

15-494/694: Cognitive Robotics

Dave Touretzky

Lecture 16:

Transformer Networks
and Large Language
Models

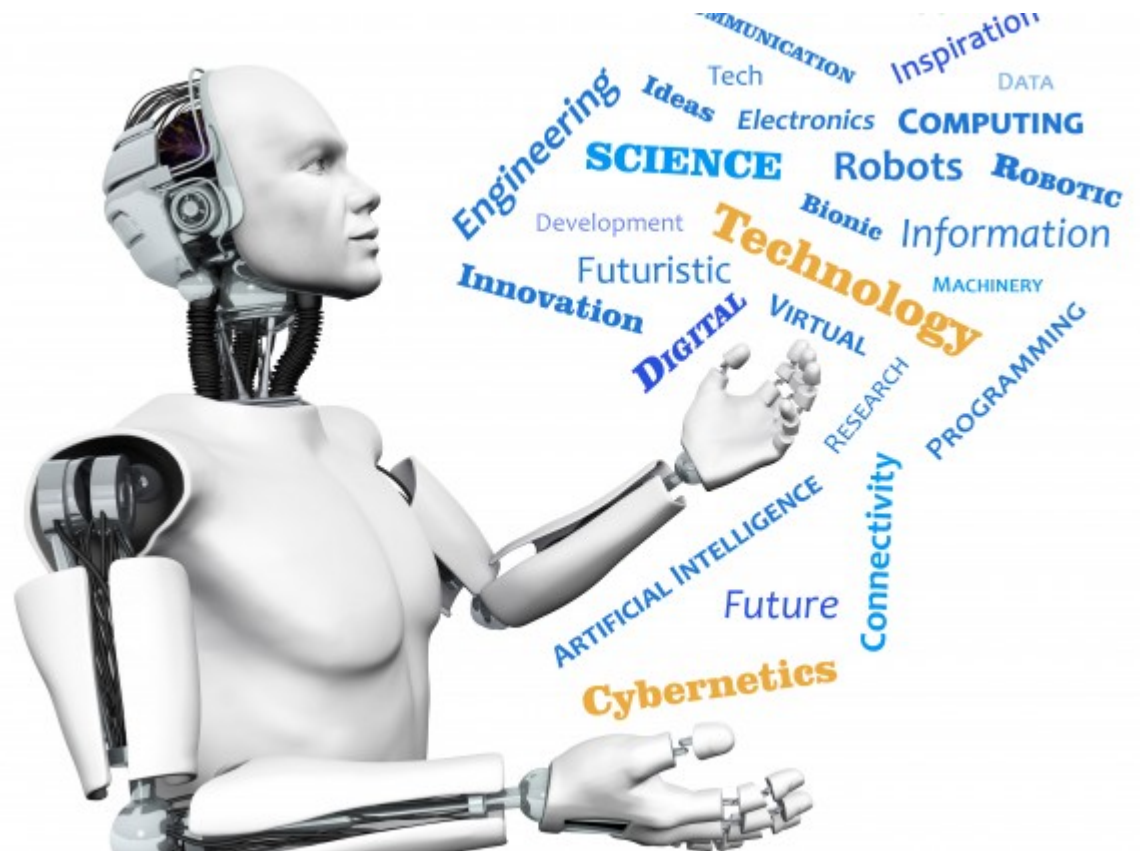


Image from <http://www.futuristgerd.com/2015/09/10>

Outline

- What are Large Language Models?
- Embeddings
 - Word Embedding Demo
- Recurrent networks: sequential behavior
- Transformer networks and attention
- BERT question answering
 - BERT-insight demo
- ChatGPT and GPT-4

Large Language Models

- Deep neural networks trained on massive amounts of text:
 - All of English Wikipedia
 - Large chunks of Reddit
 - Hundreds of books
 - Thousands of newspaper articles
- Trained to do what?
 - Predict missing words in text.
 - Predict the next word in a sentence.
 - Other language prediction tasks.

Why Are LLMs Interesting?

- When they got large enough (billions of parameters), they started to exhibit reasoning capabilities.
- They don't reason as well as human beings... yet.
- They can solve problems, translate between languages, and even write code.

Who Is Creating LLMs?

- Google: BERT, T5, LaMDA, PaLM, Gemini
- OpenAI: GPT-3, GPT-4, ChatGPT
- Anthropic: Claude
- Microsoft: Bard, Copilot
- Facebook/Meta: RoBERTa, LLaMa
- Amazon: AlexaTM 20B
- Baidu: PCL-BAIDU Wenxin (Ernie 3.0)
- DeepSeek: DeepSeek-R1, DeepSeek-V3

Reasoning

DS

Can I substitute margarine for yogurt in a recipe?



Margarine and yogurt are two very different ingredients, so they are not interchangeable in most recipes. Margarine is a butter substitute, while yogurt is a dairy product that adds a creamy texture and tangy flavor to recipes. Substituting margarine for yogurt can significantly alter the flavor, texture, and moisture content of a recipe, and is not recommended.

If you are looking for a substitute for yogurt in a recipe, it's best to choose an ingredient that has a similar flavor and texture. Some options include sour cream, buttermilk, silken tofu, or a non-dairy yogurt alternative.





Write a Python function to make all the vowels in a string be uppercase.



Here's a Python function that makes all the vowels in a string uppercase:



python

Copy code

```
def uppercase_vowels(string):  
    vowels = ['a', 'e', 'i', 'o', 'u']  
    result = ''  
    for char in string:  
        if char.lower() in vowels:  
            result += char.upper()  
        else:  
            result += char  
    return result
```

This function takes a string as input and iterates over each character in the string. If the character is a vowel (either uppercase or lowercase), it is replaced with its uppercase version using the `upper()` string method. Otherwise, the character is left unchanged. The resulting string is returned.

To use this function, you can call it with a string argument:

python

 Copy code

```
string = "hello world"
result = uppercase_vowels(string)
print(result) # "hEllo wOrld"
```

This will output the input string with all vowels capitalized.

DS

It's not necessary to convert char to lowercase for this code to work. Can you write a simpler version?



Yes, you are correct that it's not necessary to convert the character to lowercase in this case, since we are only looking for vowels and not distinguishing between uppercase and lowercase letters.



Here is a simplified version of the function that doesn't require converting the characters to lowercase:

python

Copy code

```
def uppercase_vowels(string):  
    vowels = "aeiouAEIOU"  
    result = ""  
    for char in string:  
        if char in vowels:  
            result += char.upper()  
        else:  
            result += char  
    return result
```

Wouldn't the new code still work if vowels were just 'aeiou' ?



Yes, you are correct. In fact, if we only want to capitalize lowercase vowels in the input string, we can further simplify the function and just use the string "aeiou" as the vowels string, like this:



python

Copy code

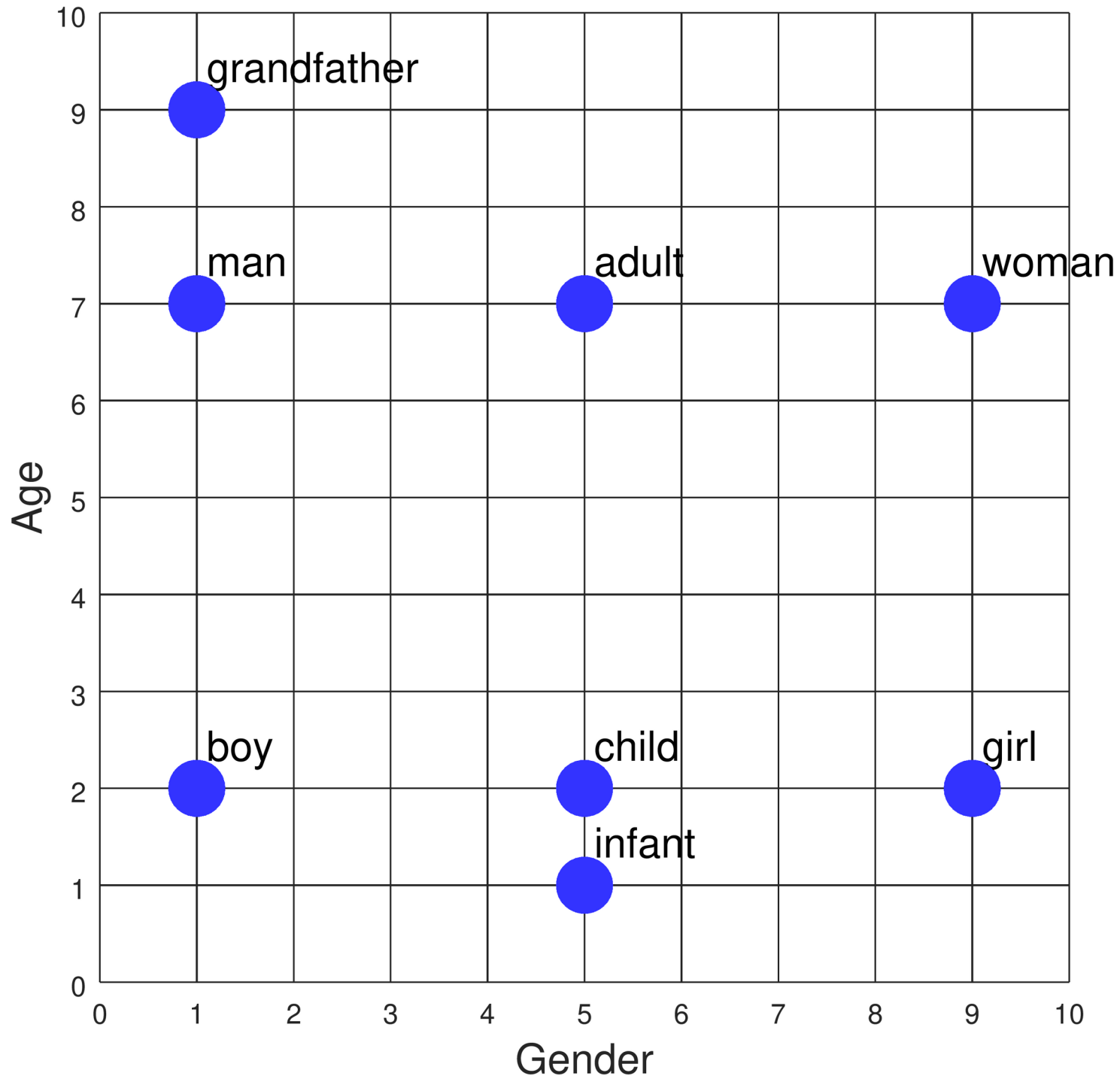
```
def uppercase_vowels(string):  
    vowels = "aeiou"  
    result = ""  
    for char in string:  
        if char in vowels:  
            result += char.upper()  
        else:  
            result += char  
    return result
```

This version works in the same way as the previous one, but it only looks for lowercase vowels in the input string and capitalizes them. All other characters are left unchanged.

