AI models and interpretability

Complex Systems 530 - Marisa Eisenberg

Large language models

- ChatGPT (UM's version: http://umgpt.umich.edu)
- Llama
- Bard

How are large language models (LLMs) like Chat-GPT structured?

- GPT = Generative Pretrained Transformer
- Generative: generates the next word
- Pretrained: what it sounds like, i.e. trained ahead of time
- Transformer: the type of neural network model that many LLMs use
  - Built to recognize and work with sequences of inputs to generate the next thing (you can do this with other stuff too, e.g. images —> video, genetic sequences, etc.)

LLMs

- Transformer models - how they work
- What are they trained on, what are they trained to do
- What they do/don’t do

How is are large language models (LLMs) like Chat-GPT structured?

- LLMs are built to predict the next word in a sequence

"Once upon a ____"

seems like it should fill in the blank with "time" but not "armadillo".

$$P(time \mid once, upon, a)$$

$$P(word_n \mid word_1, word_2, \ldots, word_{n-1})$$

- Note the input sequence in a chat-GPT session is the whole conversation so far

https://mark-riedl.medium.com/a-very-gentle-introduction-to-large-language-models-without-the-hype-5f67941fa59e

Text example, temperature

- Chat-GPT 2
- Input: “Complex systems is a fascinating”
- Top 5 most probable outputs:

and	6.8%
concept	5.4%
subject	3.6%
topic	3.5%
way	3.2%

- But if you just take the most probably output every time?

Adapted from: https://writings.stephenwolfram.com/2023/02/what-is-chatgpt-doing-and-why-does-it-work/

You get sort of repetitive nonsense

```
In[10]:= NestList[StringJoin[#, model[#, "Decision"]] &, "Complex systems is a fascinating", 15]

Out[10]= {Complex systems is a fascinating, Complex systems is a fascinating and,
          Complex systems is a fascinating and complex,
          Complex systems is a fascinating and complex system,
          Complex systems is a fascinating and complex system.,
          Complex systems is a fascinating and complex system. It,
          Complex systems is a fascinating and complex system. It is,
          Complex systems is a fascinating and complex system. It is a,
          Complex systems is a fascinating and complex system. It is a system,
          Complex systems is a fascinating and complex system. It is a system that,
          Complex systems is a fascinating and complex system. It is a system that is,
          Complex systems is a fascinating and complex system. It is a system that is not,
          Complex systems is a fascinating and complex system. It is a system that is not only,
          Complex systems is a fascinating and complex system. It is a system that is not only a,
          Complex systems is a fascinating and complex system. It is a
            system that is not only a system, Complex systems is a fascinating
            and complex system. It is a system that is not only a system of}
```

You get sort of repetitive nonsense

```
In[11]:= StringReplace[Nest[StringJoin[#, model[#, "Decision"]] &,
           "Complex systems is a fascinating", 100], "\n" .. → " "]

Out[11]= Complex systems is a fascinating and complex system. It is a system that is not only a
system of complex systems, but also a system of complex systems. It is a system
that is not only a system of complex systems, but also a system of complex
systems. It is a system that is not only a system of complex systems, but also a
system of complex systems. It is a system that is not only a system of complex
systems, but also a system of complex systems. It is a system that is not only
```

You need to inject some randomness (temperature)

```
In[12]:= NestList[StringJoin[#, model[#, {"RandomSample", "Temperature" → .8}]] &,
          "Complex systems is a fascinating", 7]

Out[12]= {Complex systems is a fascinating,
          Complex systems is a fascinating new, Complex systems is a fascinating new area,
          Complex systems is a fascinating new area of,
          Complex systems is a fascinating new area of research,
          Complex systems is a fascinating new area of research.,
           Complex systems is a fascinating new area of research.
         , Complex systems is a fascinating new area of research.
```

```
In[29]:= NestList[StringJoin[#, model[#, {"RandomSample", "Temperature" → .8}]] &,
          "Complex systems is a fascinating", 7]

Out[29]= {Complex systems is a fascinating, Complex systems is a fascinating way,
          Complex systems is a fascinating way to, Complex systems is a fascinating way to think,
          Complex systems is a fascinating way to think about,
          Complex systems is a fascinating way to think about the,
          Complex systems is a fascinating way to think about the physical,
          Complex systems is a fascinating way to think about the physical and}
```

