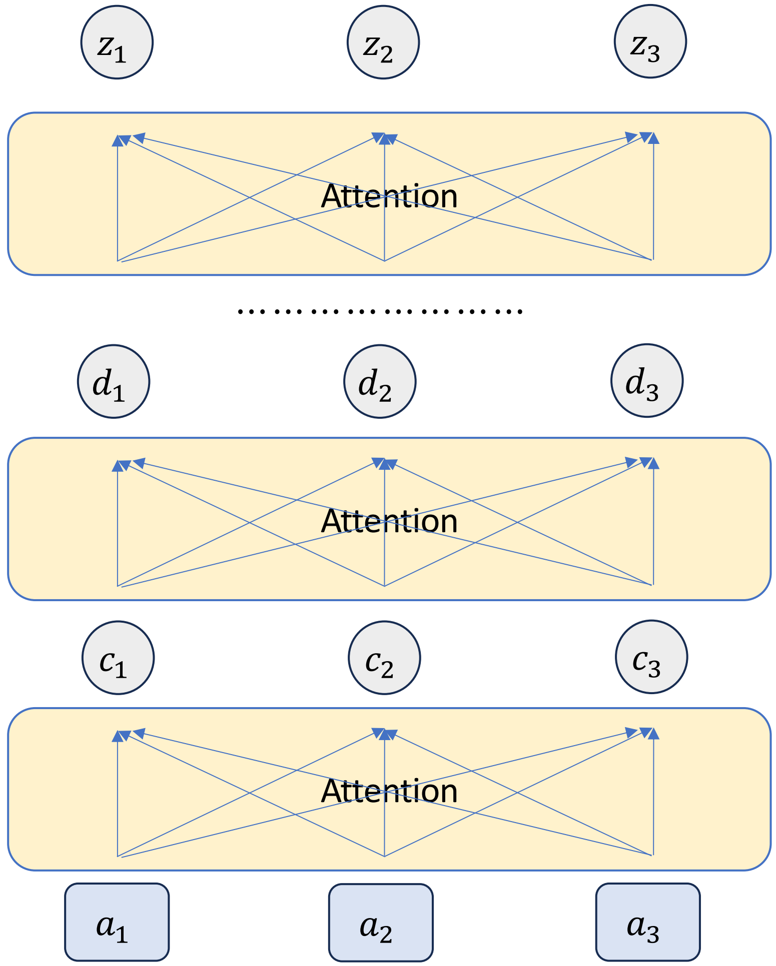
$$c_1 = \sum_i b_{1,i} \cdot v_i$$



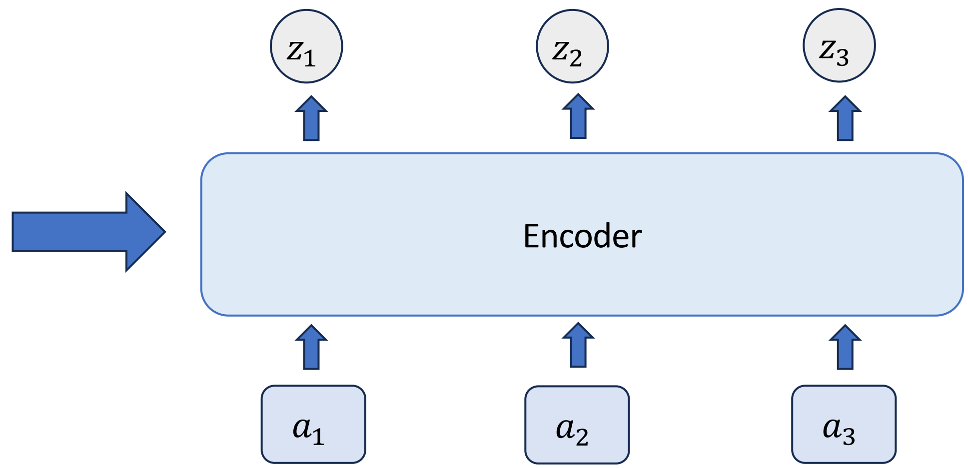
c_2
$b_{2,1}$
$b_{2,2}$
$b_{2,3}$
$b'_{2,1}$
$b'_{2,2}$
$b'_{2,3}$
q_1 k_1 v_1
q_2 k_2 v_2
q_3 k_3 v_3
a_1
a_2
a_3



c_3
$b_{3,1}$
$b_{3,2}$
$b_{3,3}$
$b'_{3,1}$
$b'_{3,2}$
$b'_{3,3}$
q_1 k_1 v_1
q_2 k_2 v_2
q_3 k_3 v_3
a_1
a_2
a_3



Because the dimension of the output is the same as the input, we can stack multiple attention layer together. And we call this the encoder.



We are missing a few notions for the complete architecture

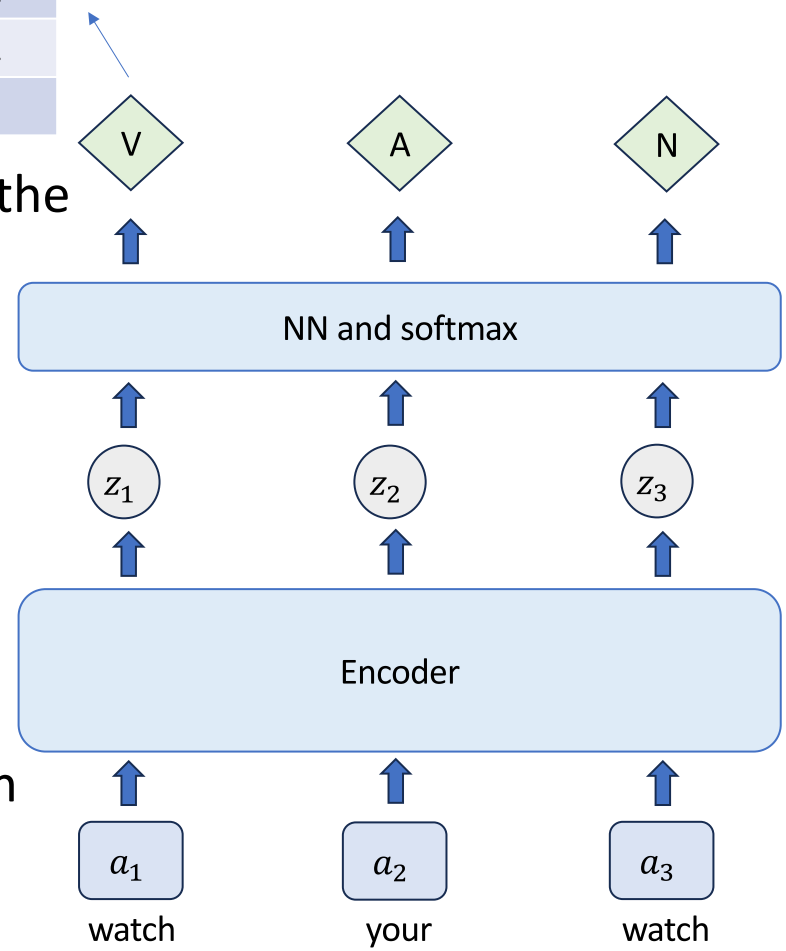
1. The layer norm
2. Multi-head attention
3. Positional encoding

For detail see https://arxiv.org/abs/1706.03762

Softmax operation

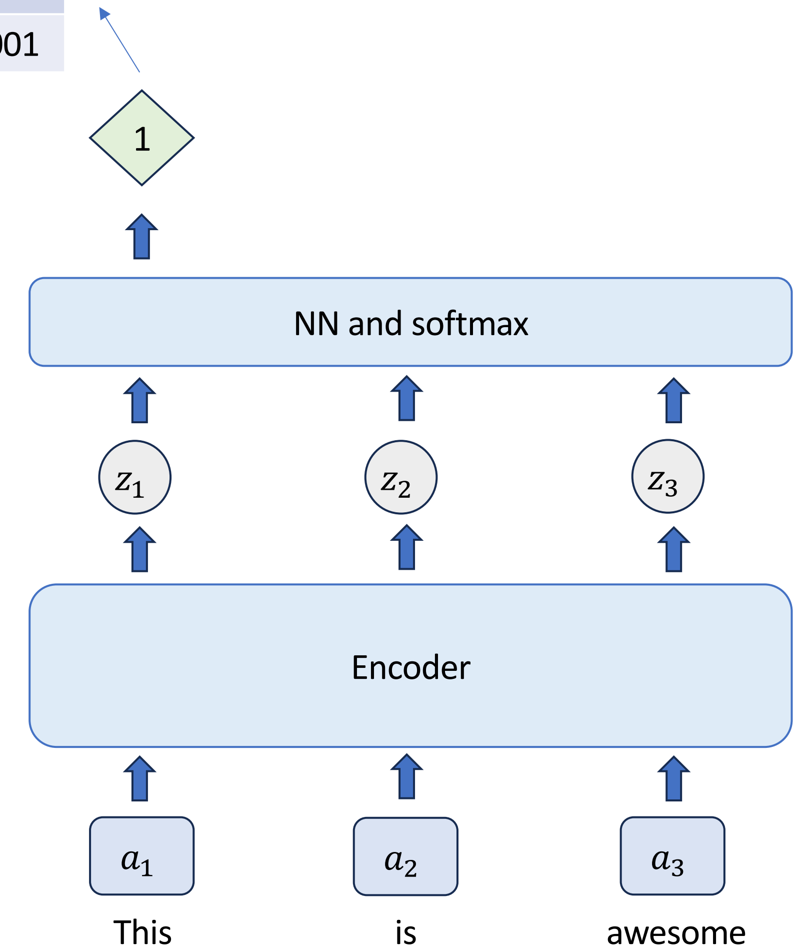
Token	Prob
V	0.92
A	0.02
…	…

- How do we finally turn the outputs of the encoder into the final token?
- We apply a softmax layer so that it outputs the probability of each token and takes the token with highest probability.
- In POS, we have around 10 tokens, but in other NLP tasks such as translation or Q&A, we have to deal with 30000+ tokens.



Softmax operation

Token	Prob
1	0.999
0	0.001

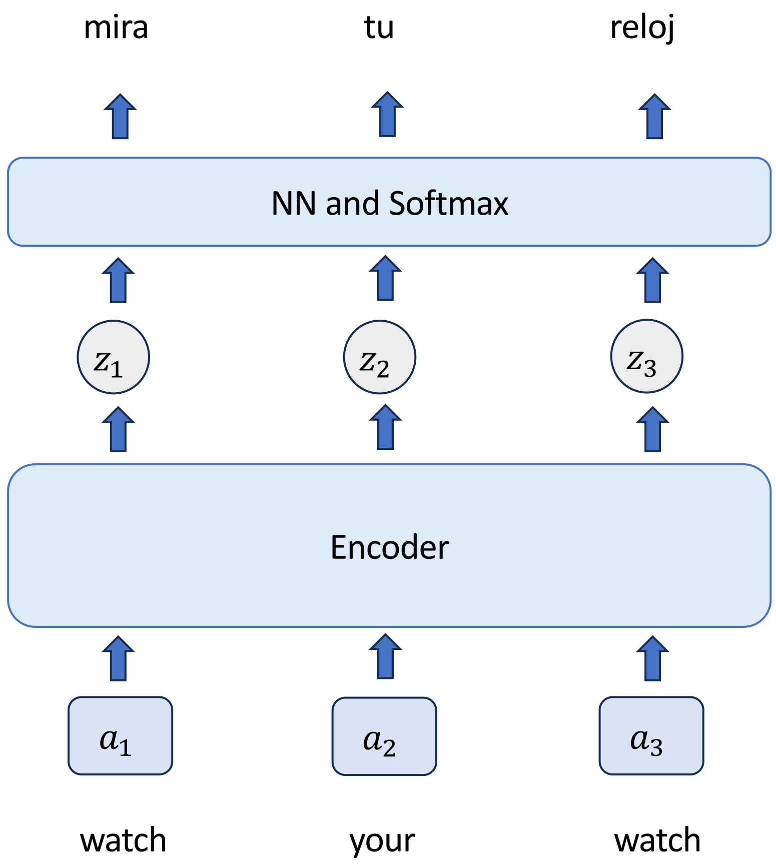


- Another example is sentiment analysis, where the output of the encoder is a binary value, indicating If the sentiment of the sentence.

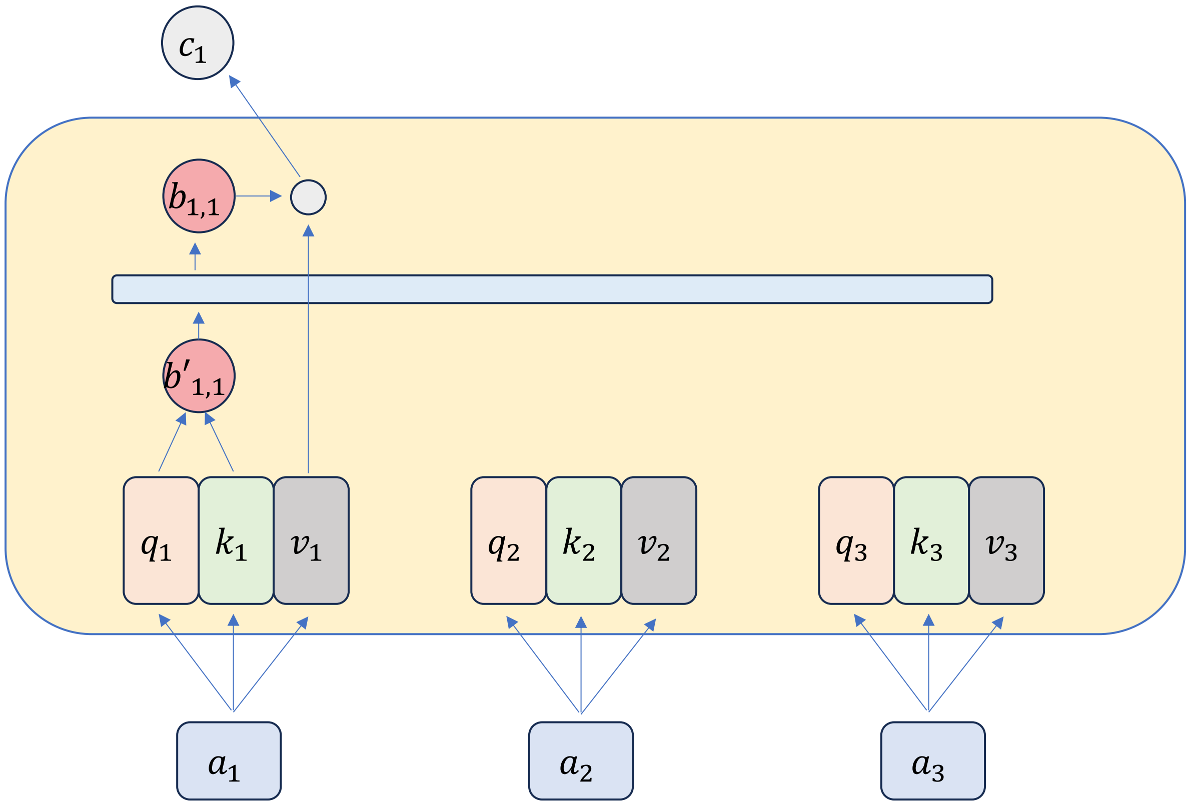
- Here, 1 stands for positive, and 0 stands for negative.