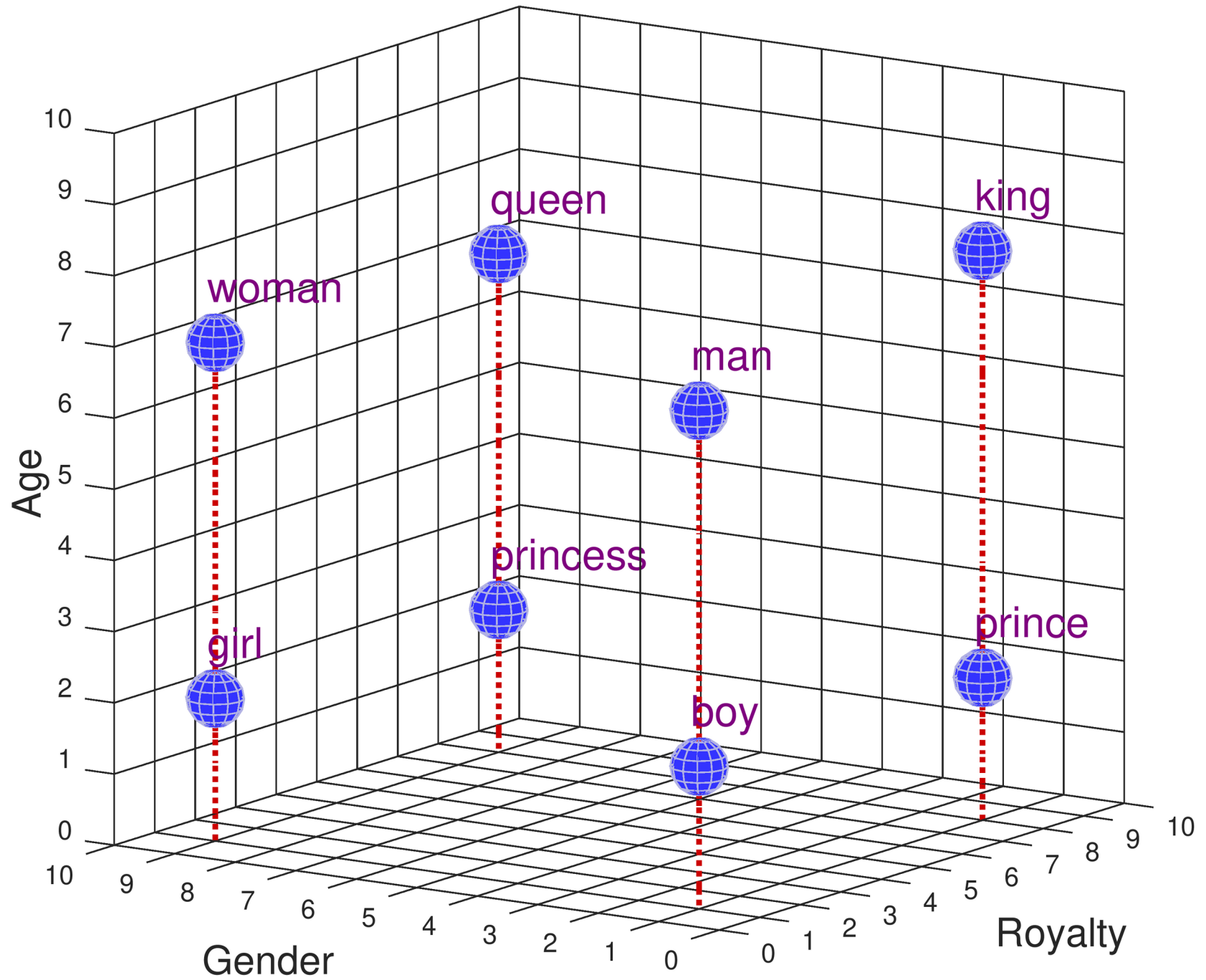
How Do LLMs Work?

- We don't know yet. This is a hot research question right now.
- Two key concepts:
 - Embeddings
 - Attention