Although it can take you in some funny directions depending how you set the level of randomness

---

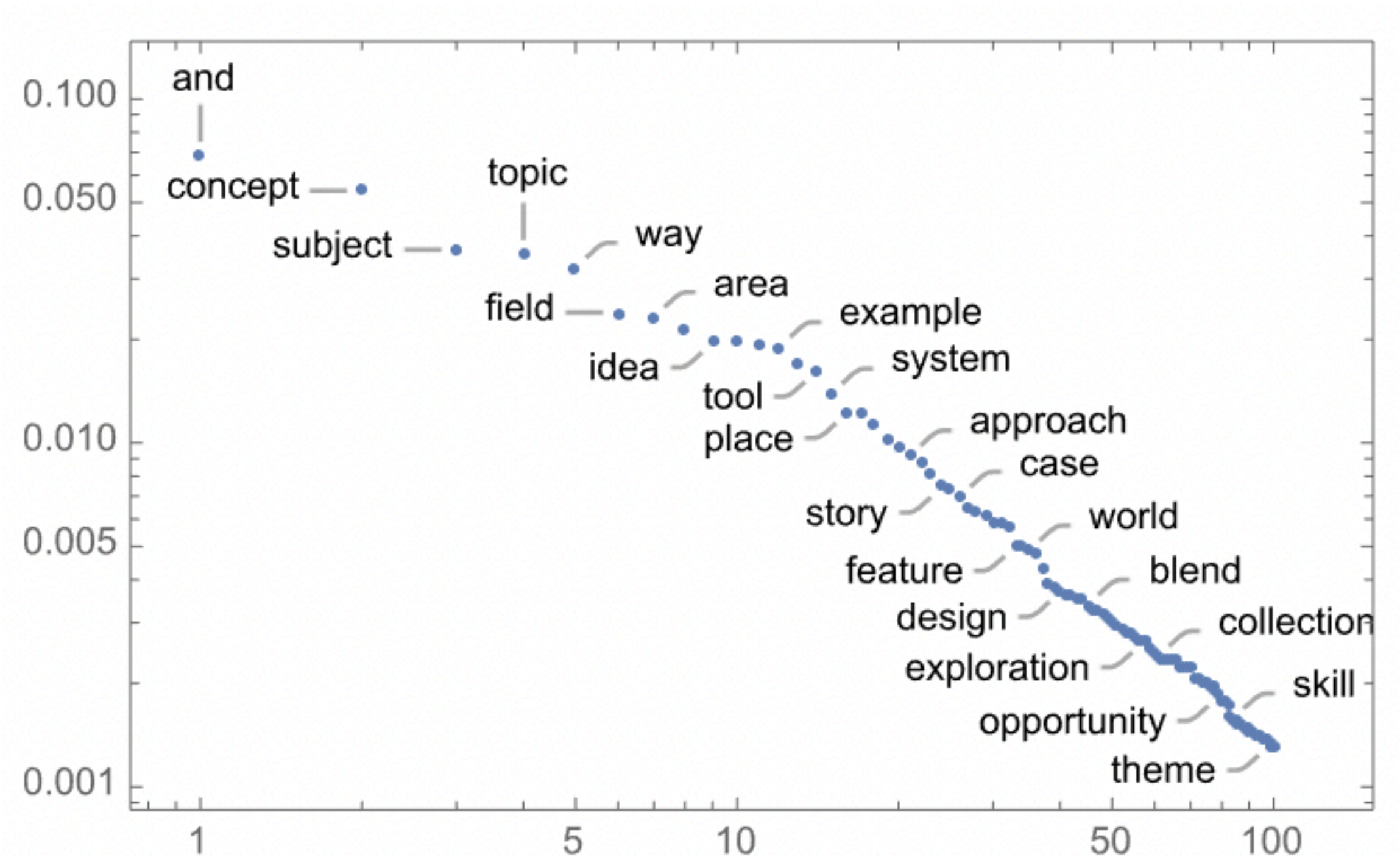
```
In[32]:= StringReplace[Nest[StringJoin[#, model[#, {"RandomSample", "Temperature" → .8}]] &,
           "Complex systems is a fascinating", 100], "\n" .. → " "]
```

```
Out[32]= Complex systems is a fascinating space, stimulating the imaginative
           imagination of the X-Men and the X-Men Unlimited universe. With our brilliant
           X-Men Unlimited Team of characters, we're bringing you the fastest, most
           creative and fun ways to experience the Marvel Universe. We've launched our
           first comic series with X-Men Unlimited, empowering the X-Men Unlimited
           franchise in a weird and wonderful new way. We've become a staple in
           comic book culture, and we think this is a great time to announce that
```

```
In[33]:= StringReplace[Nest[StringJoin[#, model[#, {"RandomSample", "Temperature" → .8}]] &,
           "Complex systems is a fascinating", 100], "\n" .. → " "]
```

```
Out[33]= Complex systems is a fascinating case study in the revolutionary development of
           automation. A few weeks ago, I was the guest of a techno-industrial symposium at TED.
           I was at the "tech-filled" stage and talked about the future of automation, entwined
           with the work of the American industrialists and the great techno-industrialists who
           wanted to make 'better' things. This was an event that was the culmination of a long
           process of exploration that was taking place in the past 10 years, starting with the
```