Other tasks

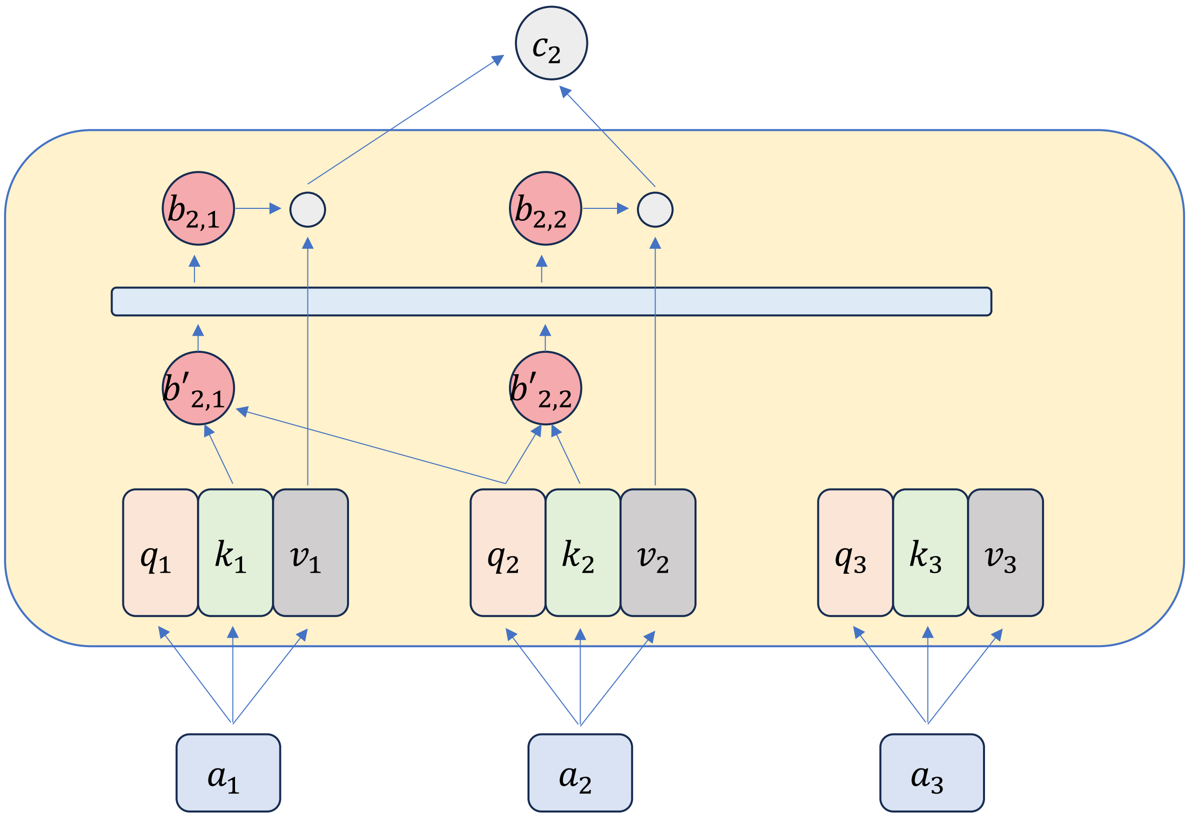
- We have shown how to use encoder for classifications.
- However, for other tasks such as translation or content generation, it is not necessary the case that each token corresponds to a specific value.
- For translation and content generation, the algorithm should know the **previous generated** context and generate tokens in a sequential manner.



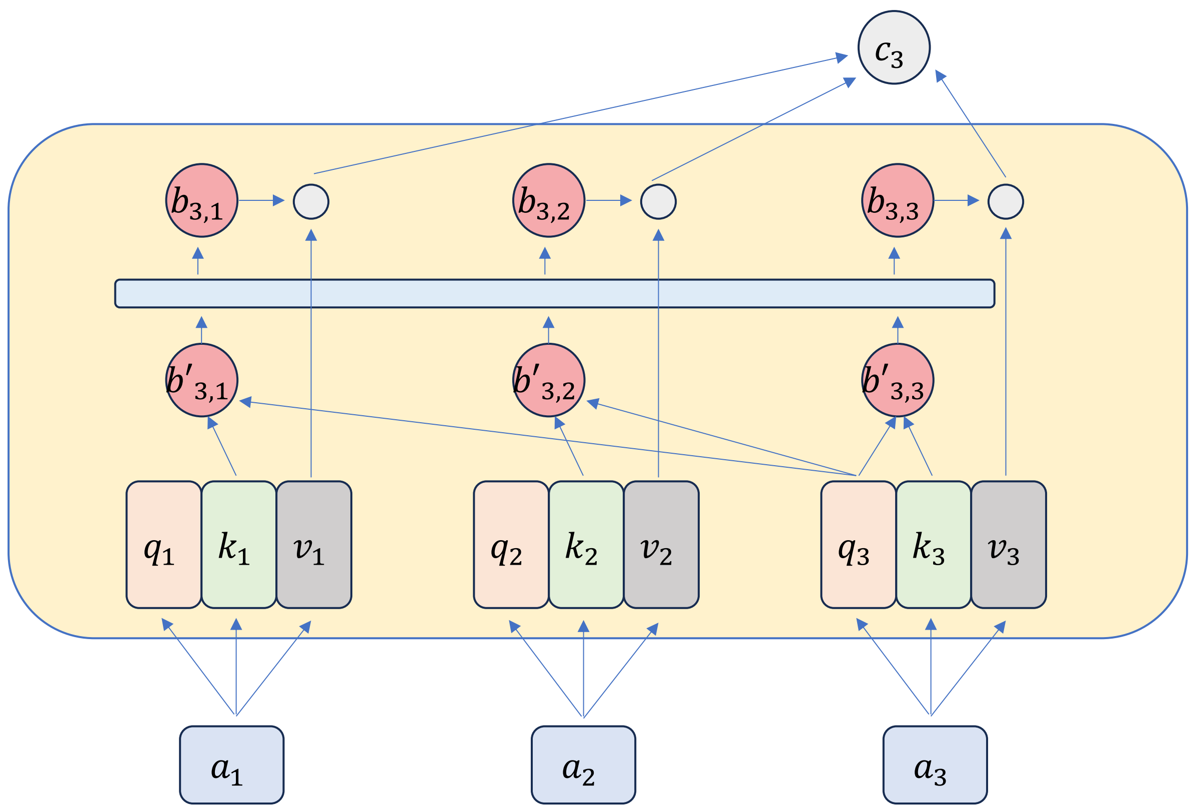
Masked self-attention



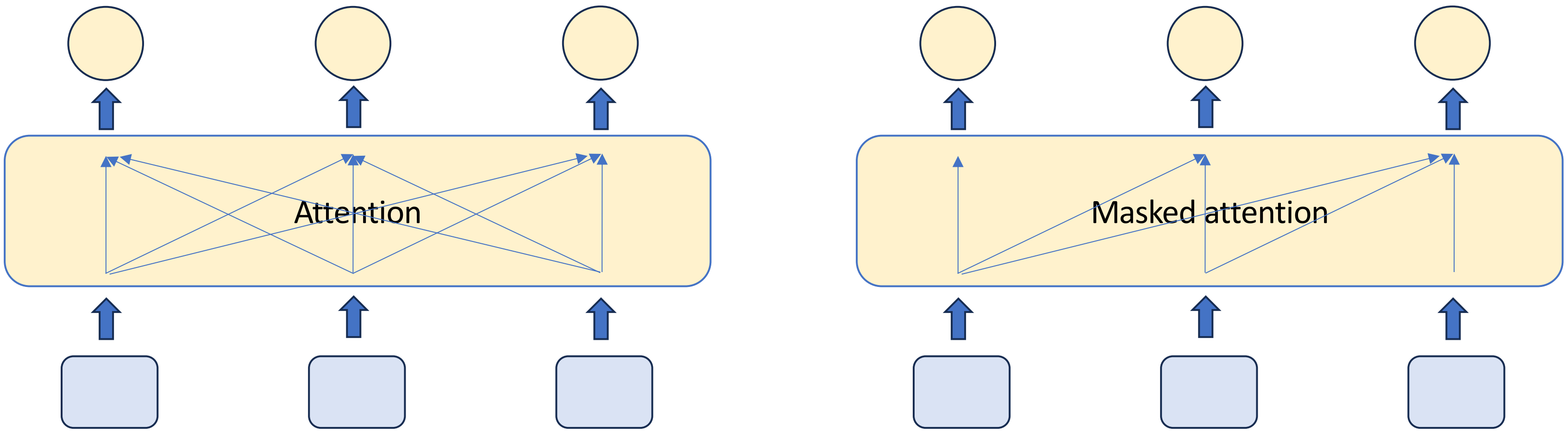
Masked self-attention



Masked self-attention

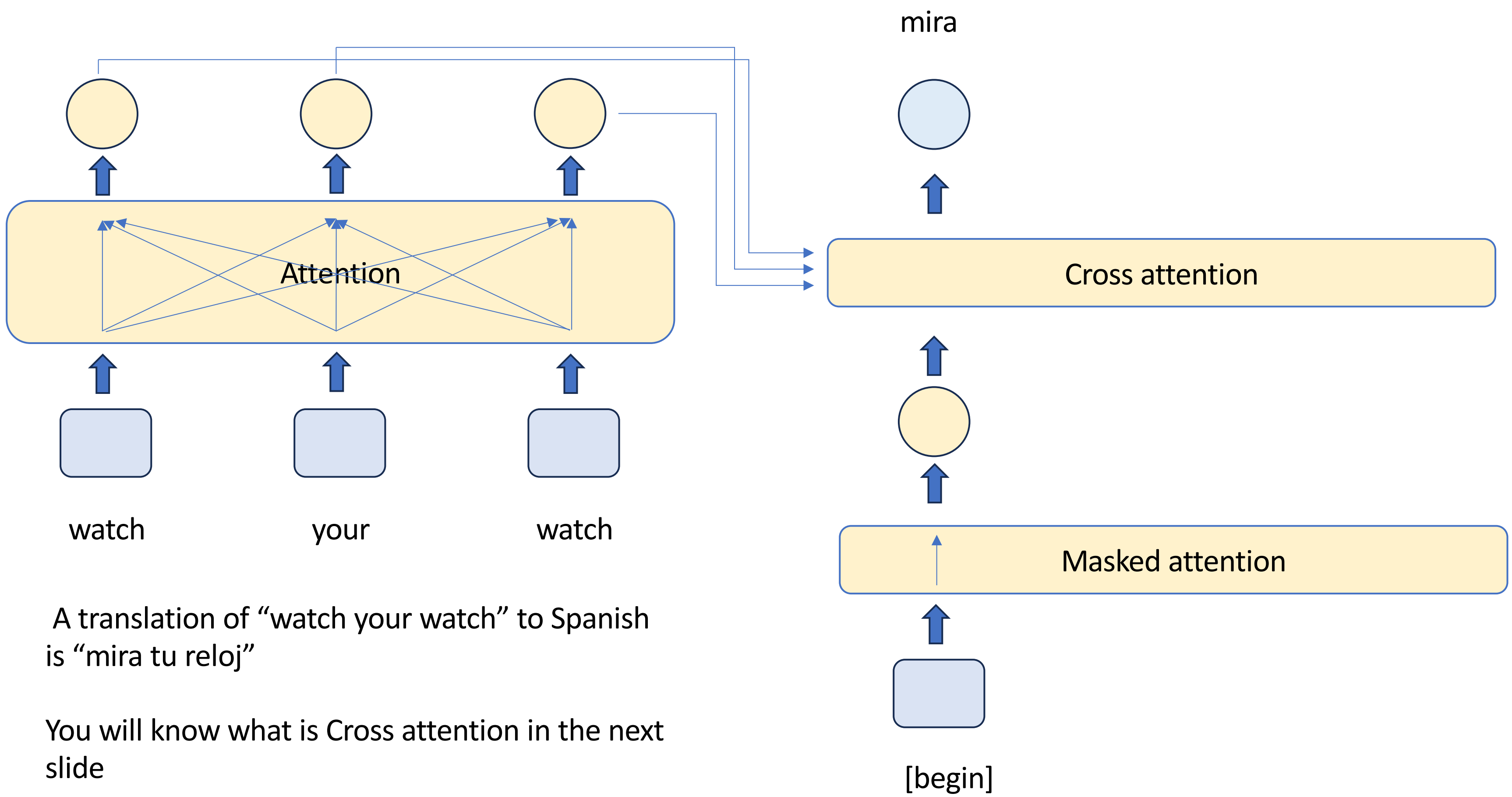


Masked self-attention



The corresponding architecture using masked attention is called the **decoder**.

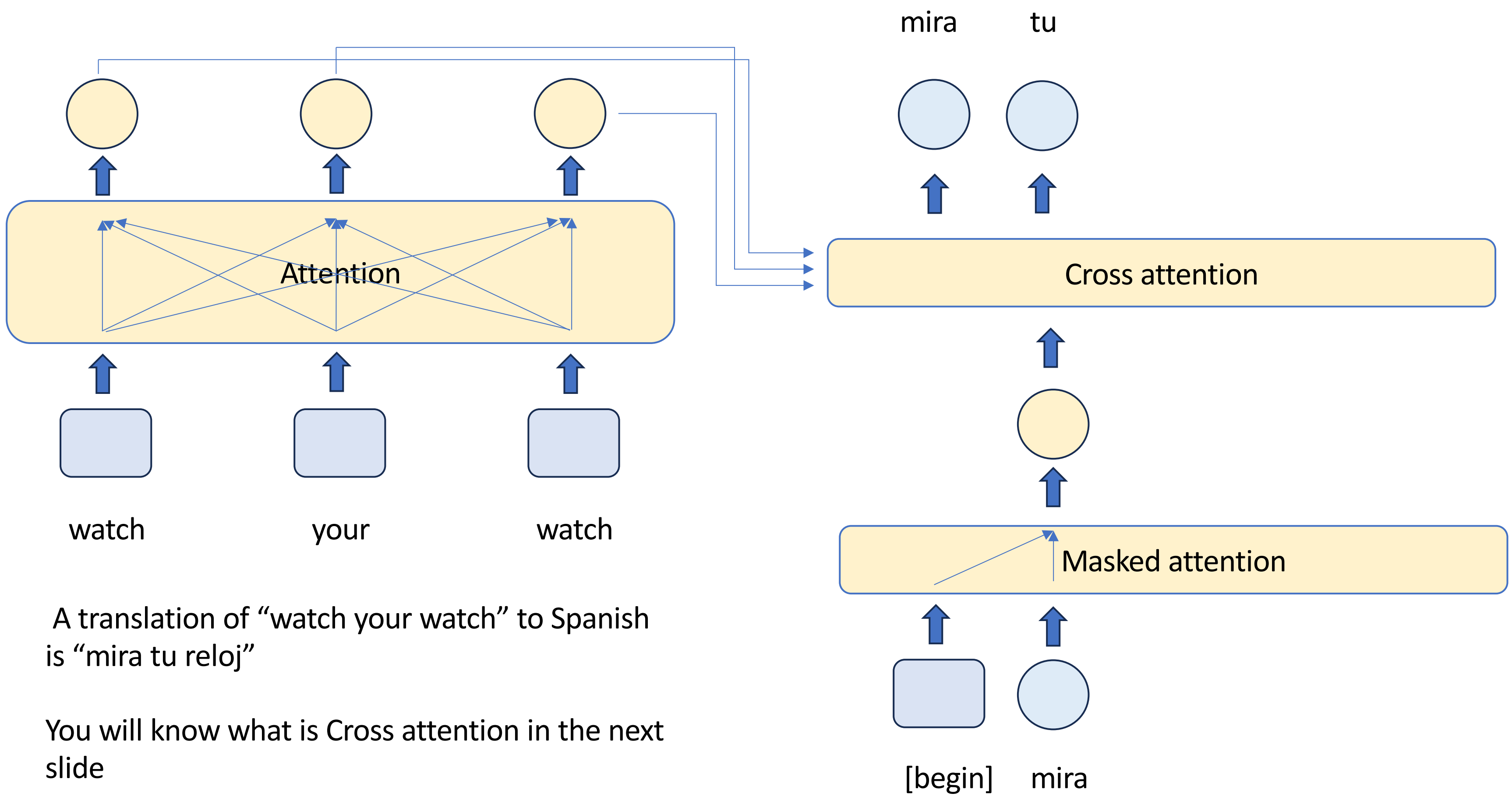
Workflow of encoder-decoder



A translation of “watch your watch” to Spanish is “mira tu reloj”