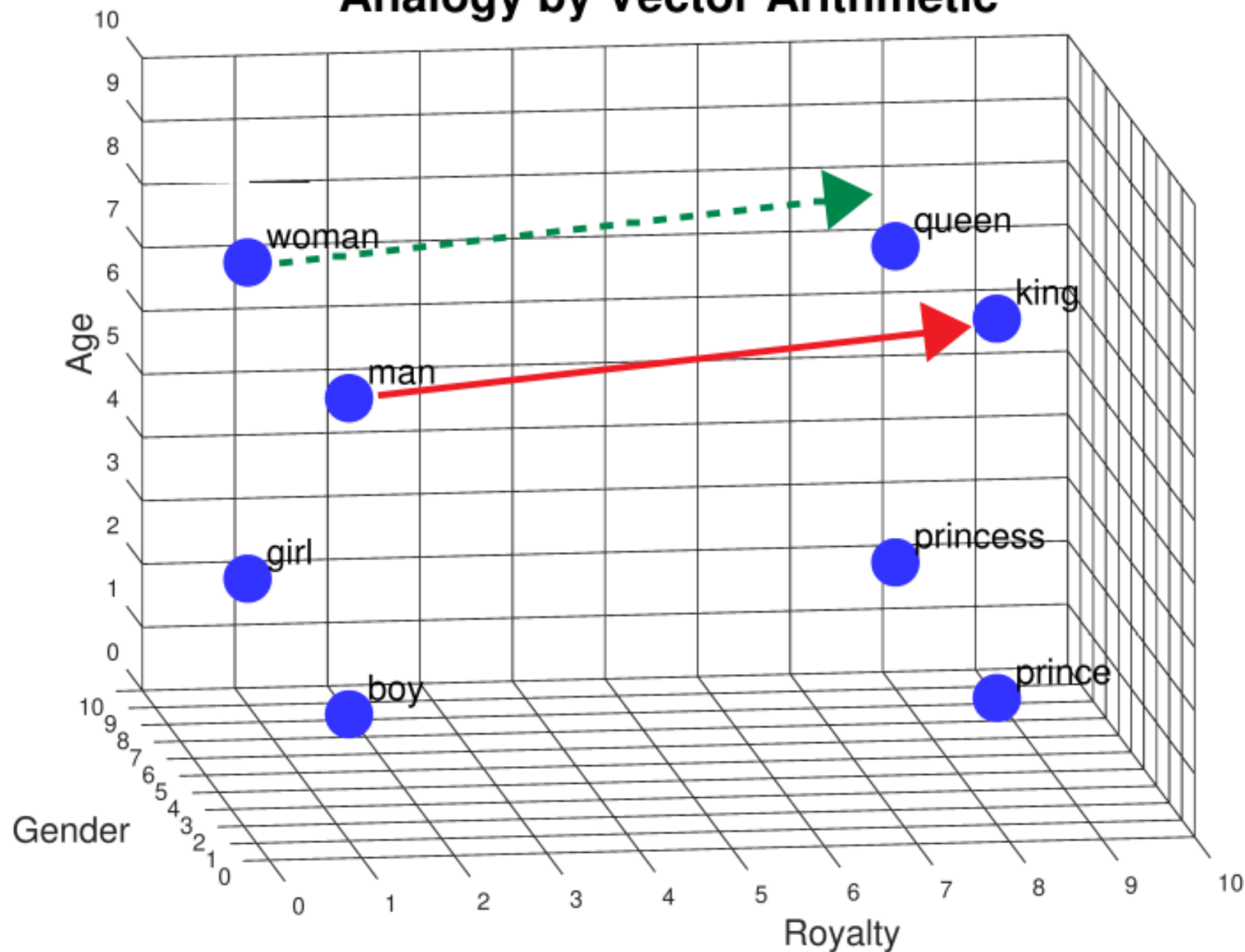
Semantic Feature Space



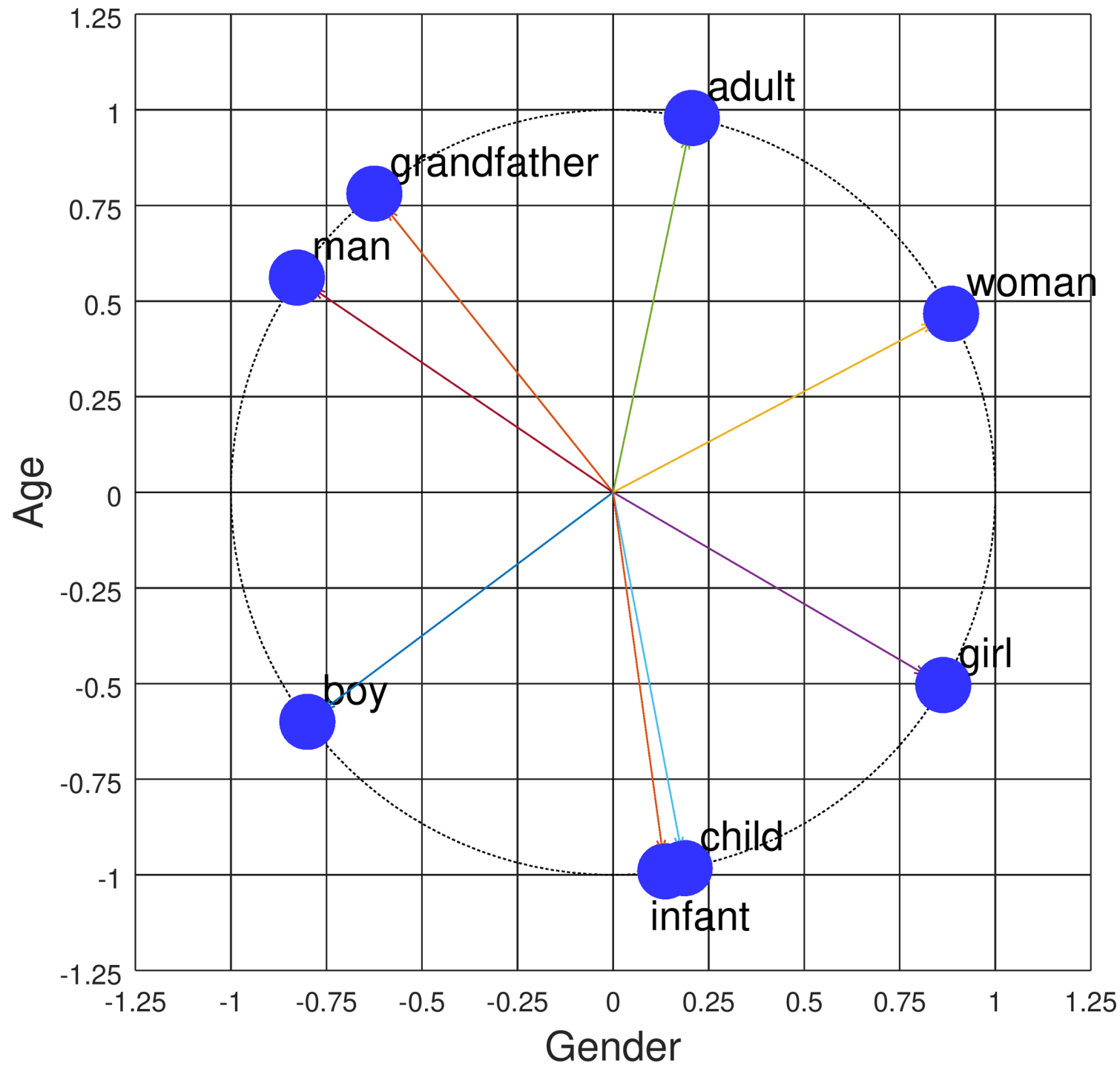
3D Semantic Feature Space



Analogy by Vector Arithmetic



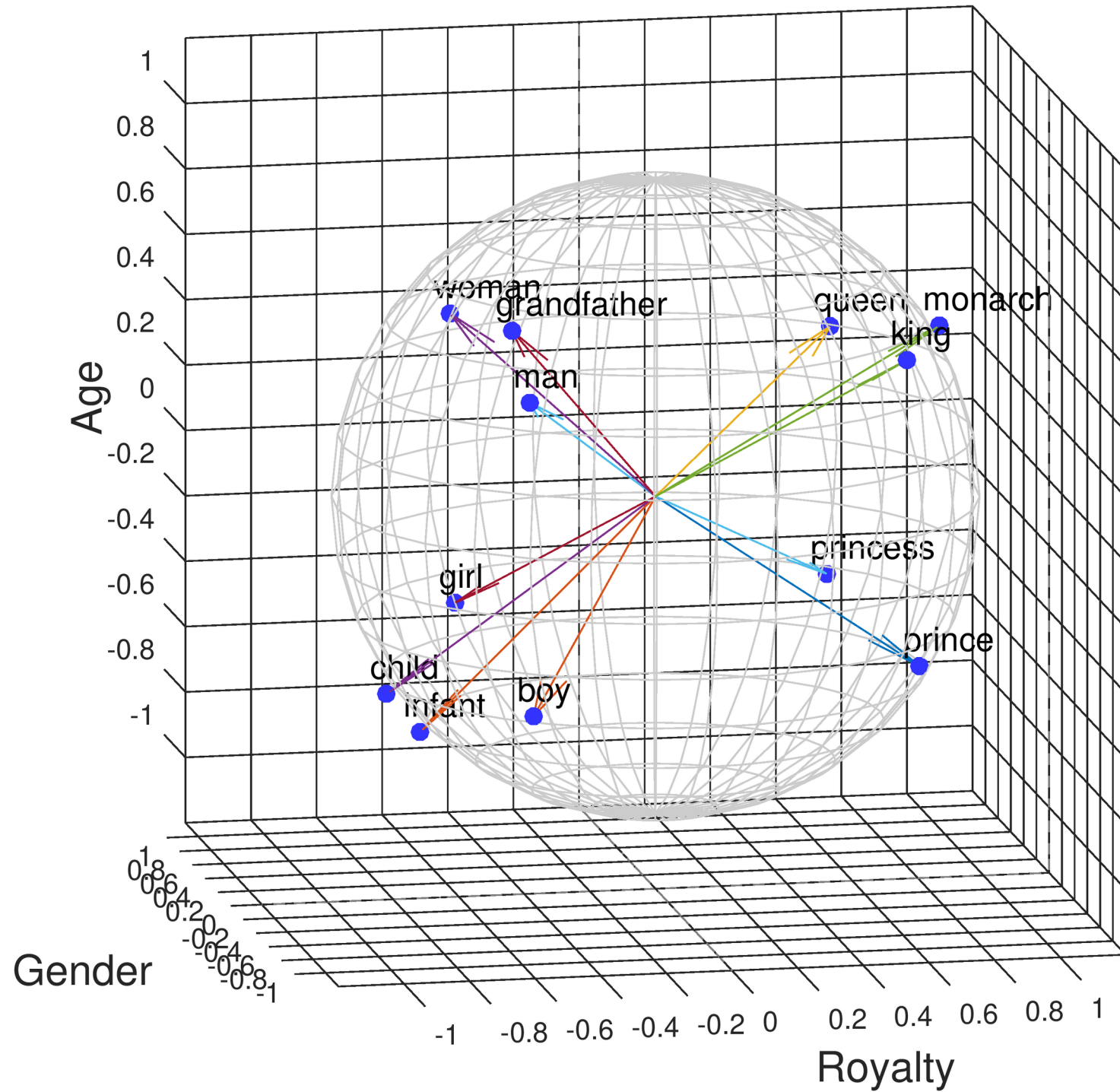
Zero-Mean 2D Unit Vectors

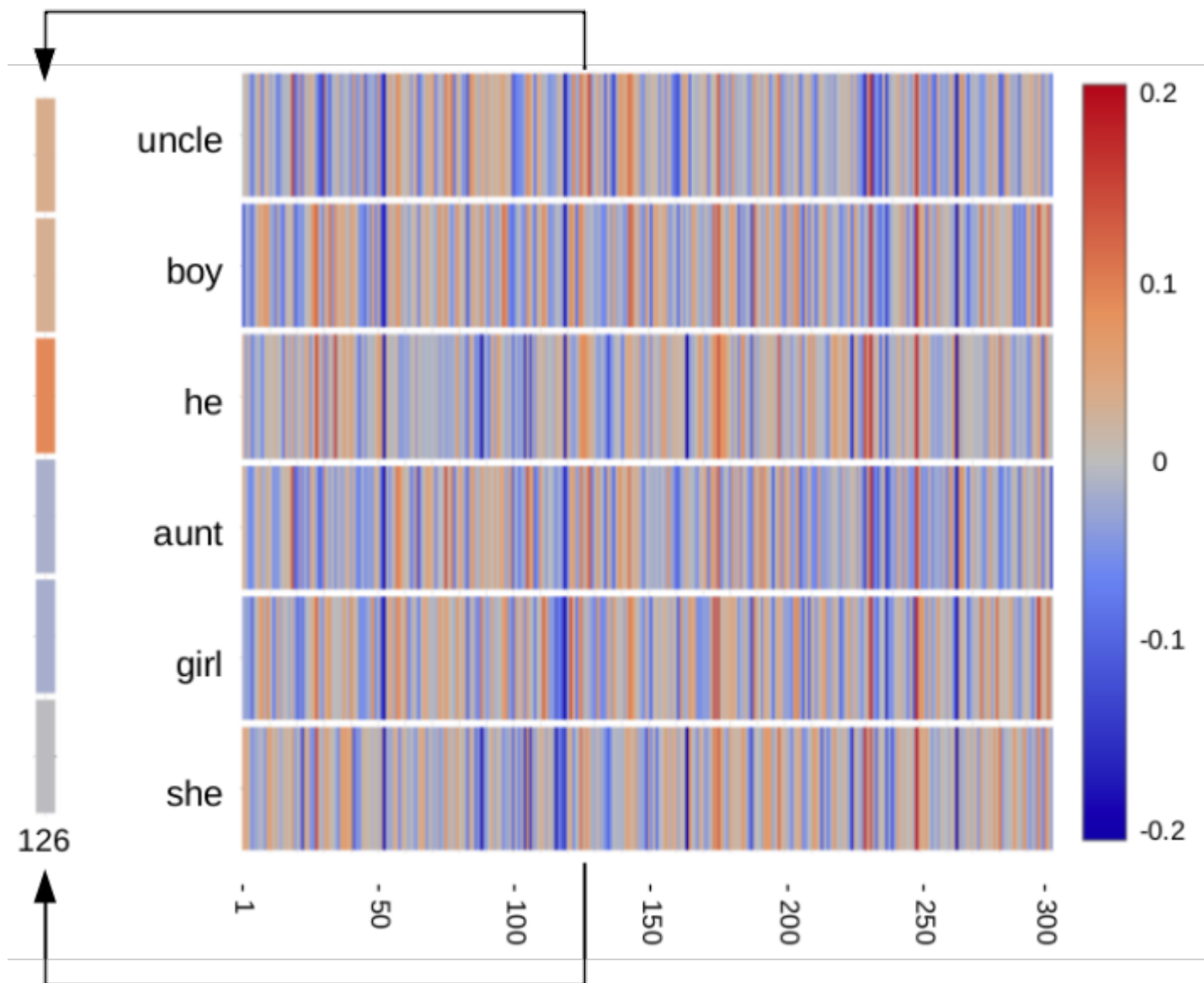


Similarity
measure:

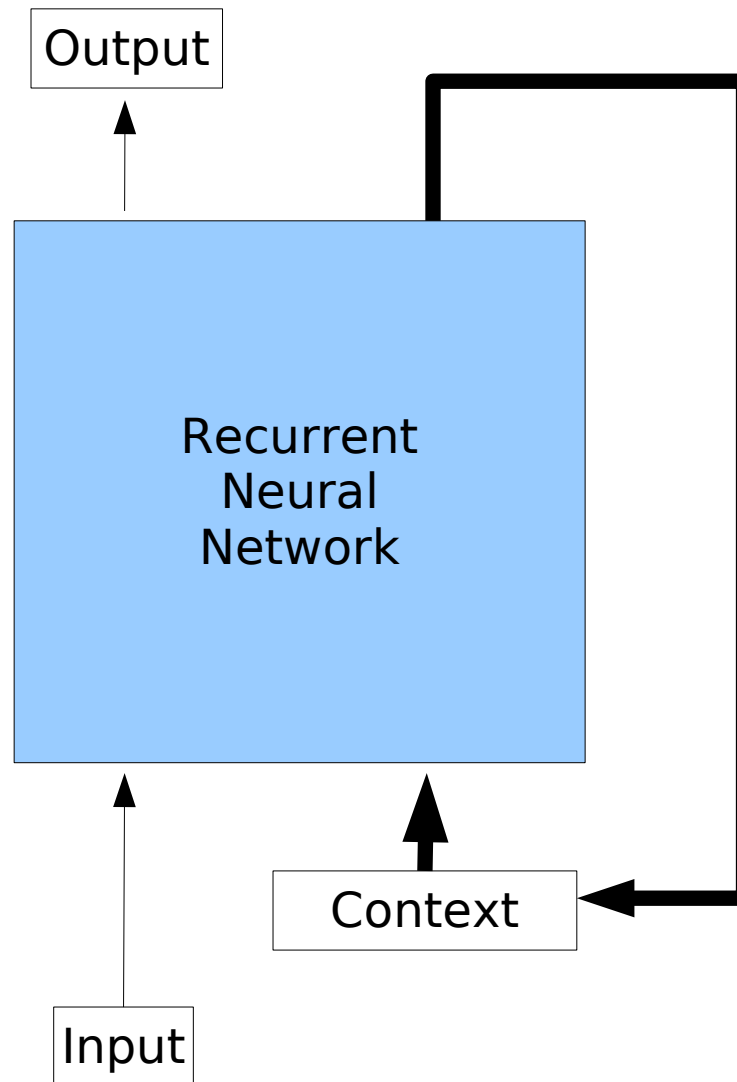
dot product is
 $\cos(\theta)$

Zero-Mean 3D Unit Vectors

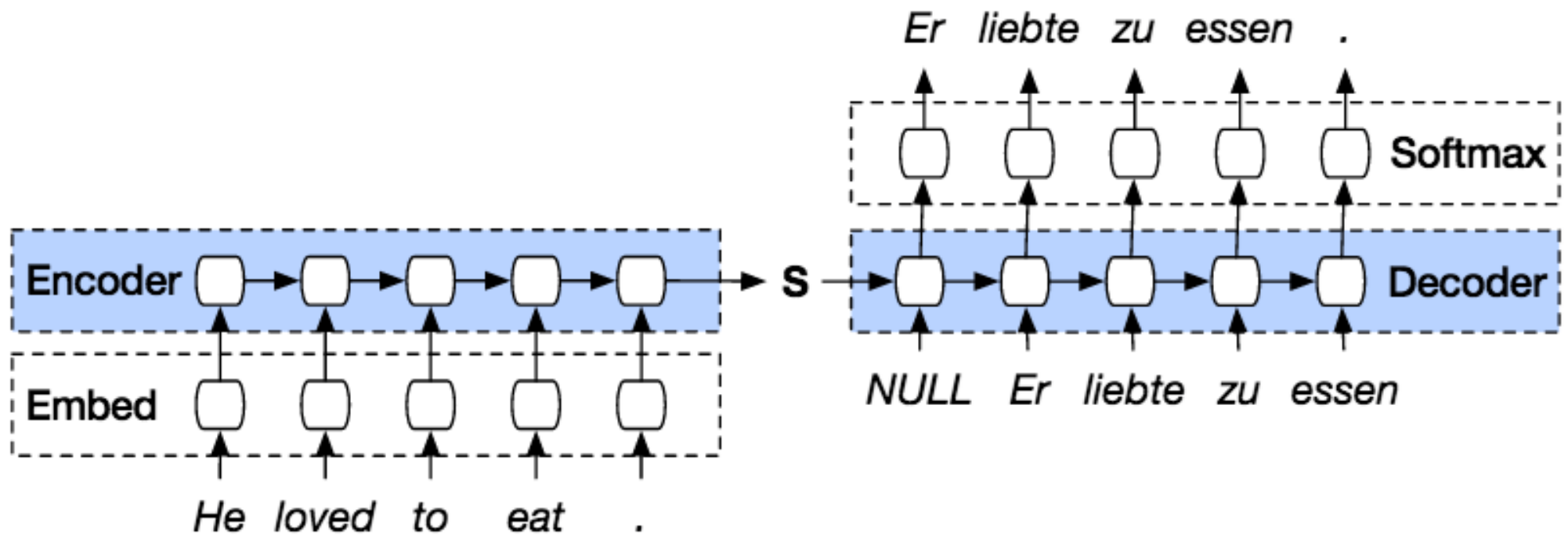




Recurrent Neural Networks



Seq2seq Model



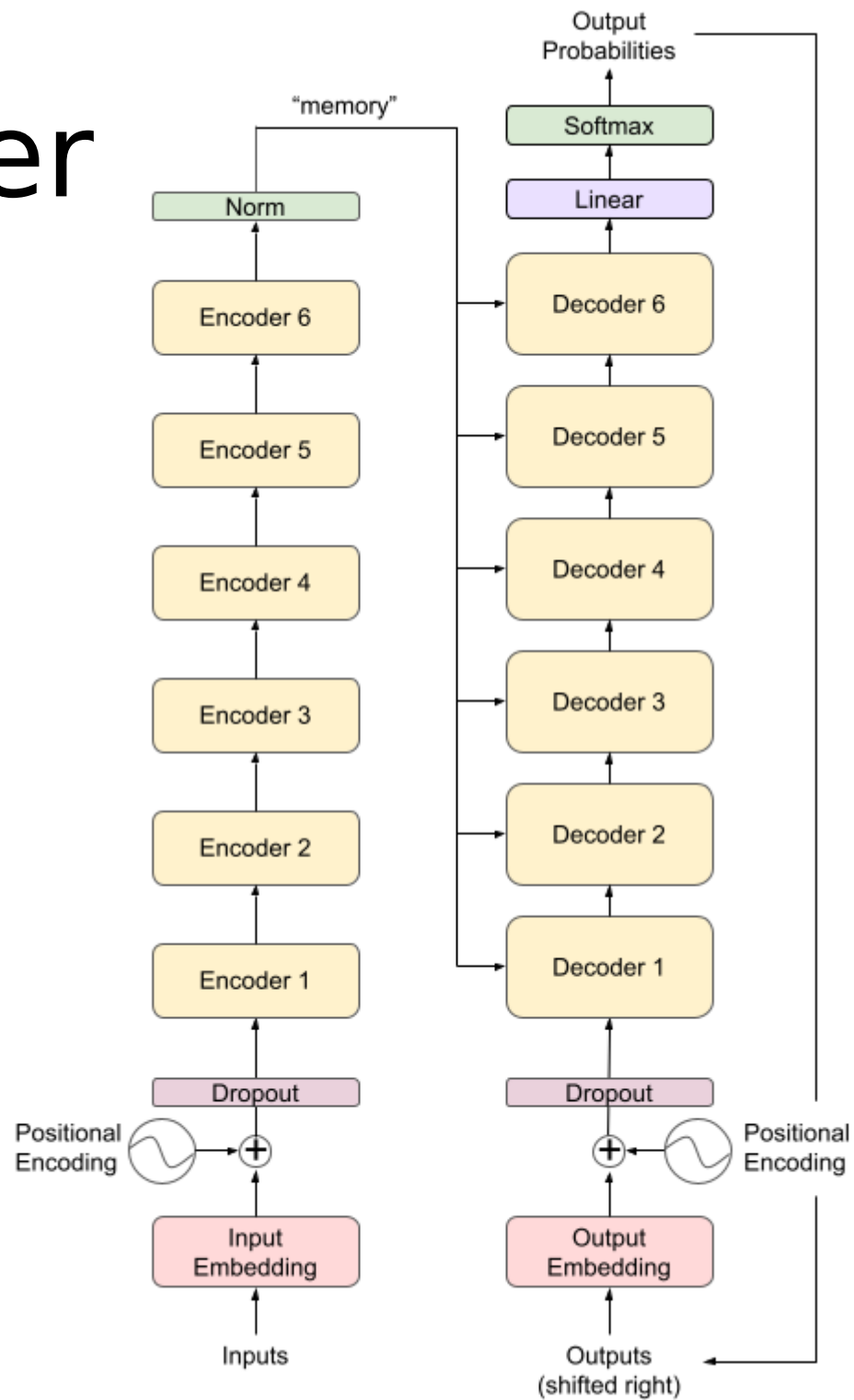
Drawbacks of RNNs

- Inherently sequential: must process one word at a time.
- Difficulty holding on to information that came much earlier in the input sequence.

Transformer Networks

- Encoder-decoder architecture.
- Encoder is pure feed-forward, not recurrent, so all words can be processed in parallel.
- Uses attention heads as subprocessors, analogous to kernels in a convolutional network but much more powerful.

Transformer



Training: Predict Next Word