Distribution of possible next words (notice it gets power-law-y)

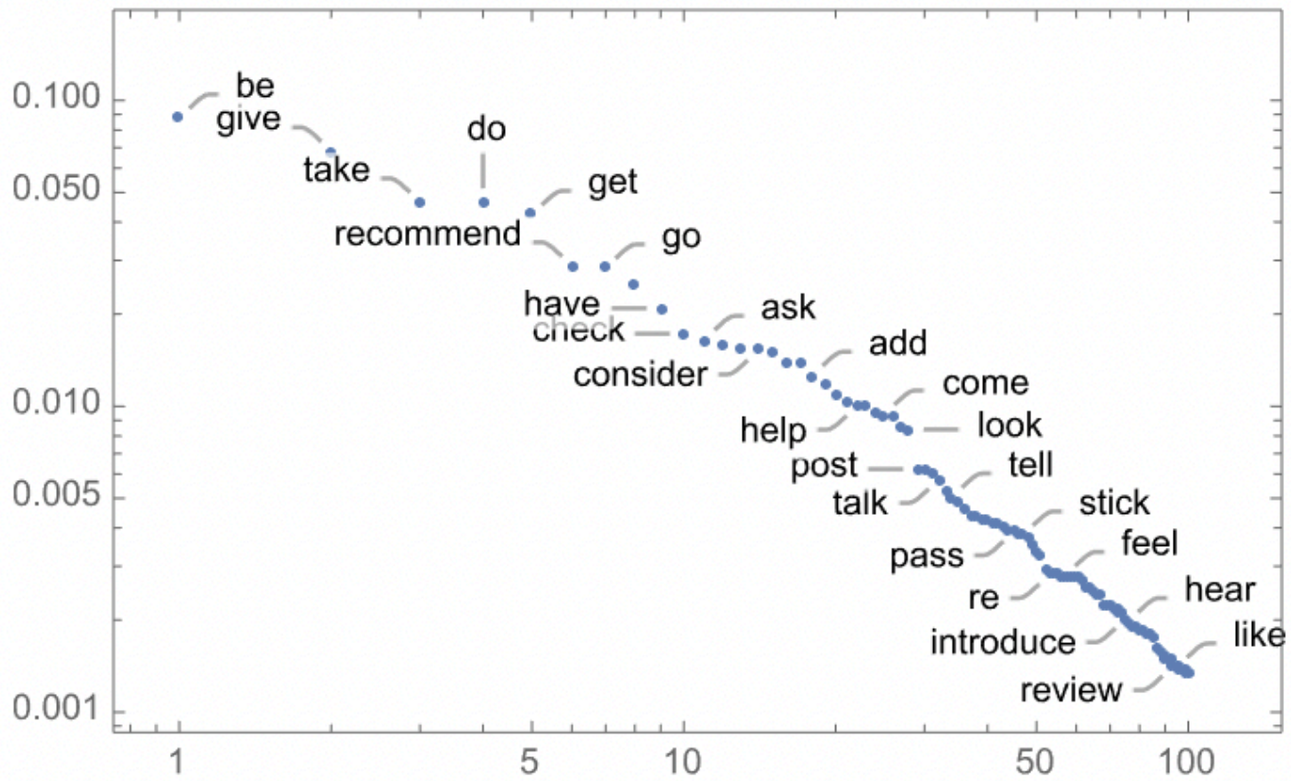


Adapted from: https://writings.stephenwolfram.com/2023/02/what-is-chatgpt-doing-and-why-does-it-work/

Distribution of possible next words (notice it gets power-law-y)

In[35]:=
```
ListLogLogPlot[
  Take[ReverseSort[model["I am very behind on lecture prep so I should definitely",
    "Probabilities"]], 100], Frame → True]
```

Out[35]=



Adapted from: https://writings.stephenwolfram.com/2023/02/what-is-chatgpt-doing-and-why-does-it-work/

How do you figure out word probabilities? Thinking about n-grams

- letter n-grams get better as we increase n

0	on gxeeetowmt tsifhy ah aufnsoc ior oia itlt bnc tu ih uls
1	ri io os ot timumumoi gymyestit ate bshe abol viowr wotybeat mecho
2	wore hi usinallistin hia ale warou pothe of premetra bect upo pr
3	qual musin was witherins wil por vie surgedygua was suchinguary outheydays theresist
4	stud made yello adenced through theirs from cent intous wherefo proteined screa
5	special average vocab consumer market prepara injury trade consa usually speci utility

- word n-grams too (random vs 2-gram examples)

of program excessive been by was research rate not here of of other is men
were against are show they the different the half the the in any were leaved

cat through shipping variety is made the aid emergency can the

cat for the book flip was generally decided to design of

cat at safety to contain the vicinity coupled between electric public

cat throughout in a confirmation procedure and two were difficult music

cat on the theory an already from a representation before a

https://writings.stephenwolfram.com/2023/02/what-is-chatgpt-doing-and-why-does-it-work/

n-grams and word probabilities

- Why not just use really long n-grams? Should generate longer sequences of words with the right overall probabilities

- But we don't have nearly enough data! Not enough English text in the world to do this

- Web: few hundred billion words
  Digitized books: ~hundred billion words.

https://writings.stephenwolfram.com/2023/02/what-is-chatgpt-doing-and-why-does-it-work/