You will know what is Cross attention in the next slide

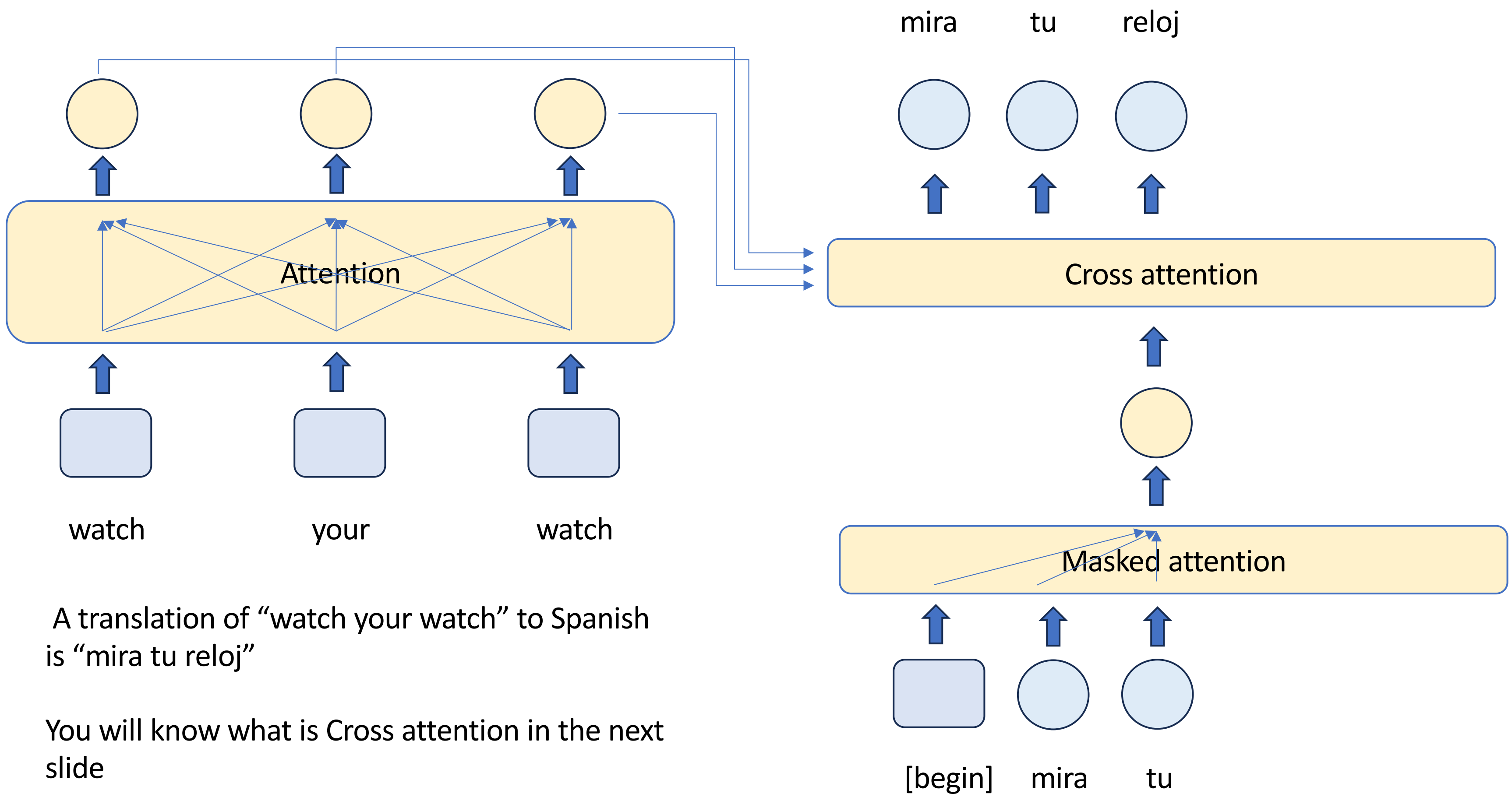
Workflow of encoder-decoder



A translation of “watch your watch” to Spanish is “mira tu reloj”

You will know what is Cross attention in the next slide

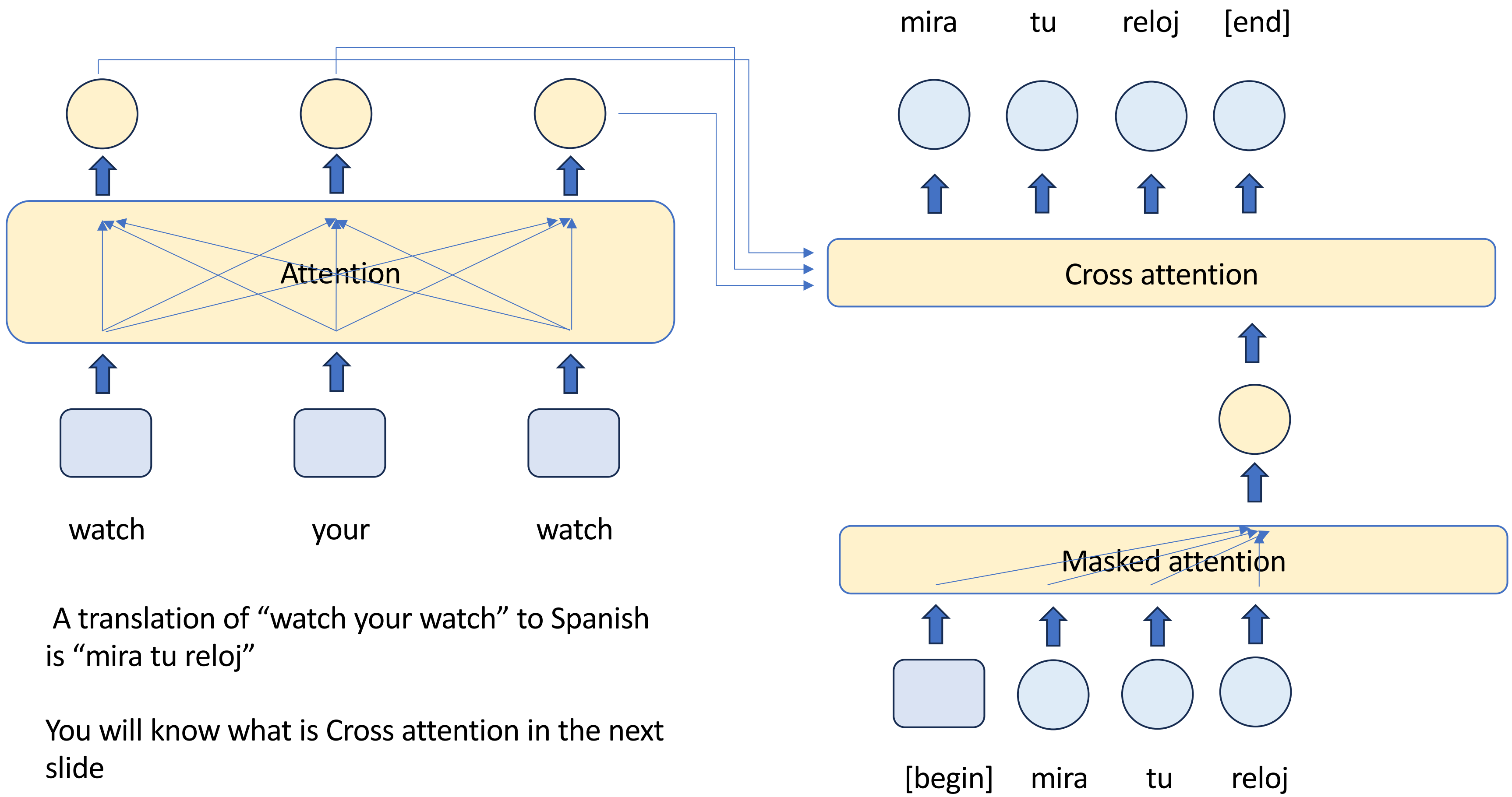
Workflow of encoder-decoder



A translation of “watch your watch” to Spanish is “mira tu reloj”

You will know what is Cross attention in the next slide

Workflow of encoder-decoder



A translation of “watch your watch” to Spanish is “mira tu reloj”