Since he was out of milk, on the way home from work John →

stopped
dropped
bought
...

Since he was out of milk, on the way home from work John dropped →

by
into
off
...

Generation: Predict Next Word

Where do eagles live? → **Eagles**

Where do eagles live? Eagles → **are**

Where do eagles live? Eagles are → **found**

Where do eagles live? Eagles are found → **on**

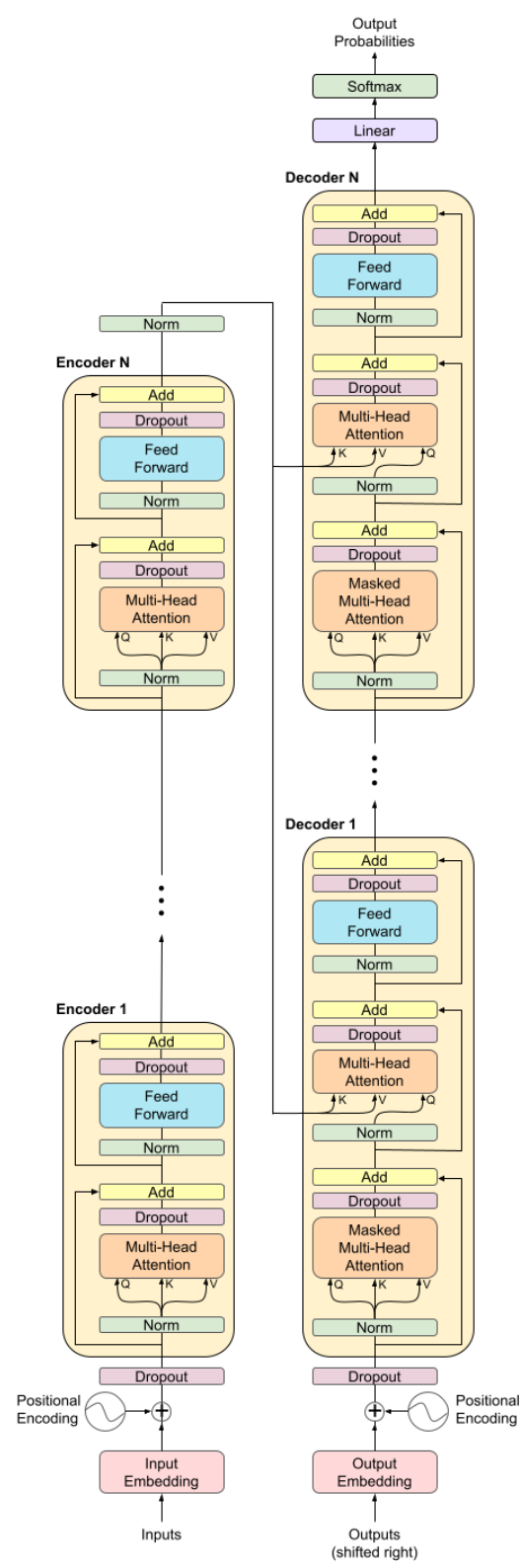
Where do eagles live? Eagles are found on → **every**

Where do eagles live? Eagles are found on every → **continent**

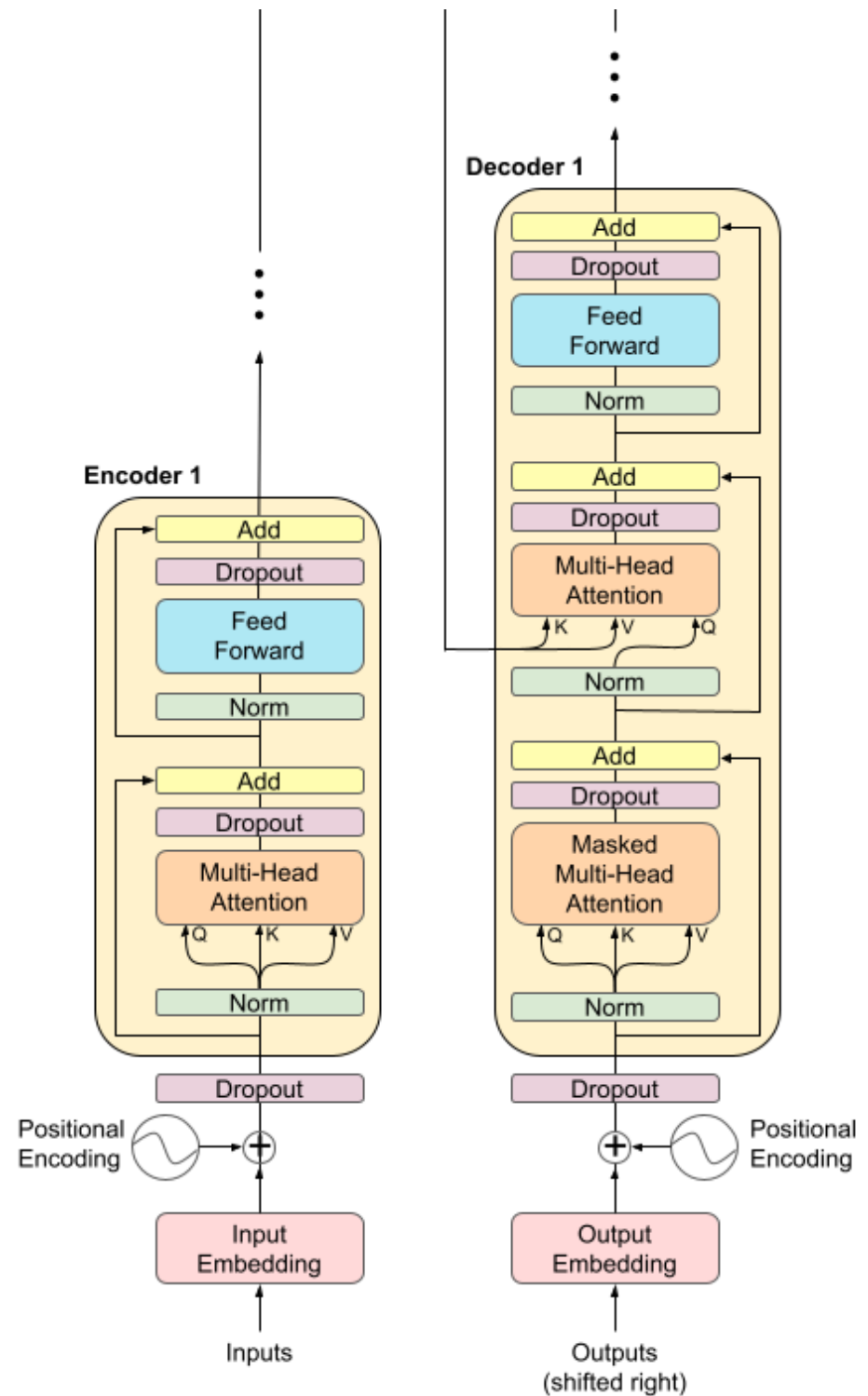
Where do eagles live? Eagles are found on every continent → **except**

Where do eagles live? Eagles are found on every continent except → **Antarctica**