- 40,000 common words —> number of 2-grams = 1.6 billion

- Number of possible 3-grams = 60 trillion

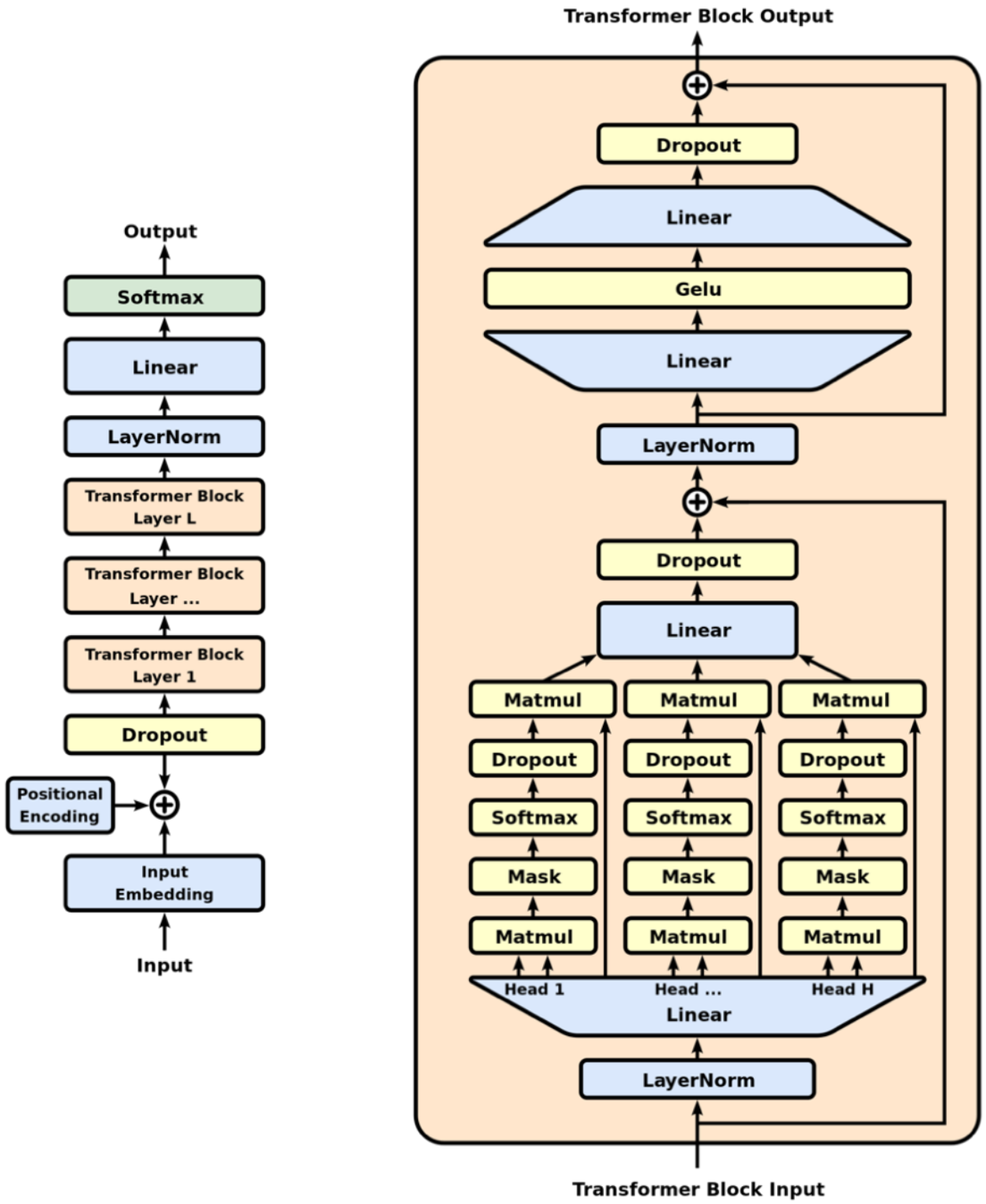
- 10 words? 20? More than would ever be possible to write down even if that’s all anyone ever did