You will know what is Cross attention in the next slide

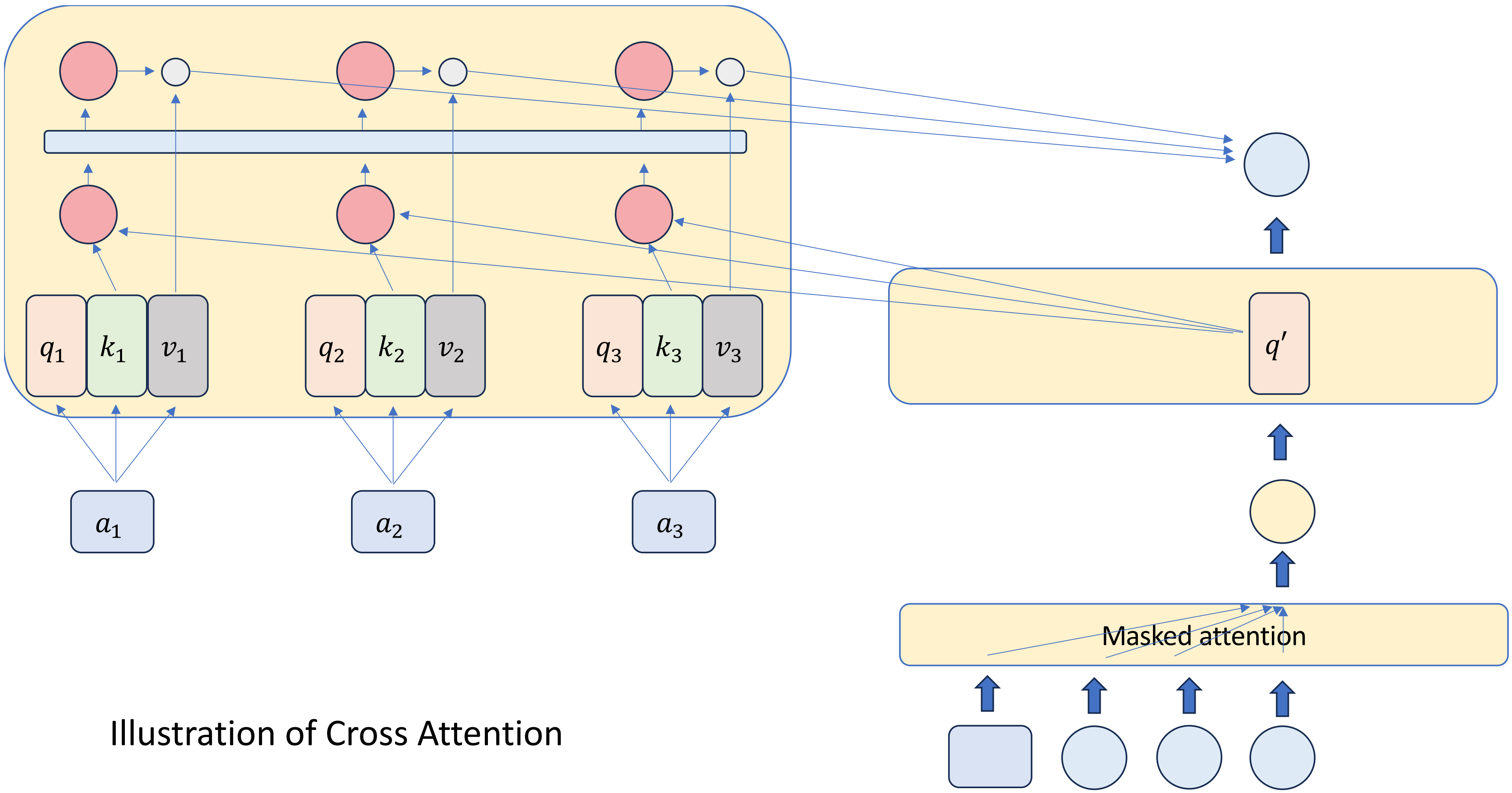


Illustration of Cross Attention

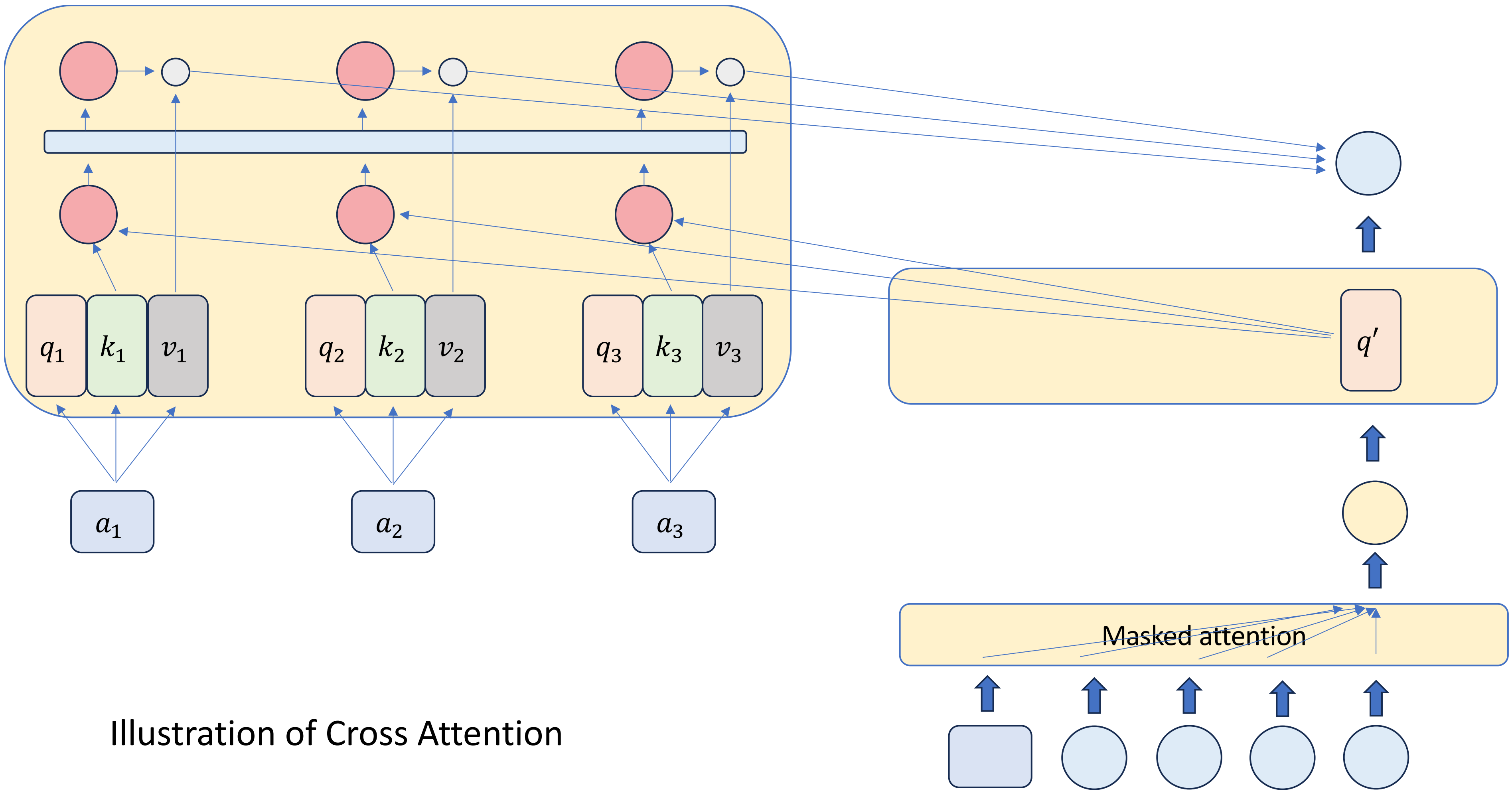


Illustration of Cross Attention

The Transformer Architecture

This is from the original transformer paper: Attention is All You Need

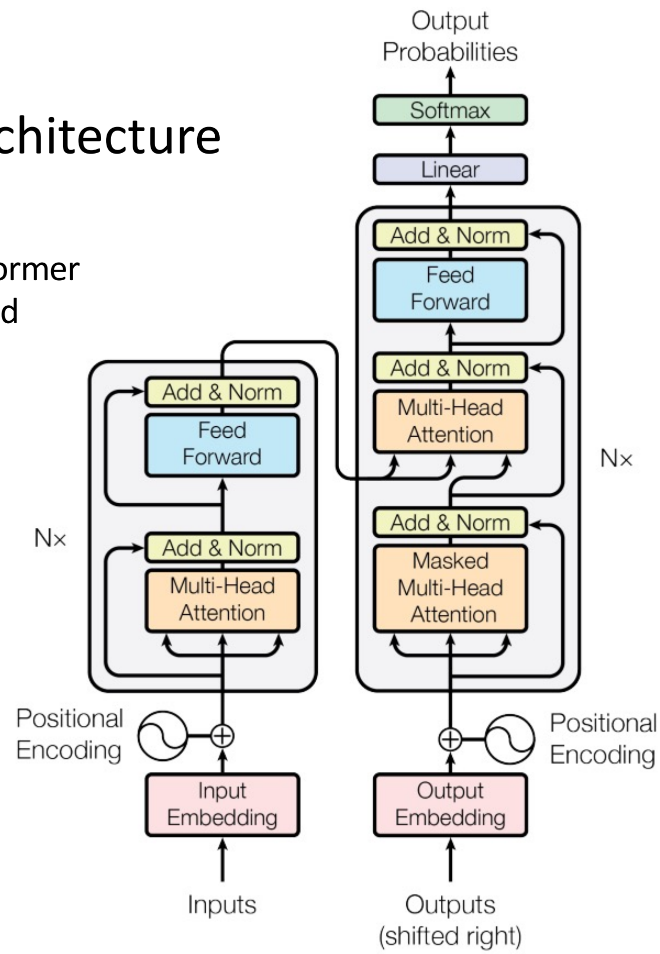
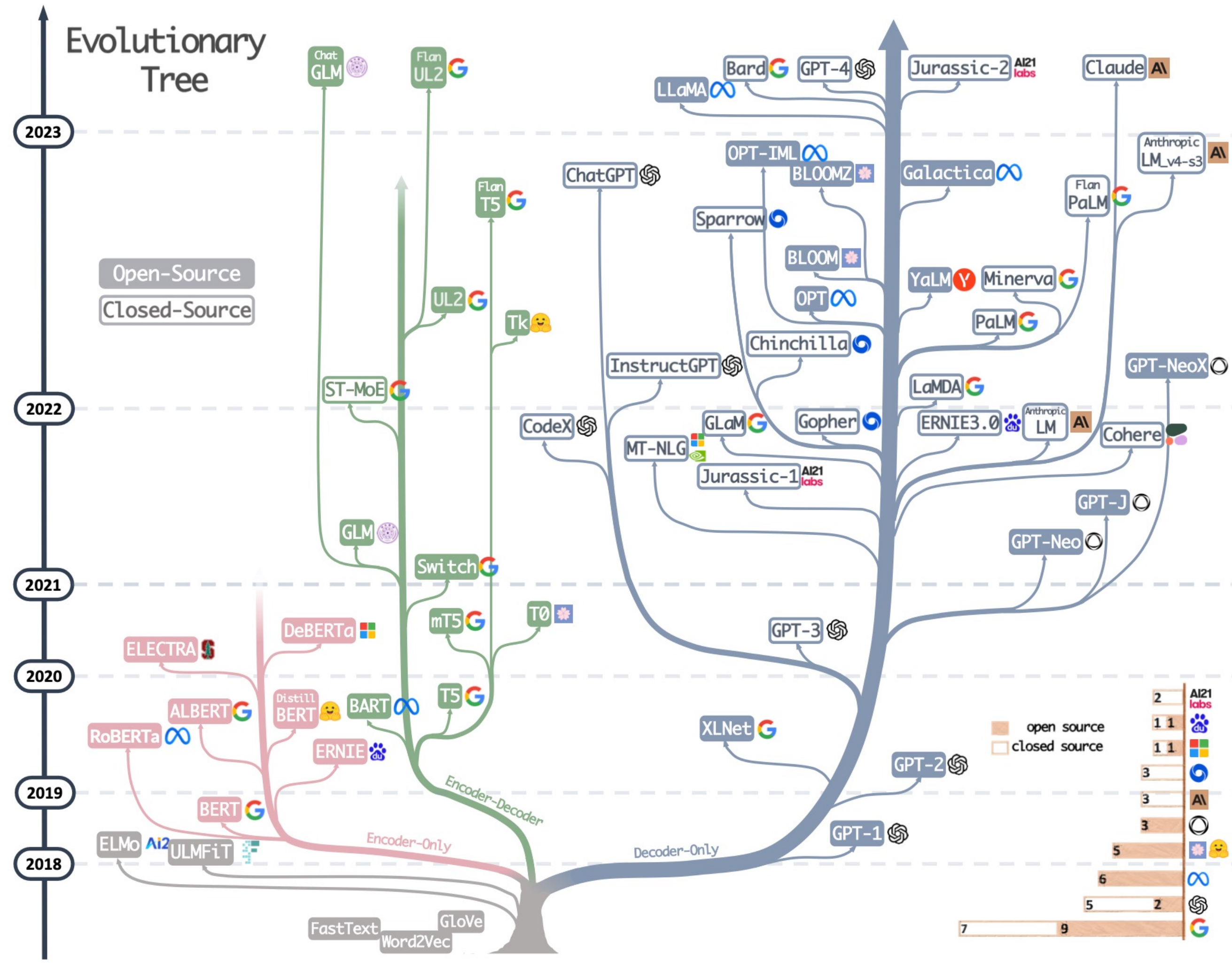


Figure 1: The Transformer - model architecture.

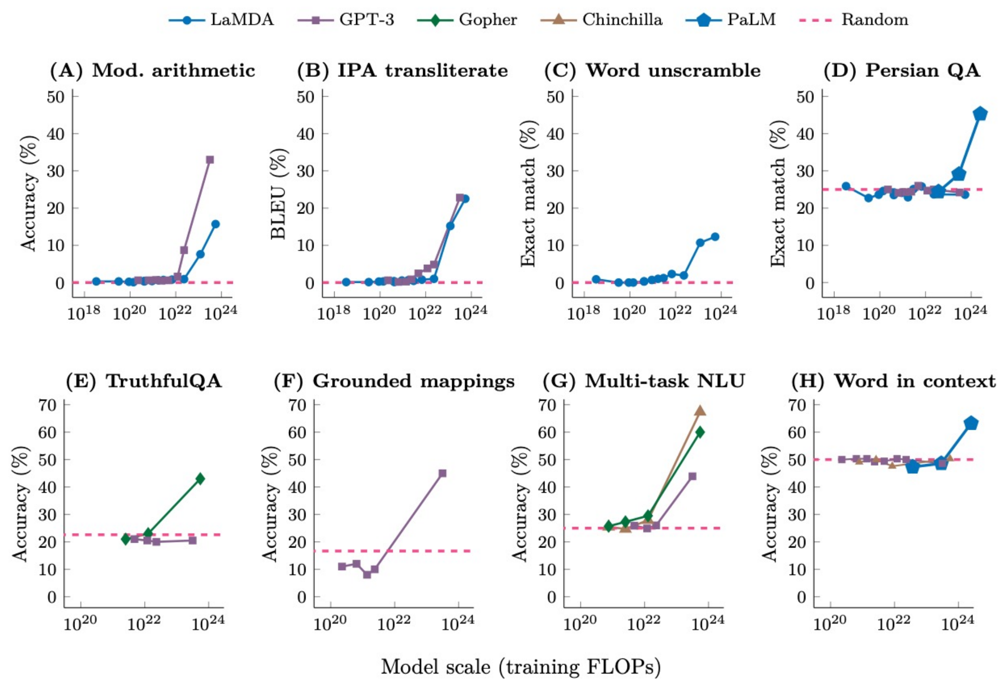
Language Models

Source:
https://arxiv.org/pdf/2304.13712



Language Models

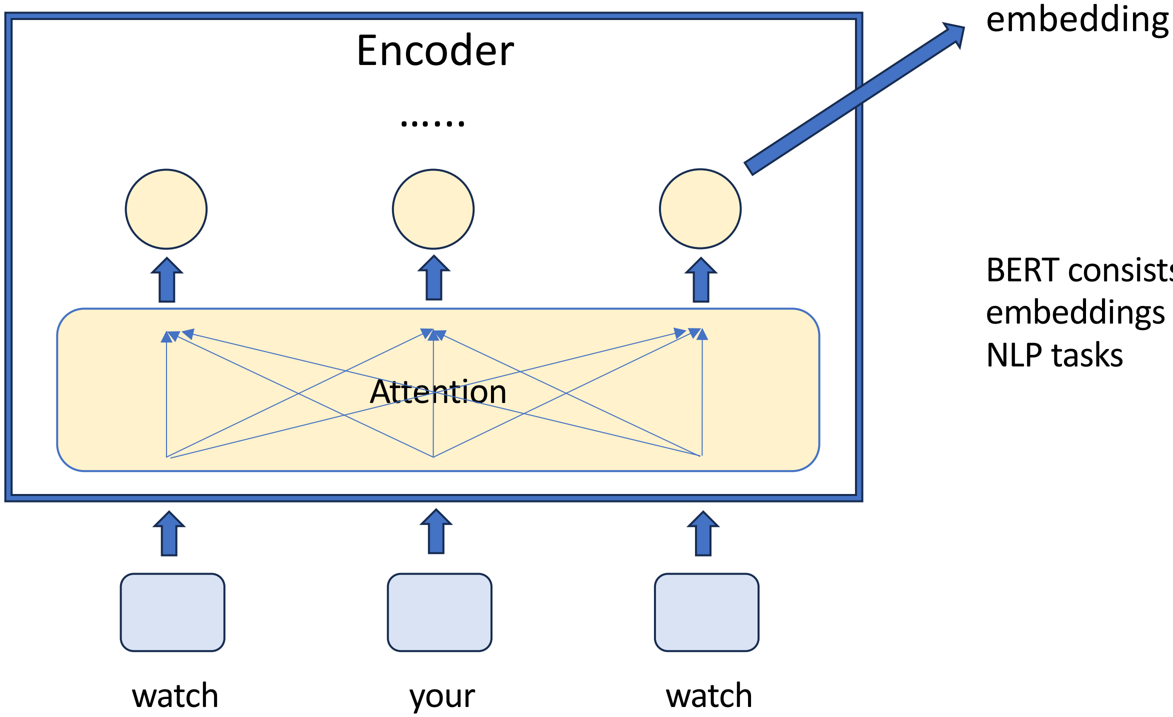
- Why Large Language Models?



Language Models

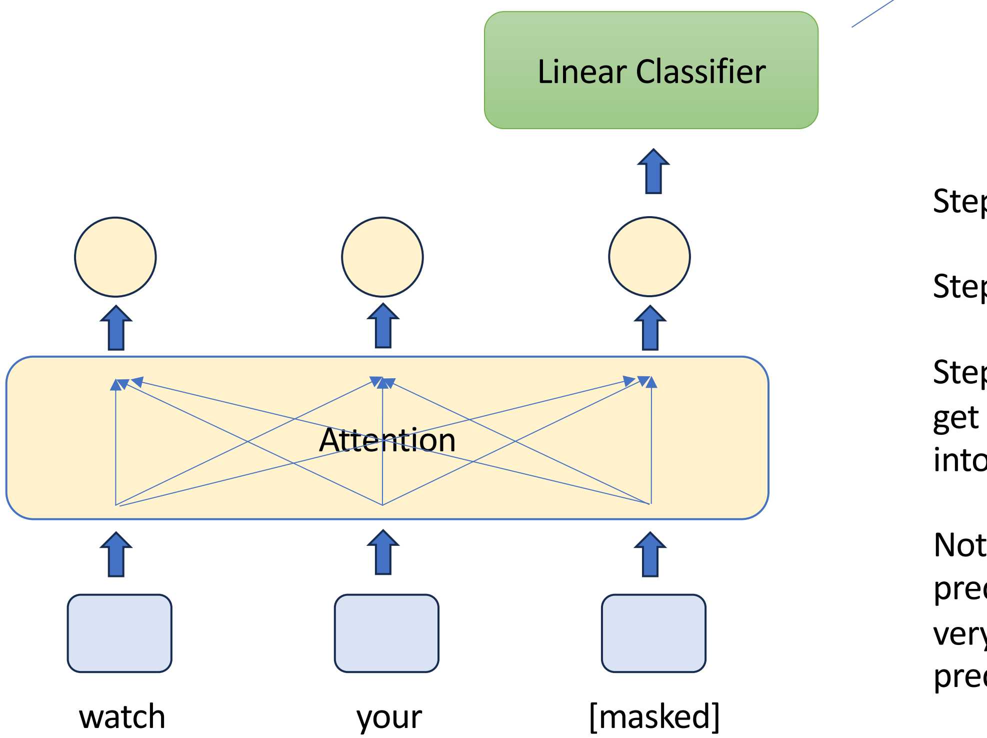
- We will introduce two types of language models
- Encoder-only: BERT (Bidirectional Encoder Representations from Transformers)
- Decoder-only: GPT (Generative pre-training)

Review: BERT



BERT consists of encoders, leveraging the embeddings after attention layers to perform NLP tasks

Training BERT



Token	Prob
Step	0.32
Watch	0.21
…	…

Step 1: Take a large corpus of written text.