Transformer

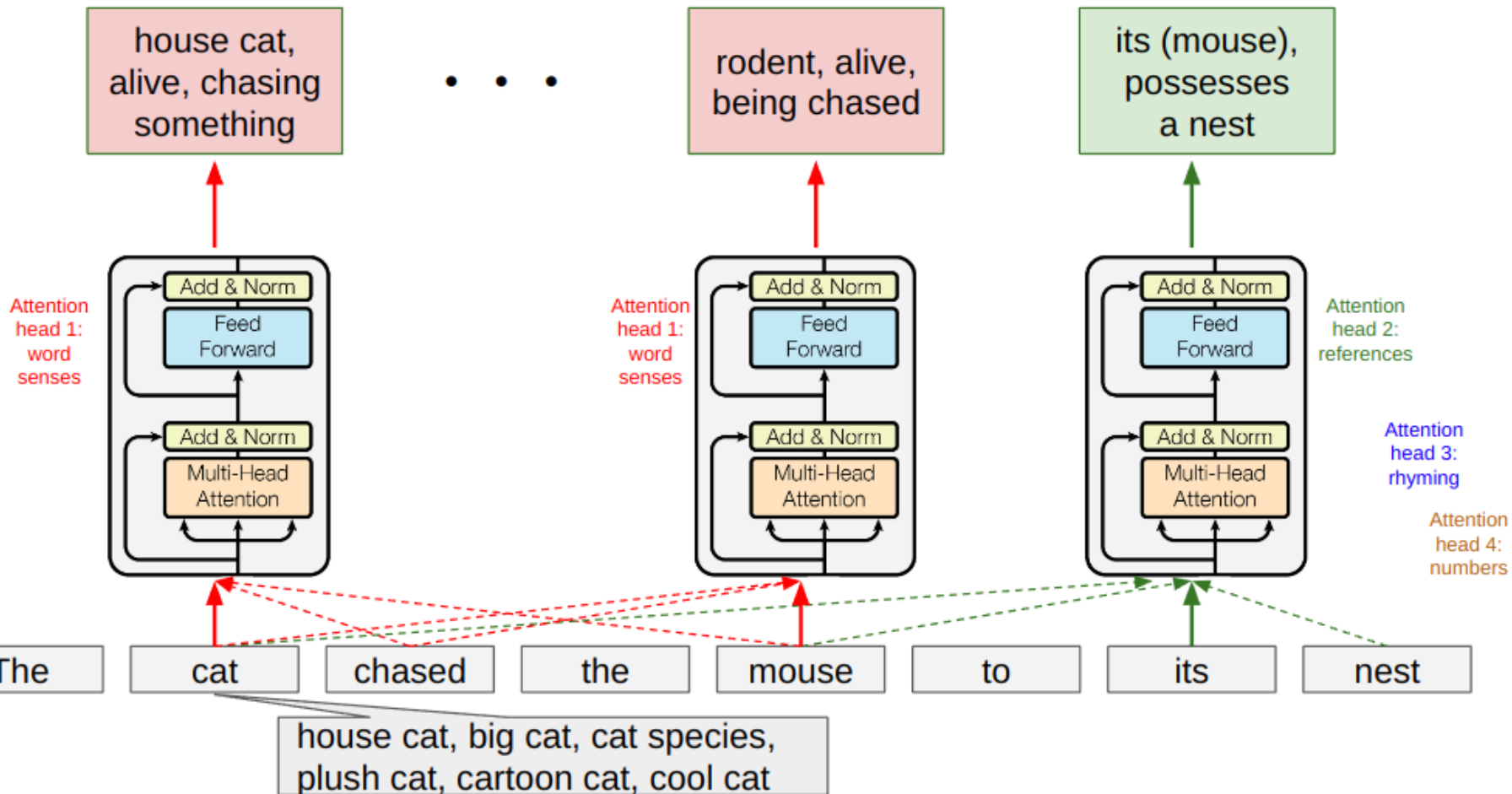


Transformer

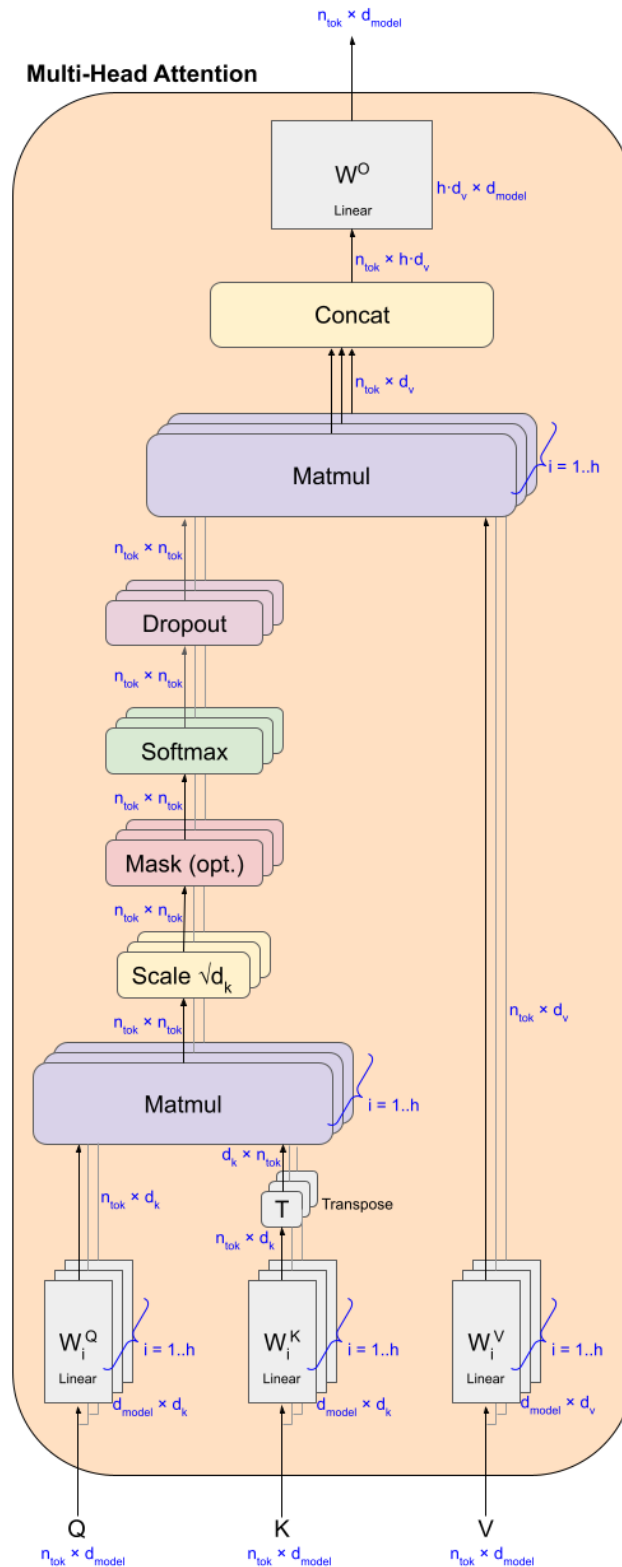


Parallel Encoding

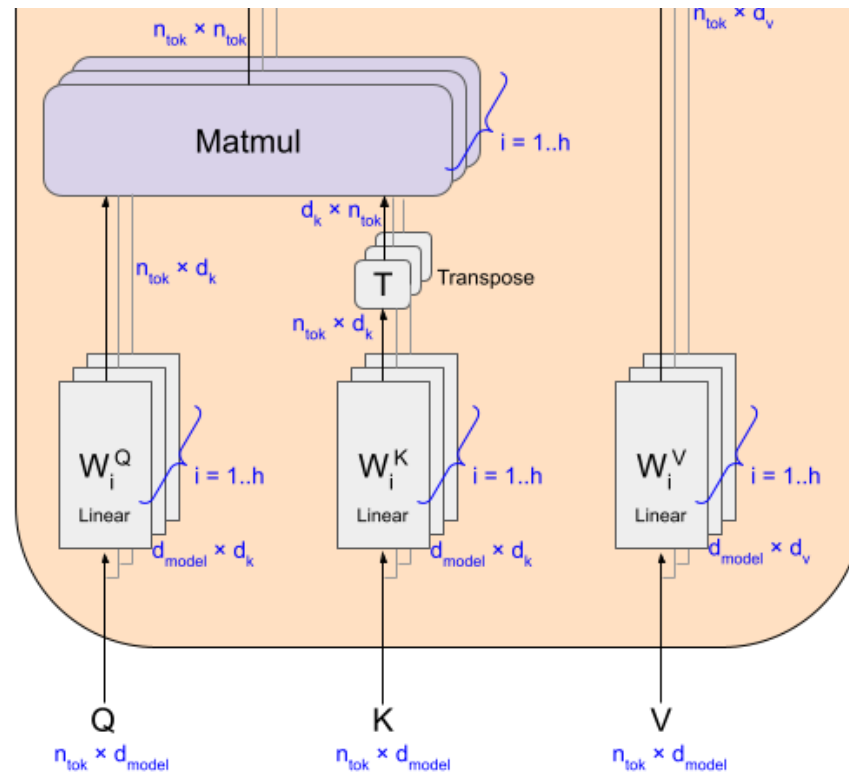
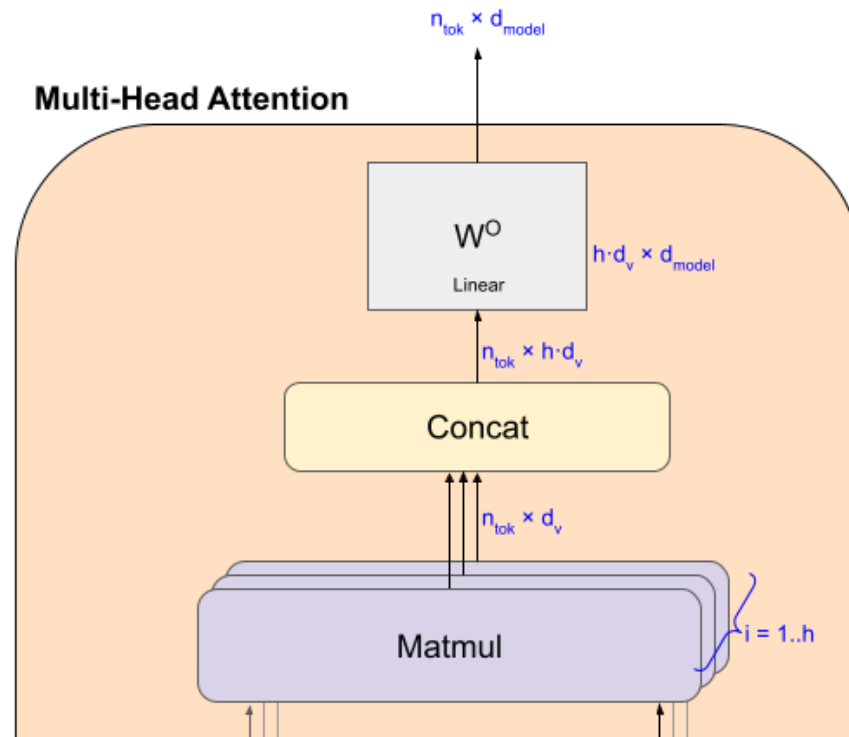
Level 1 Embeddings