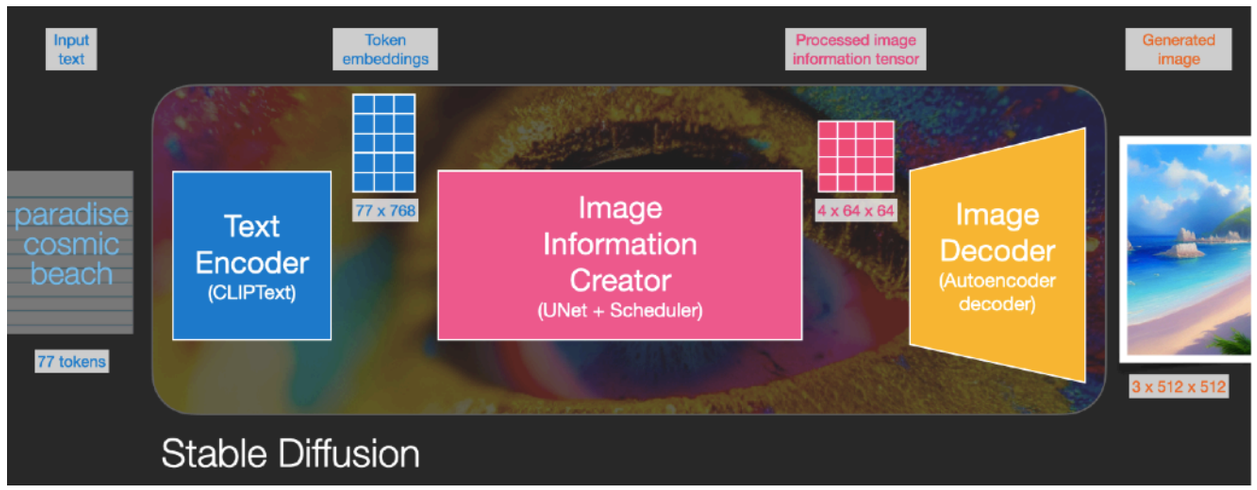
- Instead build a model that lets us estimate the sequence probabilities even if we’ve never seen that exact input string—this is what LLMs do