Step 2: Mask each token with probability 15%

Step 3: Guess which token the masked one is, get the corresponding embedding and put it into a linear classifier.

Note that linear classifier is not a powerful prediction model, so it requires BERT to train a very informative embedding to make the right prediction.

Note there is another way to training BERT, but we will skip

Using BERT for Q&A

Input:

Paragraph: Mary is 9 years old. She lives in Palo Alto, CA, and she has a friend named Lori, who lives in Sunnyvale. They like playing music instrument together.

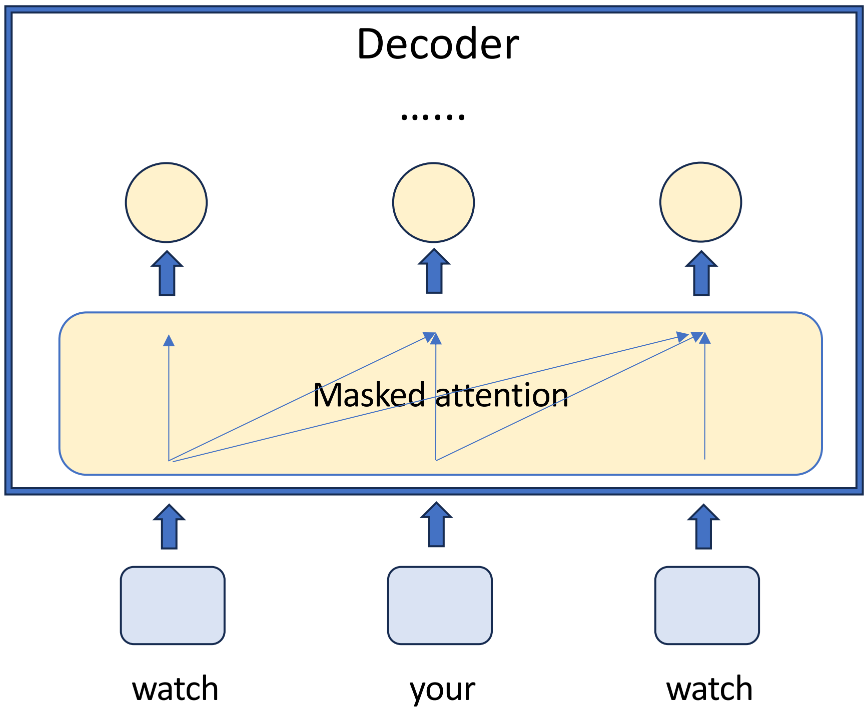
Question: Who is Mary’s friend?

See illustrations in the white board about how BERT produces the answer.

My comments on the limitations of BERT

1. Length limit
2. Flexibility

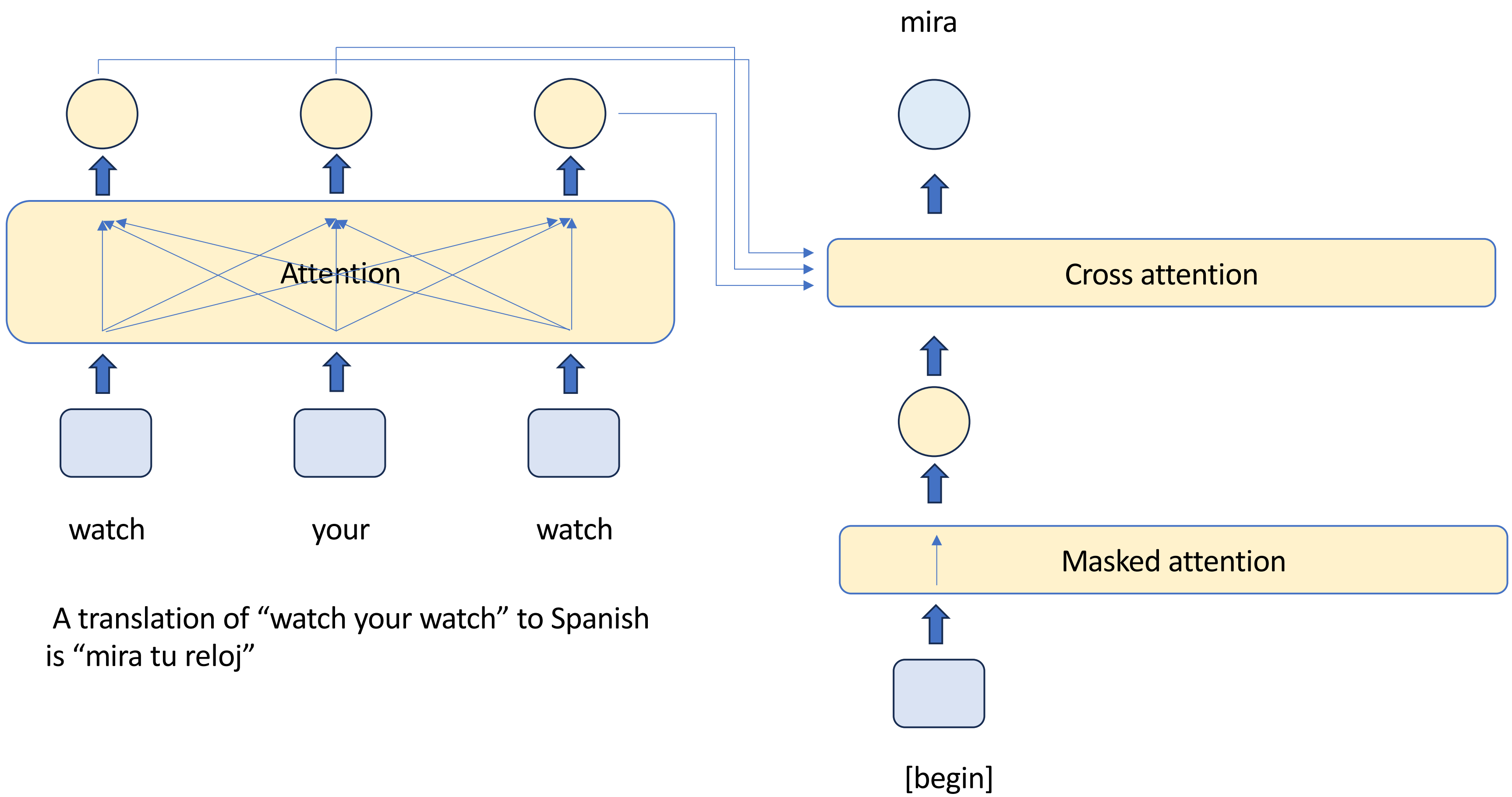
GPT (Generative Pre-Training)



GPT consists of decoders only.

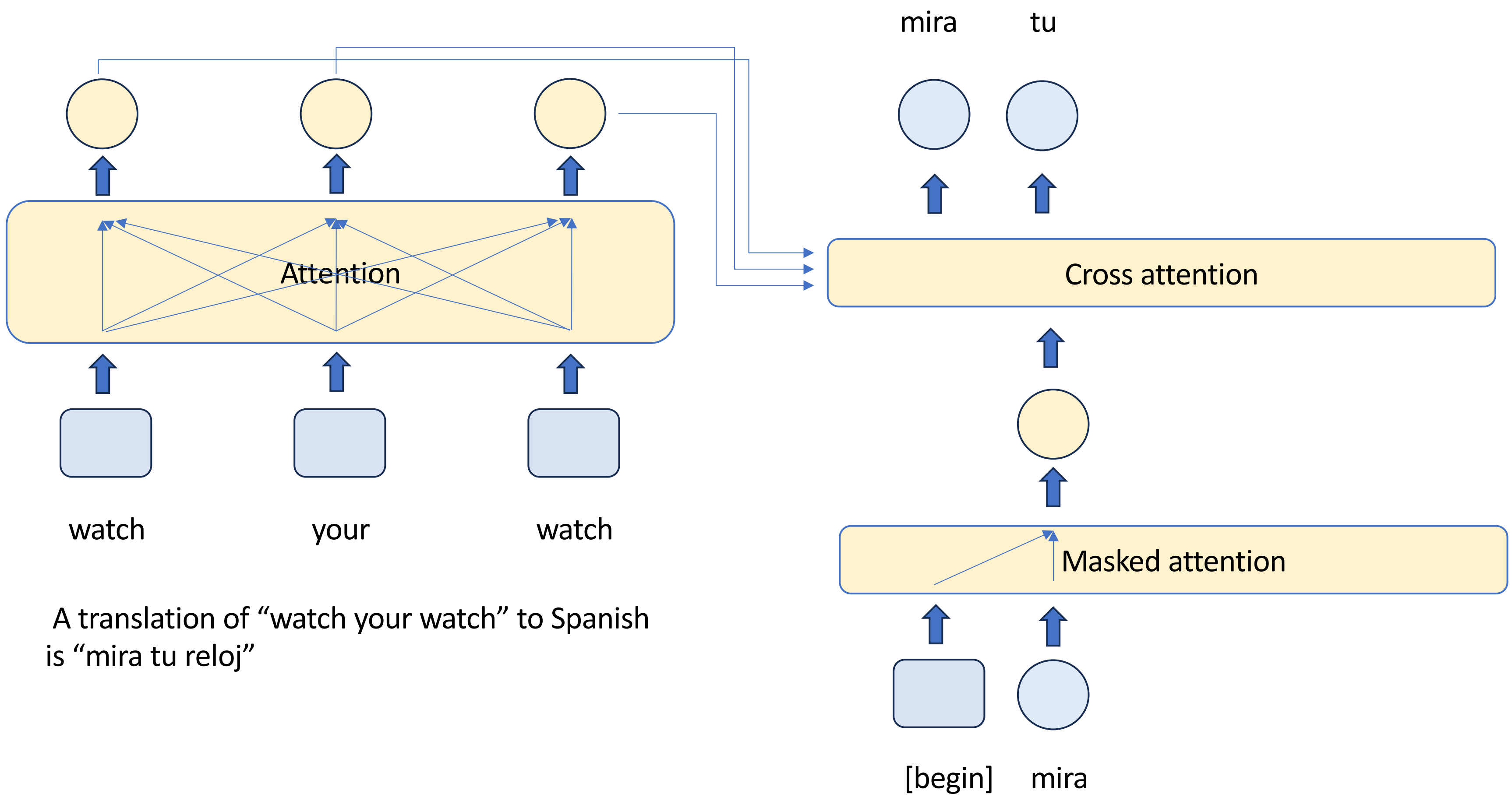
The architecture allows GPT to better perform generative tasks!

Review: workflow of the encoder-decoder structure



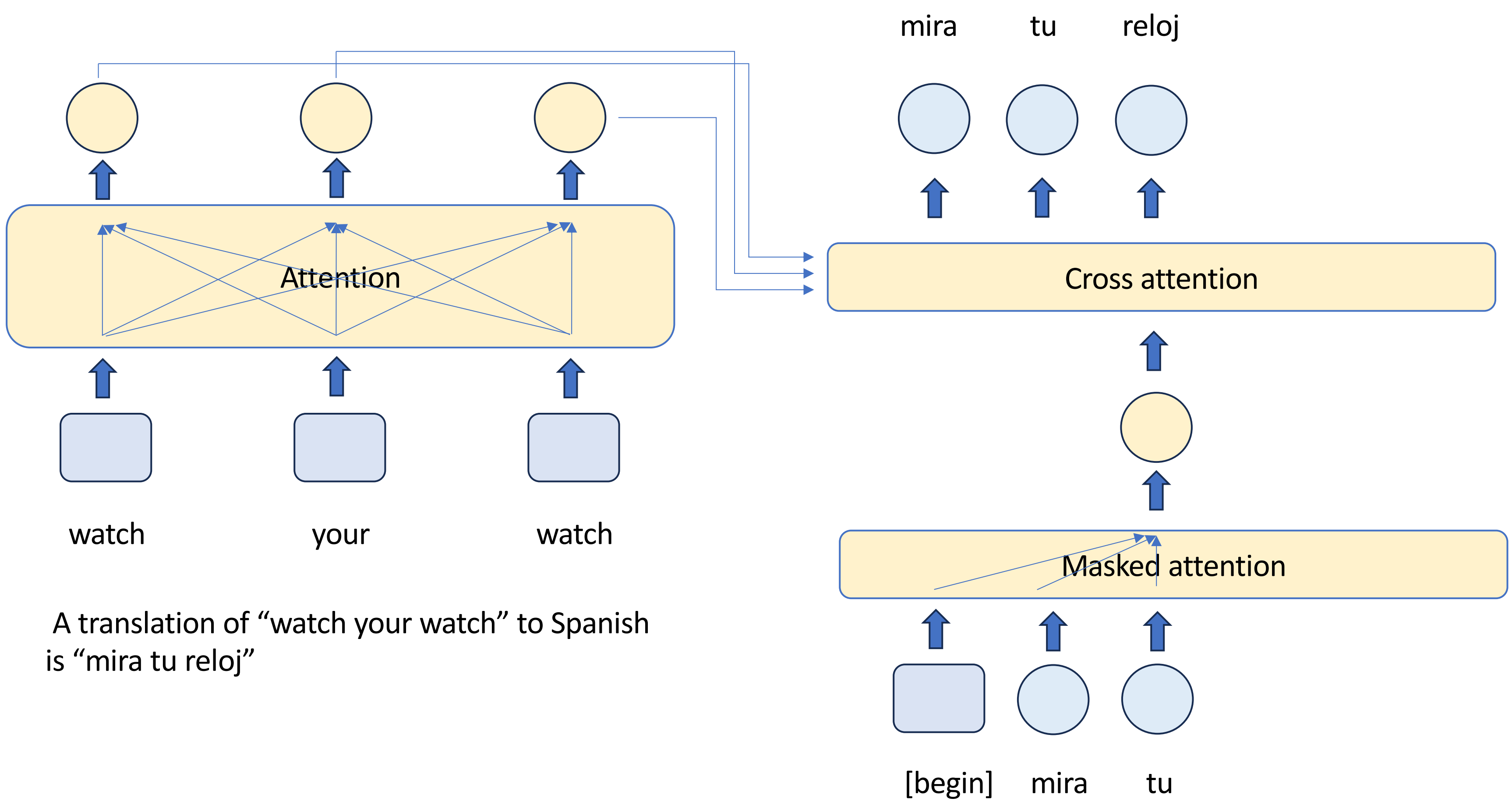
A translation of “watch your watch” to Spanish is “mira tu reloj”