Multi-Head Attention



Multi-Head Attention



Many Layers of Self-Attention Yields “Intelligence”

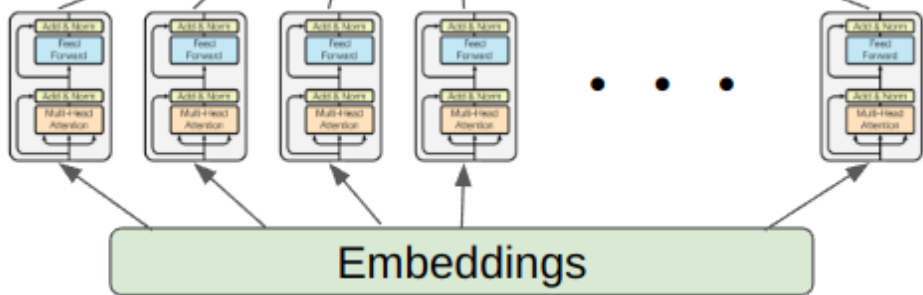
Layer 96



Layer 2



Layer 1



Embeddings

Embeddings

GPT-3 has 96 layers of self-attention, with 16 attention heads at each layer.

Every word is recoded and recombined with the other words 1536 times.

The model has 175 billion parameters (weights).

16 different attention heads in each layer

BERT

- Bidirectional Encoder Representations from Transformers (BERT)
- “Bidirectional” means the model looks at words both before and after the word of interest.
- BERT-large has 24 layers of 16 attention heads each: 340 million parameters. (GPT-3 has 175 billion.)

BERT

- Trained on Toronto book corpus (800M words) and English Wikipedia (2500M words).
- Two pre-training tasks:
 - Predict masked words (15% were masked)
 - Given two sentences, decide whether the second immediately follows the first in the training data, or is unrelated.

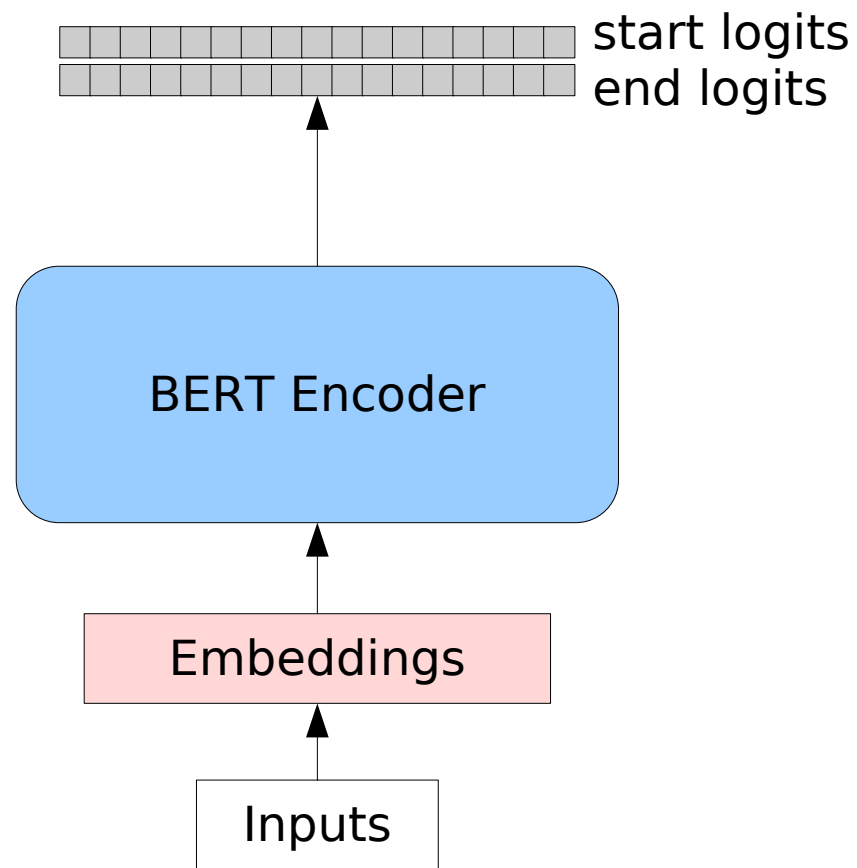
BERT

- BERT can be fine-tuned for a variety of tasks, which improves performance.
- Example tasks:
 - Extractive question answering
 - Sentiment analysis
 - Logical entailment
 - Conversational response generation

Extractive Question Answering

- Given a piece of text and a query, find a text excerpt that answers the query.
- Training set: SQUAD (Stanford Question Answering Dataset). Each item has:
 - A block of text
 - A question
 - Start and end positions of the answer
- Special circuitry tacked on: feed-forward network to assess start and end position probabilities for the extract.

BERT QNA



BERT-insight

Enter passage:

```
John and Mary went to a party. Mary bought a superamazing gift for the host.  
John brought his guitar.  
At the party, Mary gave the host a bottle of wine.  
John played songs after dinner.  
Fred was also at the party. He brought his dog with him.
```

Enter question:

```
What instrument did John play?
```

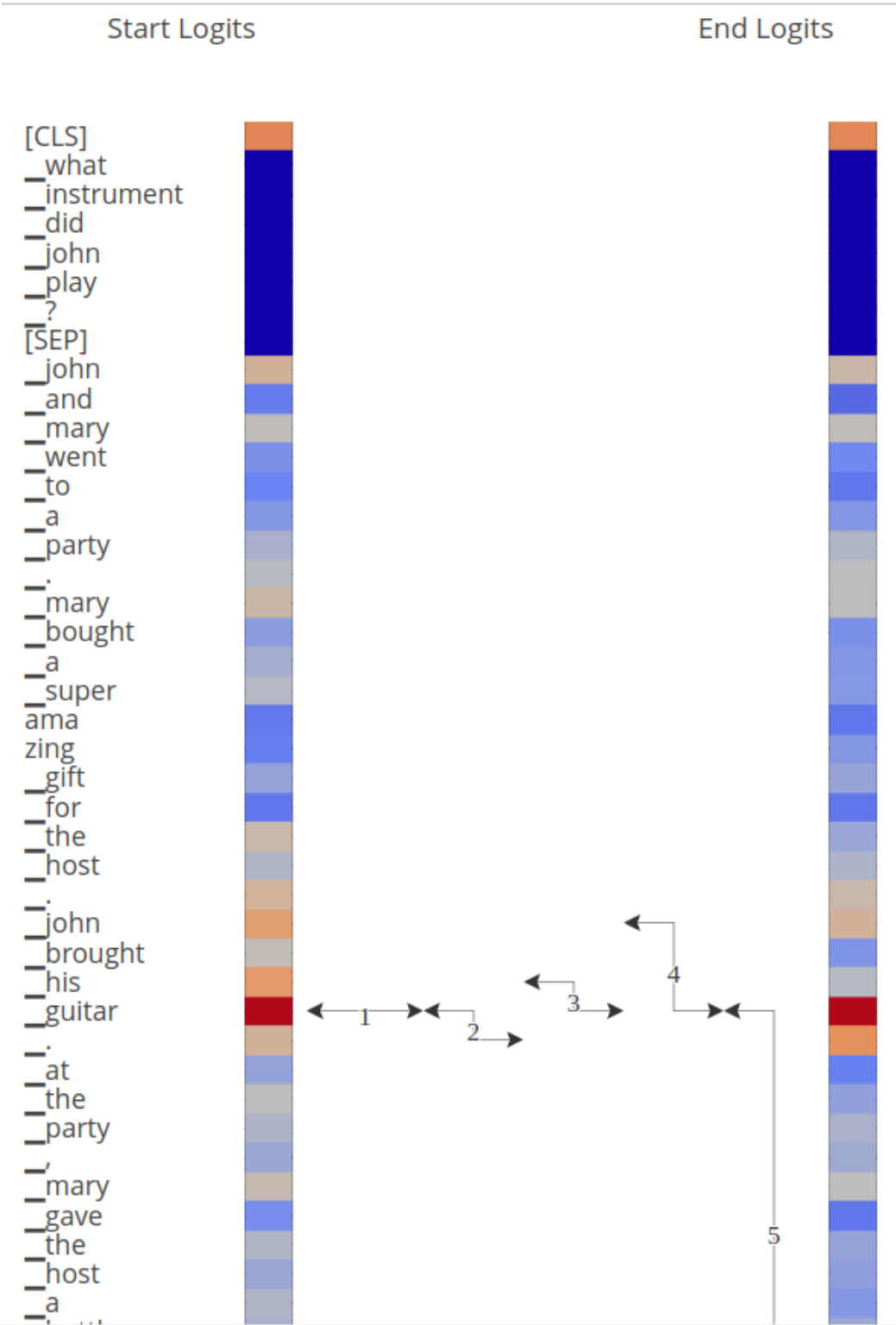
Submit

Answer:

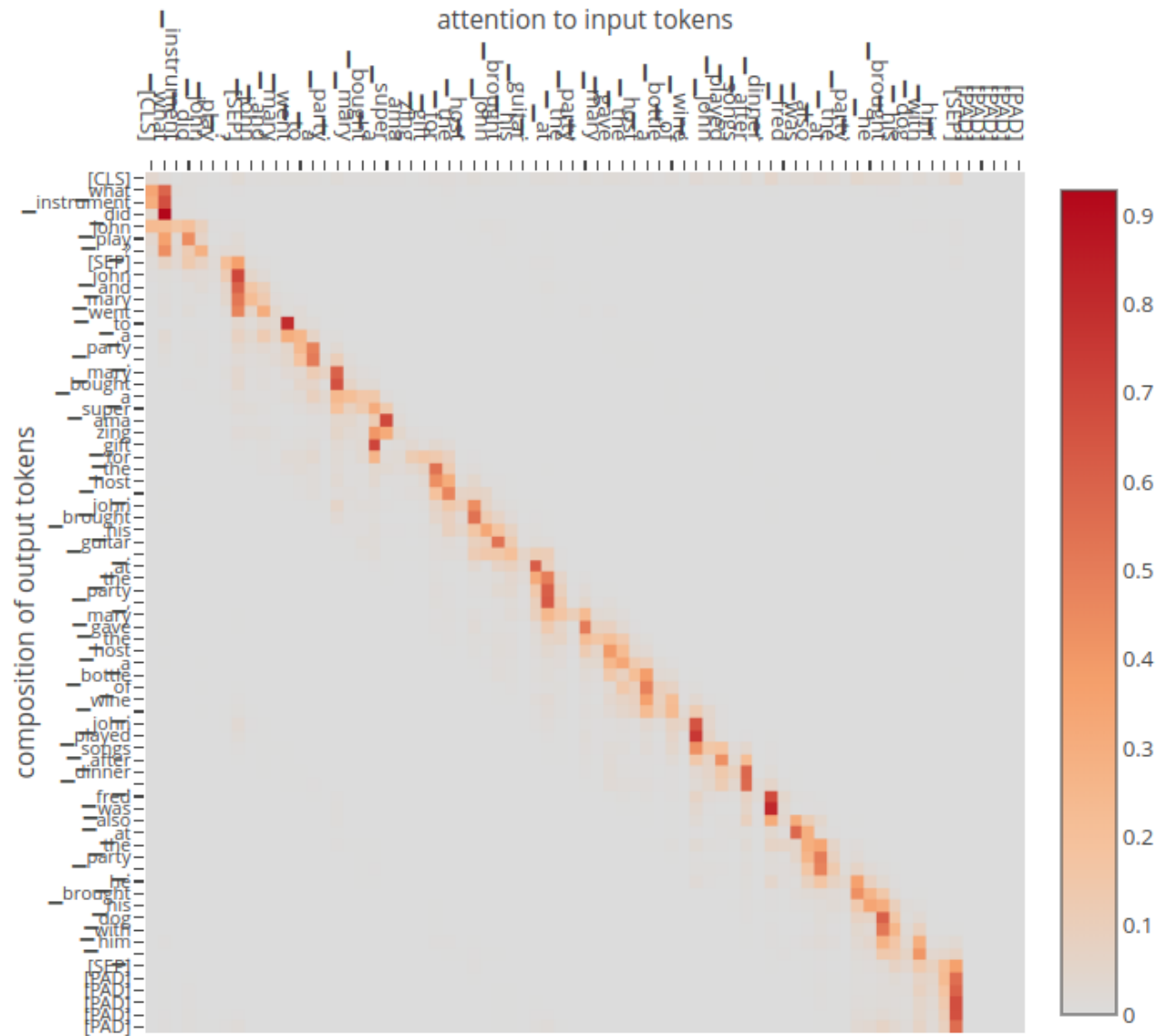
```
1: guitar | score: 20.853  
2: guitar. | score: 11.529  
3: his guitar | score: 10.345  
4: John brought his guitar | score: 9.920  
5: guitar. At the party, Mary gave the host a bottle of wine. John | score: 8.151
```

Answer:

- 1: guitar | score: 20.853
- 2: guitar. | score: 11.529
- 3: his guitar | score: 10.345
- 4: John brought his guitar | score
- 5: guitar. At the party, Mary gave t



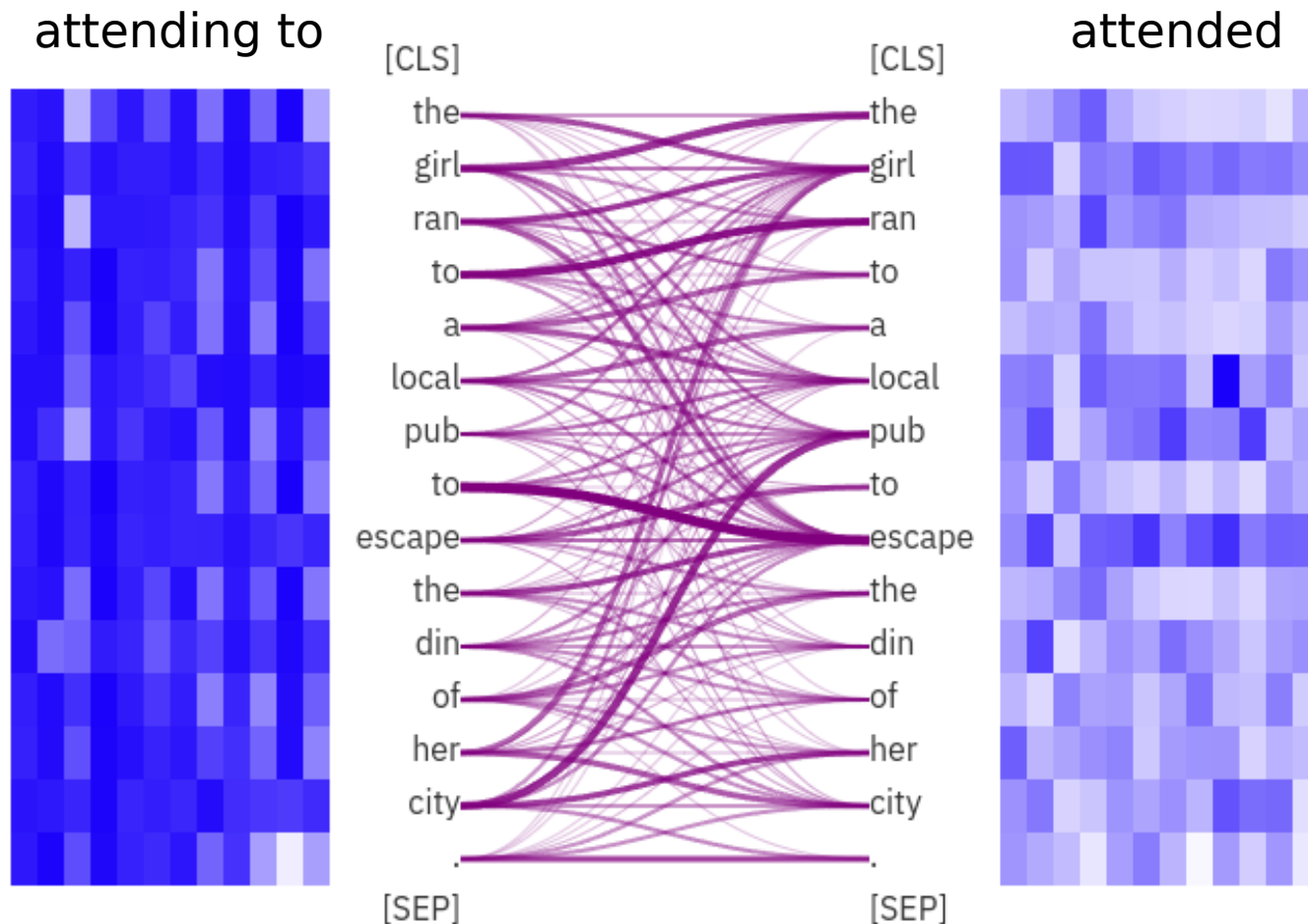
Attention Head Response Q·K



layer: bert/encoder/layer_0/attention/self/Softmax: head 1

Hugging Face Visualizer

<https://huggingface.co/bert-base-uncased>



ChatGPT

- Built on GPT 3.5 (now GPT-4)
- Pre-trained on predicting the next word in a sentence, and deciding if sentence₂ follows sentence₁.
- Fine-tuned by human raters to generate better quality responses (several thousand training examples). RLHF = Reinforcement Learning from Human Feedback.

LLMs and Robotics

- PaLM-E generates plans for robot actions, such as manipulating blocks on a table.
- PaLM-E can also take robot camera input and combine that with text to do visual question answering.
- We can use a chatbot to discuss the robot's knowledge about the world (driven by its current world map).

Key Vocabulary Terms

- Embedding
- Tokenizer
- Recurrent network
- Transformer network
- Attention head
- Large language model