Okay, so how do we do it? Usually something like:

- **Tokenize**: get the data in a numerical form we can work with
- **Embed/encode** the inputs
- **Process** them (attention, transformers, etc) - various blocks of neural networks here
- **De-embed/decode**
- Spit out the next word you chose

Modular structure (blocks) in large AI models





By Vectorization: Mrmw - Own work based on: Full GPT architecture.png:, CC0, https://commons.wikimedia.org/w/index.php?curid=146645810

http://jalammar.github.io/illustrated-stable-diffusion/

Tokens

- There are ~50,000 commonly used words in the English language

- So we can take a piece of text and represent it as a sequence of numbers that encode which word

- But we do something slightly more efficient—tokens rather than words (this is part of why LLMs can sometimes produce weird made up words

Tokens

```
prompt <- "The best way to attract bees"
tokenizer$tokenize(prompt)
```

```
tf.Tensor([    1   450  1900   982   304 13978   367   267], shape=(8), dtype=int32)
```

```
prompt |> tokenizer$tokenize() |> tokenizer$detokenize()
```

```
tf.Tensor(b'The best way to attract bees', shape=(), dtype=string)
```

```
show_tokens(prompt)
```

1	450	1900	982	304	13978	367	267
""	"The"	"best"	"way"	"to"	"attract"	"be"	"es"

https://blogs.rstudio.com/ai/posts/2023-05-25-llama-tensorflow-keras/

Token examples

```
show_tokens("ing")
```

```
  1  2348
"" "ing"
```

```
show_tokens("working")
```

```
     1      1985
""  "working"
```

```
show_tokens("flexing")
```

```
   1   8525    292
"" "flex"  "ing"
```

```
show_tokens("wonking")
```

```
   1   2113   9292
""   "won" "king"
```

More frequent tokens get lower IDs

```
show_tokens(seq(50, len = 10))
```

```
 50  51  52  53  54  55  56  57  58  59
"/" "0" "1" "2" "3" "4" "5" "6" "7" "8"
```

```
show_tokens(seq(100, len = 10))
```

```
100 101 102 103 104 105 106 107 108 109
"a" "b" "c" "d" "e" "f" "g" "h" "i" "j"
```

```
show_tokens(seq(1000, len = 10))
```

```
  1000    1001    1002    1003    1004    1005    1006    1007    1008    1009
"ied"    "ER"  "stat"   "fig"    "me"   "von" "inter"  "roid"  "ater" "their"
```

```
show_tokens(seq(10000, len = 10))
```

```
  10000    10001    10002    10003    10004    10005    10006    10007
  "ång"  "citep"    "Ill"   "rank" "sender"   "beim"    "pak" "compat"
  10008    10009
"occurs"  "diese"
```

```
show_tokens(seq(20000, len = 10))
```