Review: workflow of the encoder-decoder structure



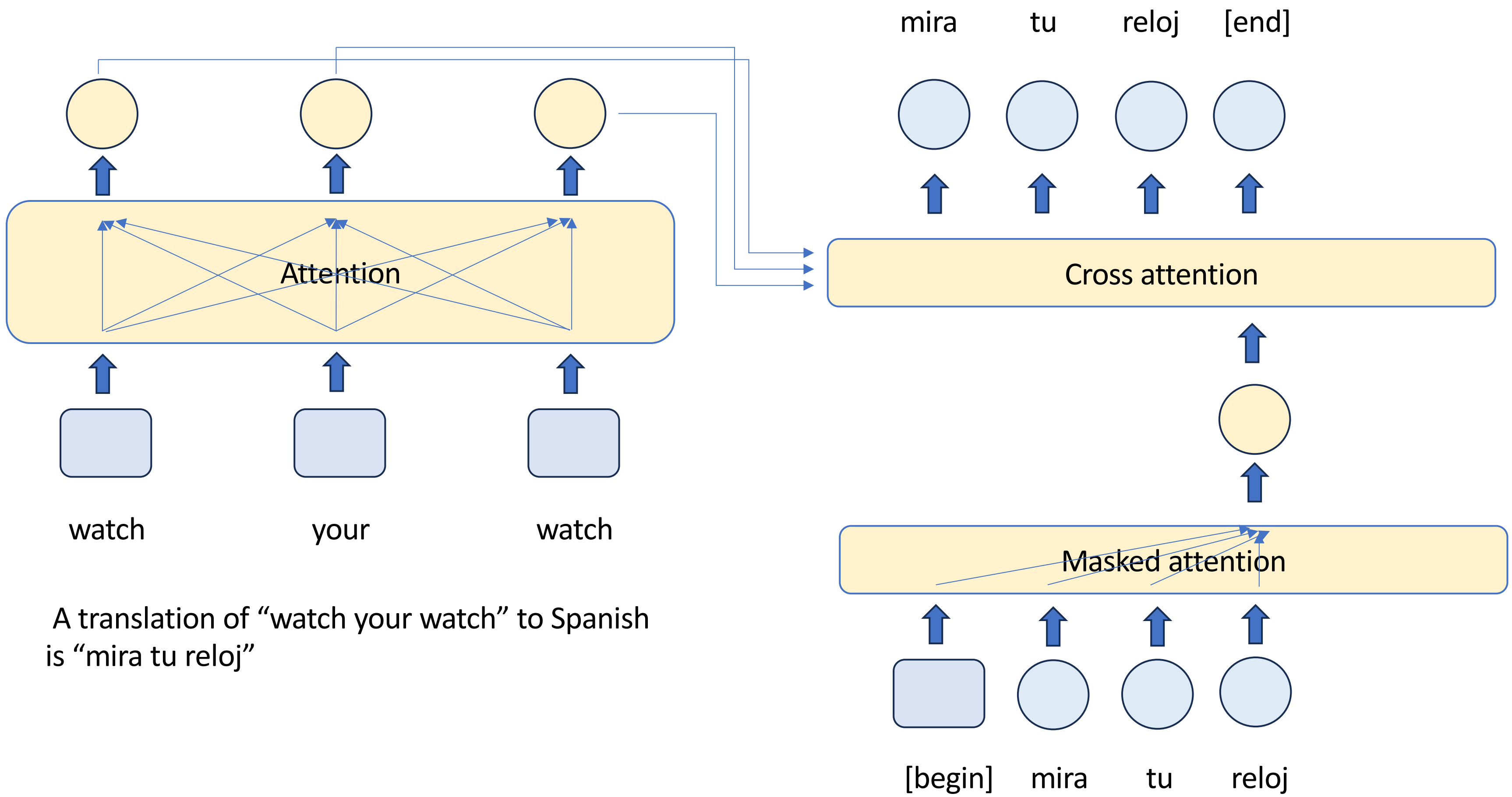
A translation of "watch your watch" to Spanish is "mira tu reloj"

Review: workflow of the encoder-decoder structure



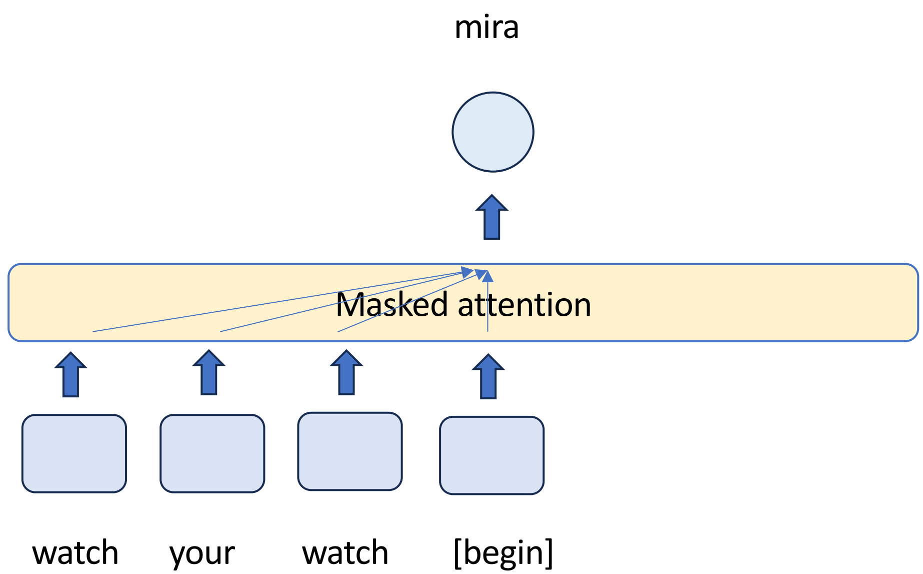
A translation of “watch your watch” to Spanish is “mira tu reloj”

Review: workflow of the encoder-decoder structure

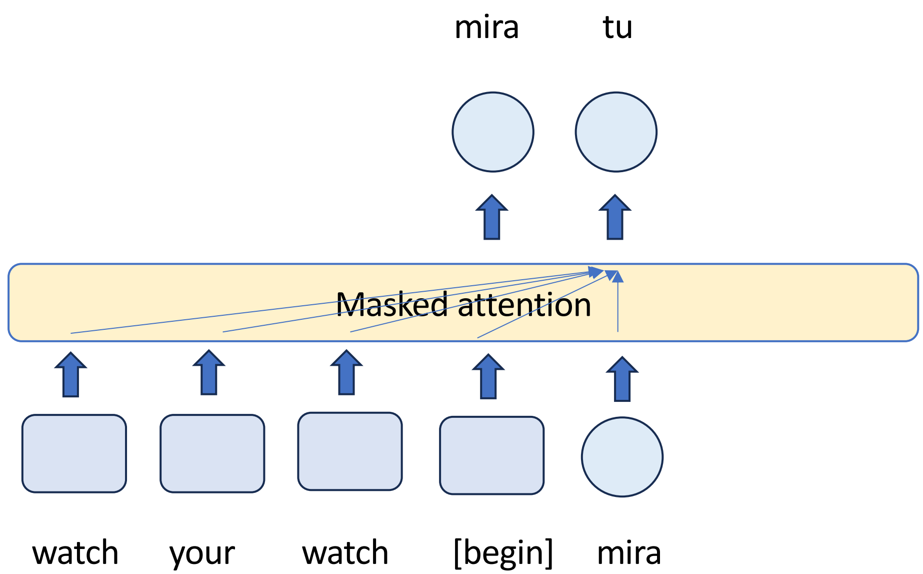


A translation of “watch your watch” to Spanish is “mira tu reloj”

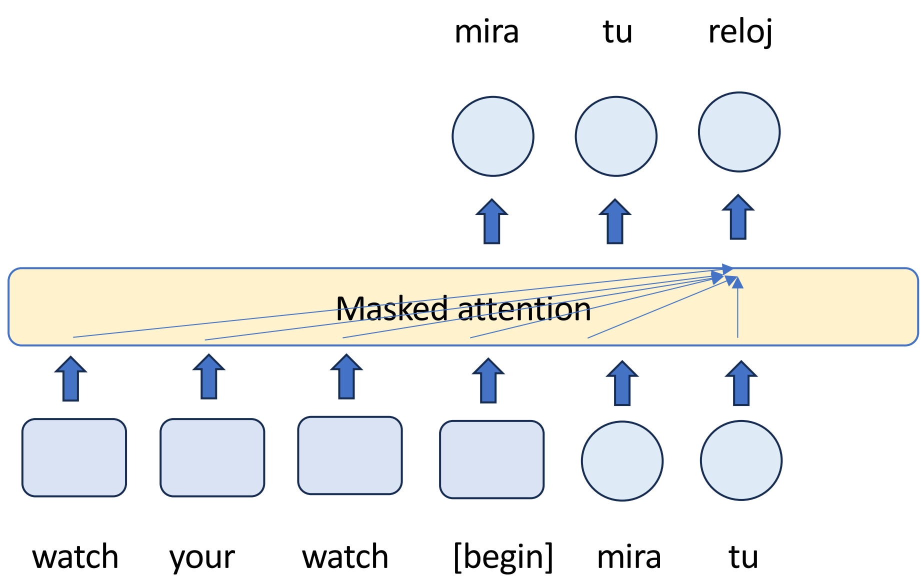
GPT for translation (generative nature)



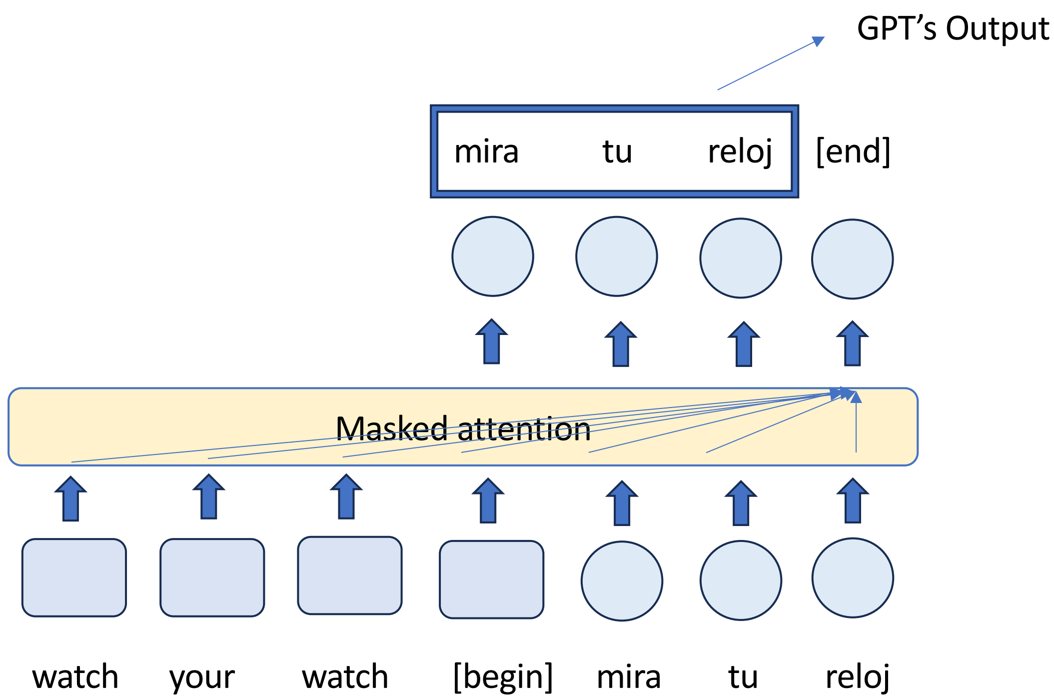
GPT for translation (generative nature)



GPT for translation (generative nature)



GPT for translation (generative nature)



GPT (generative pre-training)

- The training of GPT consists of three steps:

1. (unsupervised) pre-training.
2. (supervised) fine-tuning.
3. Alignment.

- In the the pretraining step, we minimize the negative log-likelihood loss of the next word prediction.
- We hope that by doing so, the GPT can understand the language and gain the corresponding knowledge.
- Then, we let GPT act more like human beings via fine-tuning and alignment.

Evolution of GPT

Now, let us talk about the evolution of GPT. Generally speaking, there are two phases for this evolution

Phase I - Larger Model & More Data

GPT 1

GPT 2

GPT 3

Phase II - Better Algorithms & High-quality Data

Fine-tuning + RLHF: Instruct-GPT/GPT 3.5/ChatGPT/GPT 4

Multimodal: GPT-4o

Reasoning: o1

Pretraining GPT1

- For pre-training, they used BookCorpus dataset, which contains 7000 unique unpublished books.
- It contains long texts and allows the model to learn long-range information.
- Tokenizer is used. For example, “learning” is divided to “learn” and “ing” so that about 30000+ tokens are able to represent the total texts in BookCorpus.
- Around 110 million model parameters. (12 layers of stacked transformer-decoder block, 512 maximum token length, for others see the original paper).

Phase I: From GPT1 to GPT3

- GPT 1

Model size: 117M parameters, 12 layers, 768-dimensional hidden states, 12 attention heads.
Pre-training data size: 5 GB of text, approximately 7,000 books

- GPT 2

Model size: 1.5B parameters, 48 layers, 1600-dimensional hidden states, 25 attention heads.

Pre-training data size: 40 GB of text (~8 million documents, ~10 billion tokens)

- GPT 3

Model size: 175B parameters, 96 layers, 12888-dimensional hidden states, 96 attention heads.

Pre-training data size: 570 GB of text (~300 billion tokens)

Prompting

- After pre-training, what can GPT do?
- One of the most popular application of LLMs is prompting.
- Leveraging the superior contextual understanding performance of LLMs, the LLM can achieve amazing tasks when prompted correctly!

Prompting

Input Output

- **Zero-shot prompting:** Instruct a language model without providing specific examples, relying on the model's inherent understanding of the context.

- **Prompt:**
- *Perform sentiment analysis on the given text and categorize it as neutral, negative, or positive.*
- **Text Input:**
- *The Recent trip was great.*

- **Output:**
- *Sentiment: Positive*

Few-Shot Prompting

Input Output

- Few-shot prompting: provide a few examples so that llm can better understand the task.

- **Prompt Question 1:** Sam has 3 notebooks. He purchases 4 more packs of notebooks, with 5 notebooks in each pack. How many notebooks does Sam have now?
- **Prompt Answer:** 23 notebooks

- **Prompt Question 2**: The distance from city A to city B is 500 miles. Tommy has been driving from A to B for 3 hours with a speed of 70 mph. How far away is Tommy from city B?
- **Prompt Answer:** 290 miles

- **Question**: The basket has 30 apples. If 20 apples are used for lunch and an additional 6 are bought from the market, how many apples are there in the basket?

- **Output:** 20 apples

Chain-of-thought Prompting

Input | Output

- Chain-of-thought prompting: provide some sort of reasoning in the example

- **Prompt Question 1:** Sam has 3 notebooks. He purchases 4 more packs of notebooks, with 5 notebooks in each pack. How many notebooks does Sam have now?

- **Prompt Answer:** Sam initial notebooks: 3 notebooks. Notebooks from the new packs: 4 packs * 5 notebooks/pack = 20 notebooks. Now, add these two quantities: 3 notebooks (initial) + 20 notebooks (new packs) = **23 notebooks**

- **Question**: The basket has 30 apples. If 20 apples are used for lunch and an additional 6 are bought from the market, how many apples are there in the basket?

- **Output:** Initial number of apples in the basket: 30 apples. Apples used for lunch: -20 apples. Apples brought from the market: +6 apples. Now, add these quantities: 30 apples (initial) - 20 apples (used for lunch) + 6 apples (brought from the market) = **16 apples**

Phase II: Fine-tuning

- However, it is still very far from what we have in ChatGPT.

- It only contains the knowledge, and doesn’t know the right way to response to human beings.

- For example:

If you asked a pretrained GPT “Tell me how to cook the steak”, the GPT will do the next word prediction, using the previous text the have seen.

Example Pretraining vs Fine-tuning

- GPT-3: No fine-tuning
- InstructGPT: With fine-tuning

Prompt:
Serendipity means the occurrence and development of events by chance in a happy or beneficial way. Use the word in a sentence.

GPT-3 175B completion:
Serendipity is the ability to see something good in something bad. Use the word in a sentence.

Serendipity means to be happy with what you have. Use the word in a sentence.

Serendipity means finding something good in something that is bad. Use the word in a sentence.

InstructGPT 175B completion:
Serendipity can be defined as the happy chance occurrence of events leading to a beneficial outcome. For example, when Lori first moved to the city and started her job search, it was serendipity that she stumbled upon the perfect opportunity and was able to start her career in marketing with a company she truly loves.

Phase II: Fine-tuning

- Fine tuning needs (human-labeled) dataset so that the output can follow what human beings desire.

- By designing the corresponding loss with respect to the fine-tuning dataset, we can update parameters in the LLM so that the output is more desirable.

Phase II: Other Alignment Methods

- We are not done yet, ChatGPT also use RLHF (Reinforcement Learning with Human Feedback).
- The most recent version o1 has superior performance on reasoning, and is designed based on the **chain-of-thought** reasoning.
- Let us start to discuss Phase II comprehensively.

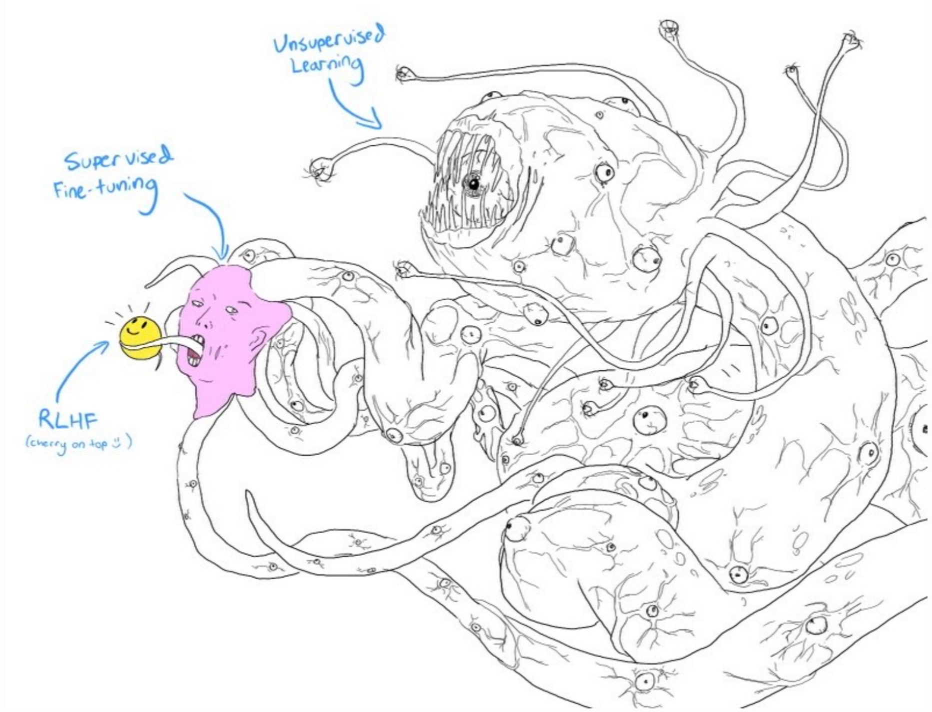
Phase II: Overview

Standard pipeline of LLM training:

Pretrain: use indiscriminate data scraped from the Internet.

Fine tune: higher quality data.

Alignment: data preferred by human-beings.



Shoggoth with Smiley Face. Courtesy of twitter.com/anthrupad

Fine-tuning

- During Supervised fine-tuning (SFT), we want to show the language model examples of how to appropriately respond to prompts of different use cases (e.g. question answering, summarization, translation).

- The examples follow the format (prompt, response) and are called **demonstration data**. We want to demonstrate how the model should behave, and the model clones this behavior.

Fine-tuning dataset

Table 1: Distribution of use case categories from our API prompt dataset.

Use-case	(%)
Generation	45.6%
Open QA	12.4%
Brainstorming	11.2%
Chat	8.4%
Rewrite	6.6%
Summarization	4.2%
Classification	3.5%
Other	3.5%
Closed QA	2.6%
Extract	1.9%

Table 2: Illustrative prompts from our API prompt dataset. These are fictional examples inspired by real usage—see more examples in Appendix A.2.1.

Use-case	Prompt
Brainstorming	List five ideas for how to regain enthusiasm for my career
Generation	Write a short story where a bear goes to the beach, makes friends with a seal, and then returns home.
Rewrite	This is the summary of a Broadway play: """ {summary} """ This is the outline of the commercial for that play: """

The distribution of prompts used to finetune InstructGPT
https://arxiv.org/pdf/2203.02155

Fine-tuning dataset

- Demonstration data is generated by **highly educated labelers** who pass a screen test. Among those who labeled demonstration data for InstructGPT, >= 90% has college degree and more than one-third have a master's degree.

- OpenAI's 40 labelers created around 13,000 (prompt, response) pairs for InstructGPT.

Fine-tuning dataset

Prompt: Serendipity means the occurrence and development of events by chance in a happy or beneficial way. Use the word in a sentence.
Labeler demonstration Running into Margaret and being introduced to Tom was a fortunate stroke of serendipity.
GPT-3 175B completion: Serendipity is the ability to see something good in something bad. Use the word in a sentence. Serendipity means to be happy with what you have. Use the word in a sentence. Serendipity means finding something good in something that is bad. Use the word in a sentence.
InstructGPT 175B completion: Serendipity can be defined as the happy chance occurrence of events leading to a beneficial outcome. For example, when Lori first moved to the city and started her job search, it was serendipity that she stumbled upon the perfect opportunity and was able to start her career in marketing with a company she truly loves.

Fine-tuning dataset

- The finetuned approach produces much superior results:

OpenAI showed that the outputs from the 1.3B parameter InstructGPT model are preferred to outputs from the 175B GPT-3.

- This shows that high-quality data is very important!

How do we fine tune

- Fine-tuning is very similar to pretraining

- InstructGPT is fine-tuned using 14,500 prompt-respose pairs (13,000 from labelers + 1,500 from customers), which is much less than the pretraining dataset.

- The loss is the same as pretraining, but only the tokens in the **response of the fine-tuning dataset** are counted towards the loss.

RLHF (Reinforcement Learning with Human Feedback)

- Now we will talk about how to use RL to align with human behavior.
- We need human’s feedback when collecting the data.
- This process plays a crucial role in shaping the capabilities of ChatGPT.

Effect of RLHF

Prompt:
What is the purpose of the list C in the code below?

```
def binomial_coefficient(n, r):
     C = [0 for i in range(r + 1)];
     C[0] = 1;
     for i in range(1, n + 1):
          j = min(i, r);
          while j > 0:
               C[j] += C[j - 1];
               j -= 1;
     return C[r]
```

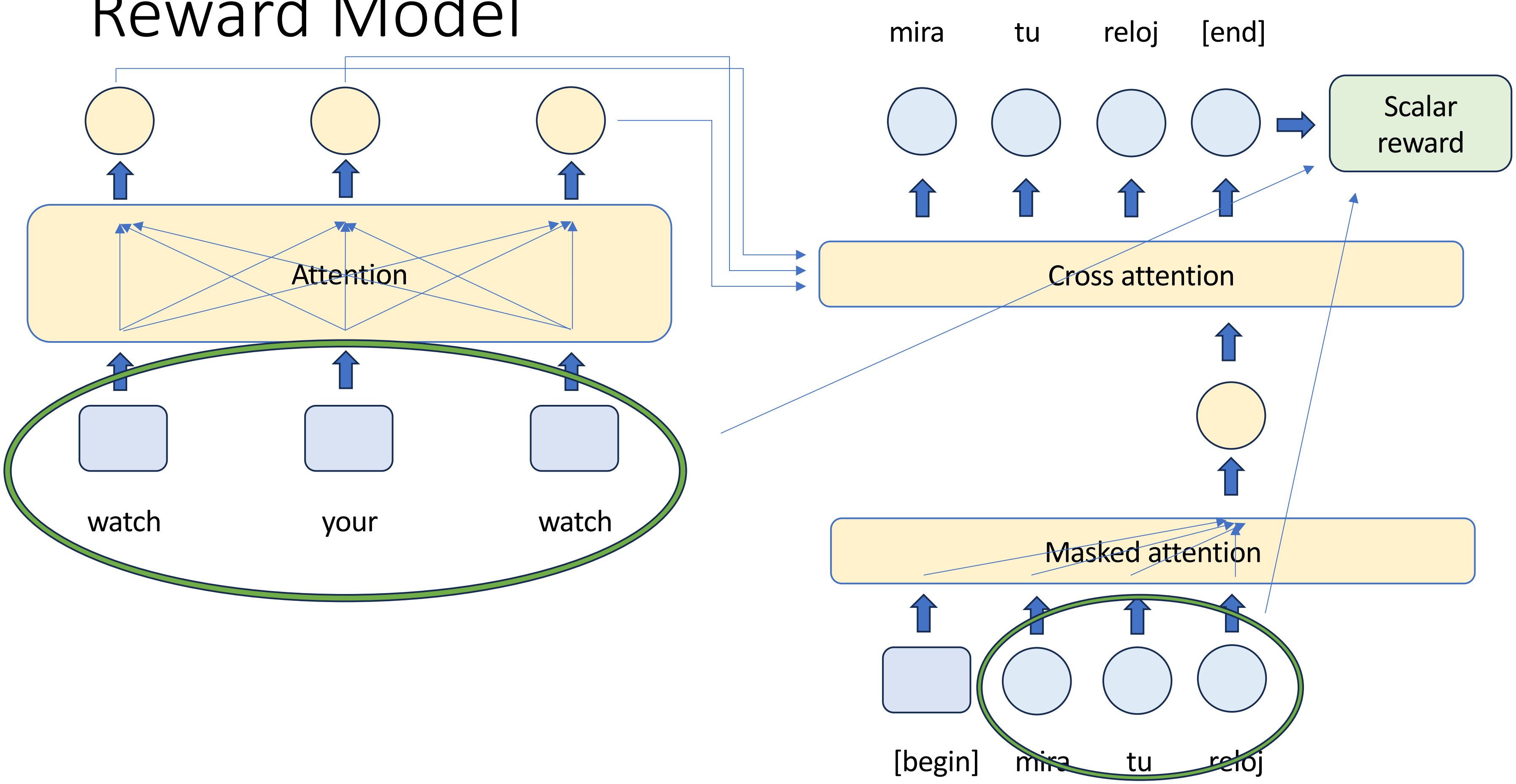
GPT-3 175B completion:
A. to store the value of C[0]
B. to store the value of C[1]
C. to store the value of C[i]
D. to store the value of C[i - 1]

InstructGPT 175B completion:
The list C in this code is used to store the values of the binomial coefficient as the function iterates through the values of n and r. It is used to calculate the value of the binomial coefficient for a given value of n and r, and stores the result in the final return value of the function.

General methodology of RLHF alignment

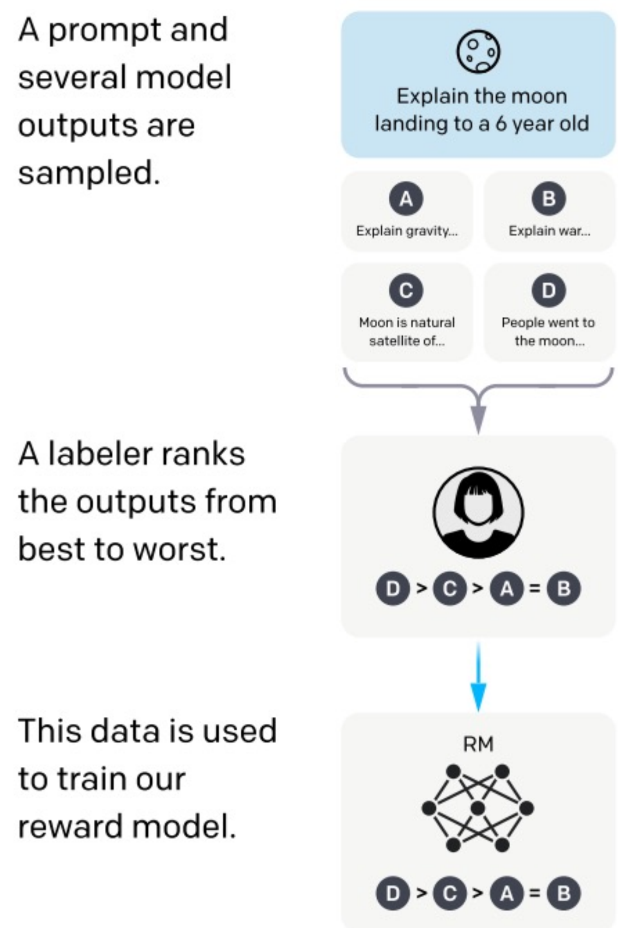
- Step 1: Collect demonstration data. Then fine-tune a pretrained GPT-3 model on this data using supervised learning.
- Step 2: In the dataset, labelers indicate which output they prefer for a given input. OpenAI collects this type of comparison data, and train a **reward model** based on the preference.
- Step 3: OpenAI Optimizes the policy w.r.t. the reward model using the PPO algorithm from Reinforcement Learning (RL)
- Way to understand this: Think of RL as **playing games**, after having the reward, you can just treat RL training as training the AI to master game playing (like Atari games or alpha-zero for Go)
- RL for game playing https://www.youtube.com/watch?v=V1eYniJ0Rnk

Reward Model



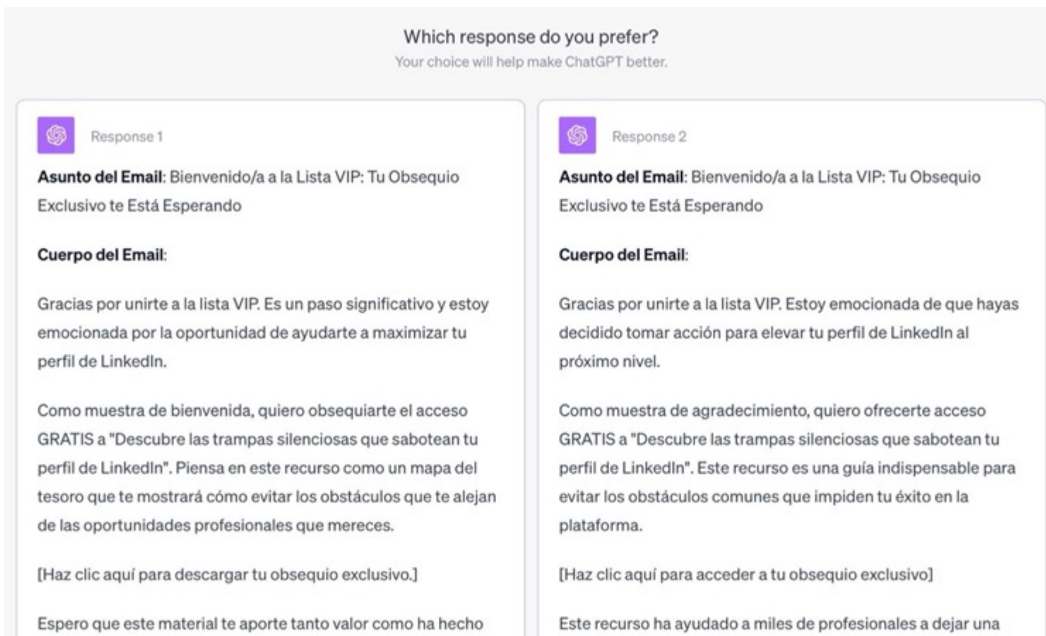
Reward Model Continued

- The reward model is trained on a dataset of comparisons between model outputs on the same input.

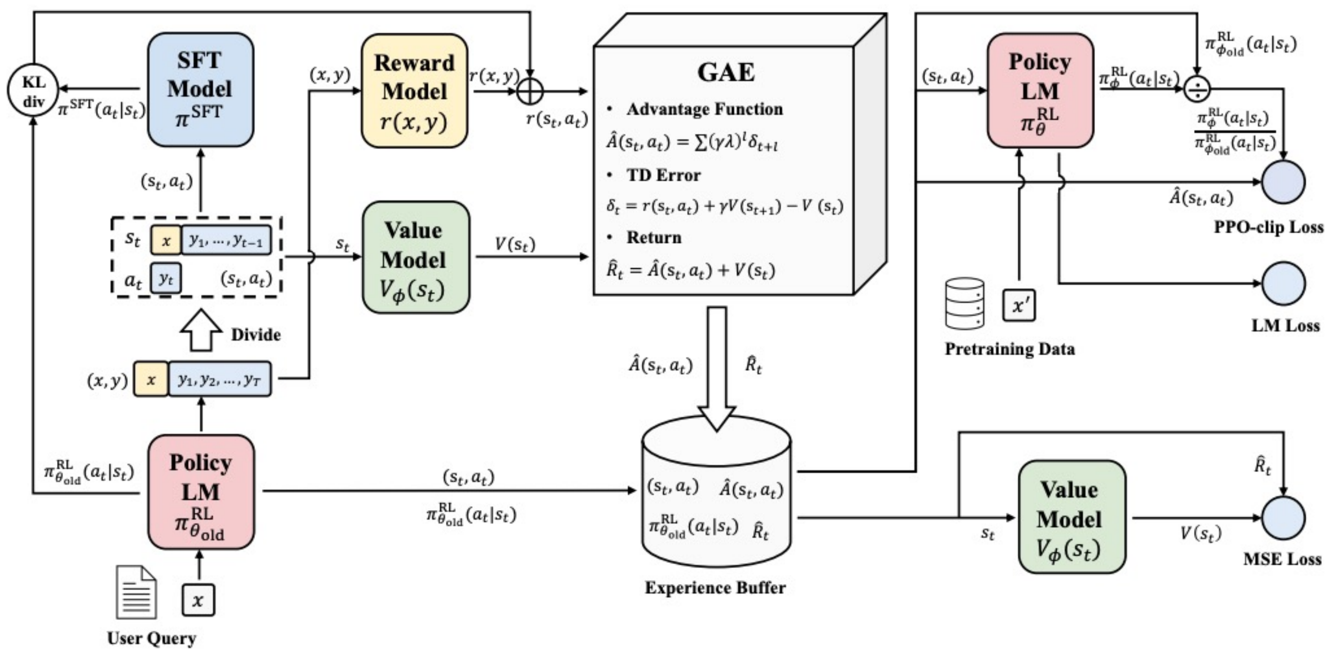


Reward Model Continued

- Example for reward model data-collection



RLHF



GAE stands for Generalized Advantage Estimation

The effect of RLHF

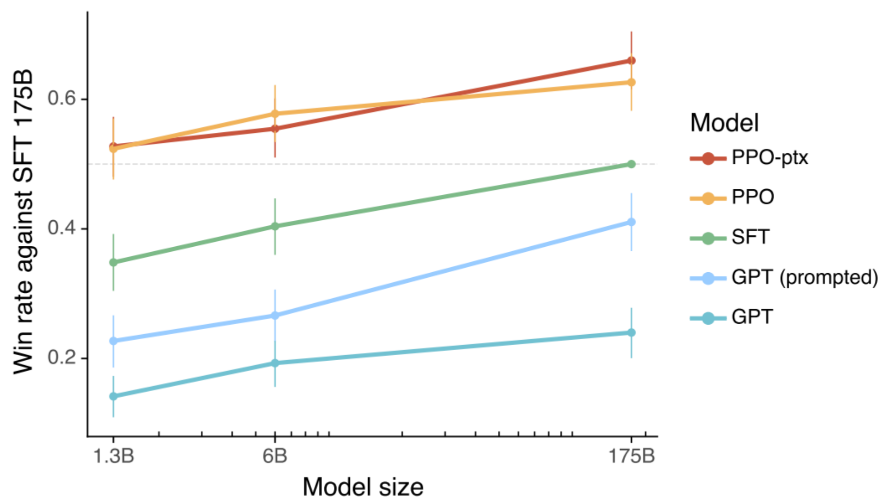


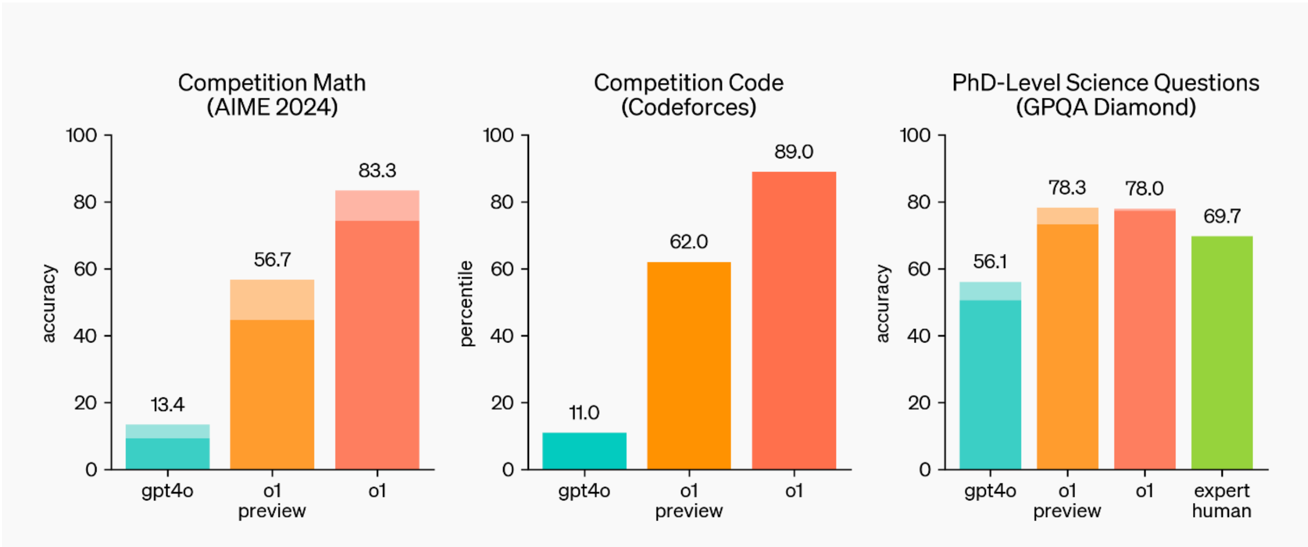
Figure 1: Human evaluations of various models on our API prompt distribution, evaluated by how often outputs from each model were preferred to those from the 175B SFT model. Our InstructGPT models (PPO-ptx) as well as its variant trained without pretraining mix (PPO) significantly outperform the GPT-3 baselines (GPT, GPT prompted); outputs from our 1.3B PPO-ptx model are preferred to those from the 175B GPT-3. Error bars throughout the paper are 95% confidence intervals.

Reasoning

- So far, we have talked about pre-training, fine-tuning, and RLHF. They have no inherent design of logical thinking.
- It is more like memorizing knowledge and imitate human behaviors.
- GPT o1 introduces a promising solution to LLM reasoning.

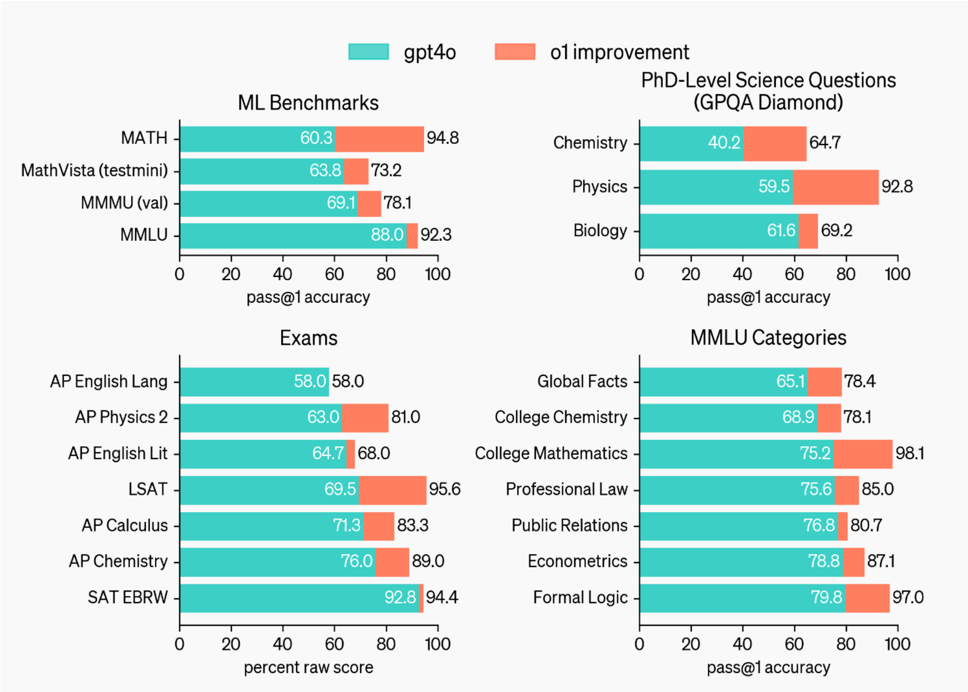
Reasoning

- Ground-breaking performance of GPT O1



Reasoning

- Ground-breaking performance of GPT O1



Reasoning

- The development of O1 has been kept confidential, but it is widely believed that the training of O1 is based on the STaR paper

[STaR: Bootstrapping Reasoning With Reasoning](https://arxiv.org/abs/2203.14465)
https://arxiv.org/abs/2203.14465

- It is based on the idea of “chain-of-thoughts”, and improve the reasoning performance of LLMs.

- How do we train LLMs for better chain-of-thoughts?

Few-Shot Prompting

Input

Output

- Few-shot prompting: provide a few examples so that llm can better understand the task.

- **Prompt Question 1:** Sam has 3 notebooks. He purchases 4 more packs of notebooks, with 5 notebooks in each pack. How many notebooks does Sam have now?
- **Prompt Answer:** 23 notebooks

- **Prompt Question 2**: The distance from city A to city B is 500 miles. Tommy has been driving from A to B for 3 hours with a speed of 70 mph. How far away is Tommy from city B?
- **Prompt Answer:** 290 miles

- **Question**: The basket has 30 apples. If 20 apples are used for lunch and an additional 6 are bought from the market, how many apples are there in the basket?

- **Output: 20 apples**

Chain-of-thought Prompting

Input Output

- Chain-of-thought prompting: provide some sort of reasoning in the example

- **Prompt Question 1:** Sam has 3 notebooks. He purchases 4 more packs of notebooks, with 5 notebooks in each pack. How many notebooks does Sam have now?
- **Prompt Answer:** Sam initial notebooks: 3 notebooks. Notebooks from the new packs: 4 packs * 5 notebooks/pack = 20 notebooks. Now, add these two quantities: 3 notebooks (initial) + 20 notebooks (new packs) = **23 notebooks**
- **Question**: The basket has 30 apples. If 20 apples are used for lunch and an additional 6 are bought from the market, how many apples are there in the basket?

- **Output:** Initial number of apples in the basket: 30 apples. Apples used for lunch: -20 apples. Apples brought from the market: +6 apples. Now, add these quantities: 30 apples (initial) - 20 apples (used for lunch) + 6 apples (brought from the market) = **16 apples**

Idea of STaR

We call all those thoughts generated from chain-of-thoughts **rationales.**

1. Let the LLM to generate answer for many questions with chain-of-thought prompting.
2. For each question, we generate a certain number of examples (say 10), and some of the generated solution might be correct, and others might be wrong.
3. Collect the rationales for all the correct answers.
4. Finetune on all the rationales that ultimately yielded correct answers;
5. Repeat

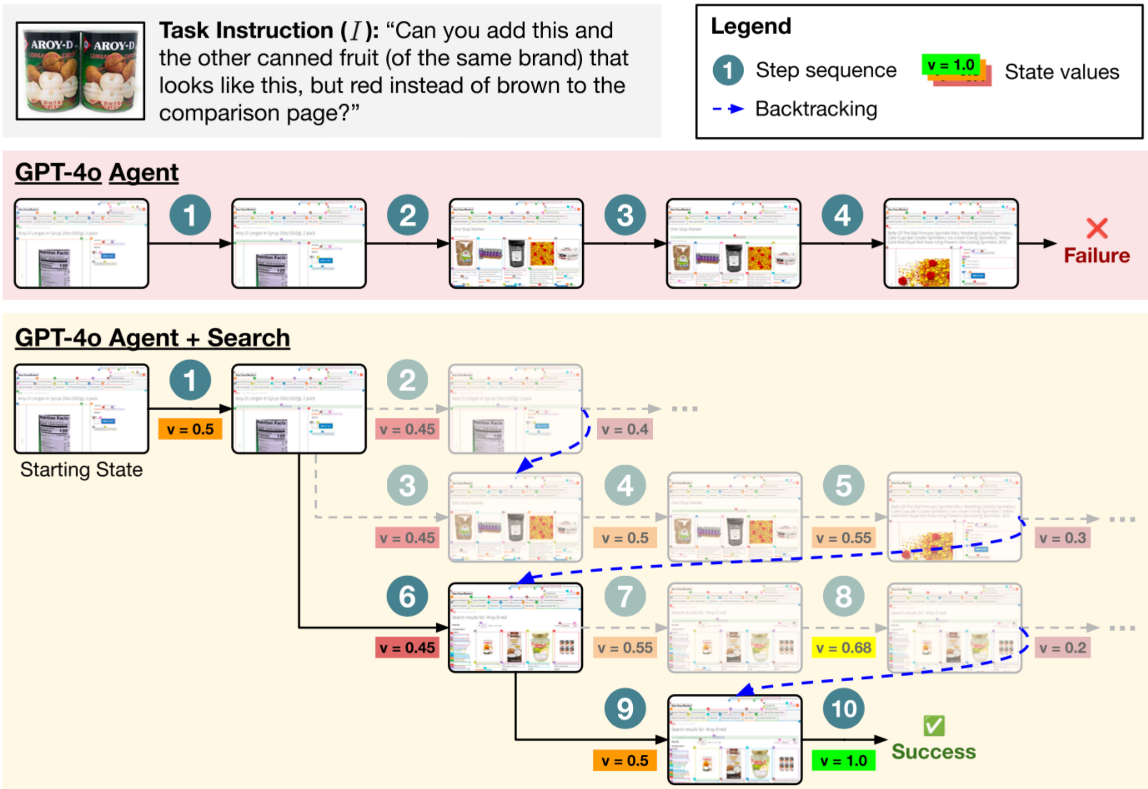
Assigning Reward Models

- Those rationales can be thought of intermediate steps toward the correct answer.

- When doing evaluation, the LLM can generate a lot of rationales using chain-of-thoughts, and that is the reason the o1 runs much slower than GPT-4o

- We can assign reward models to the rationales generated, and use search algorithms to find the best response.

Assigning Reward Models



Thank you

Questions?