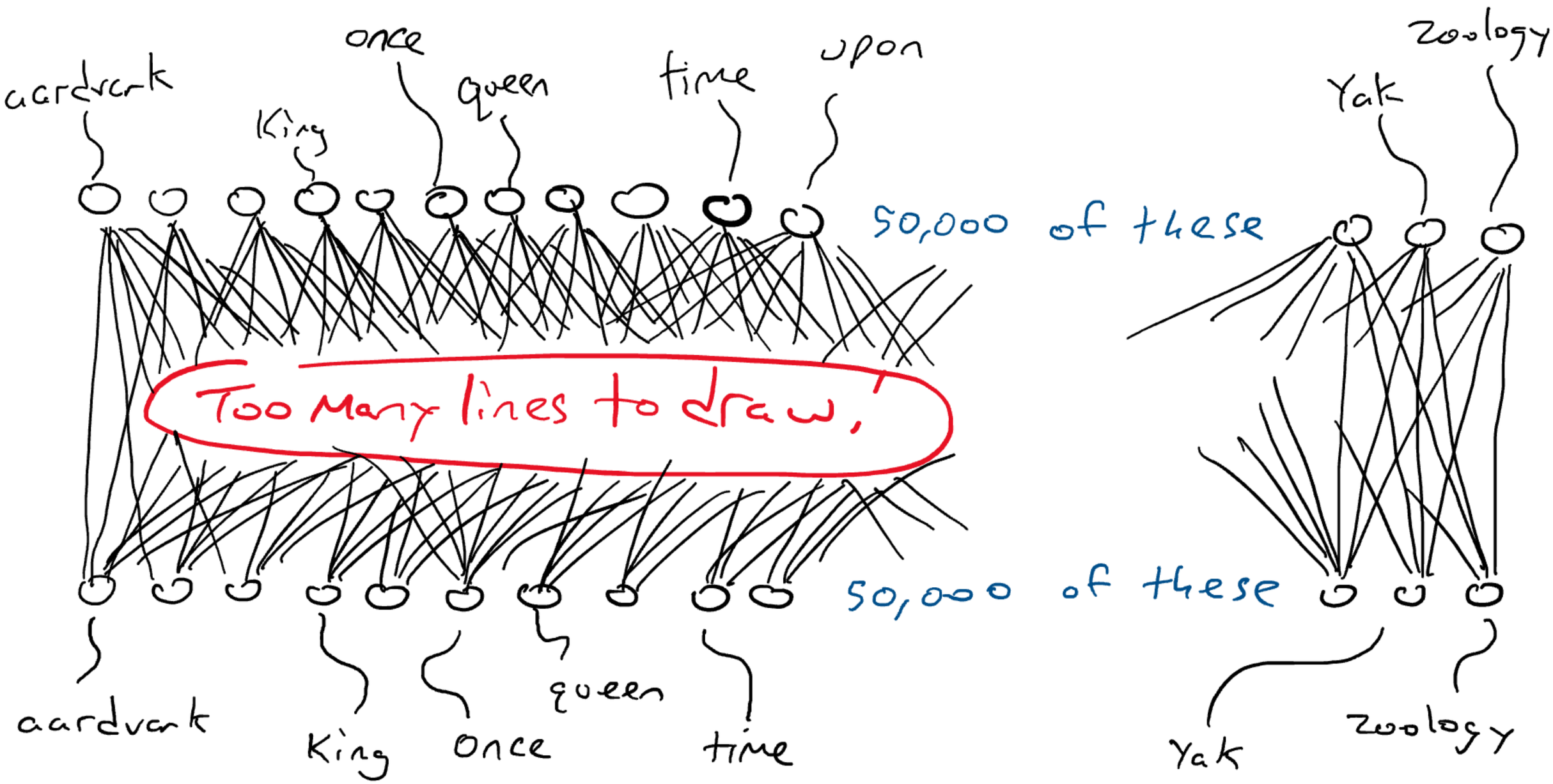
```
   20000     20001     20002     20003     20004     20005     20006     20007
 "admit" "Comment"     "стя"    "Vien"      "ці"  "permut"     "cgi"    "crít"
   20008     20009
"Console"    "ctic"
```

```
show_tokens(seq(to = as.integer(tokenizer$vocab_size()) - 1, len = 10))
```

```
31990 31991 31992 31993 31994 31995 31996 31997 31998 31999
  "ỏ"   "げ"   "べ"   "边"   "还"   "黃"   "왕"   "收"   "弘"   "给"
```

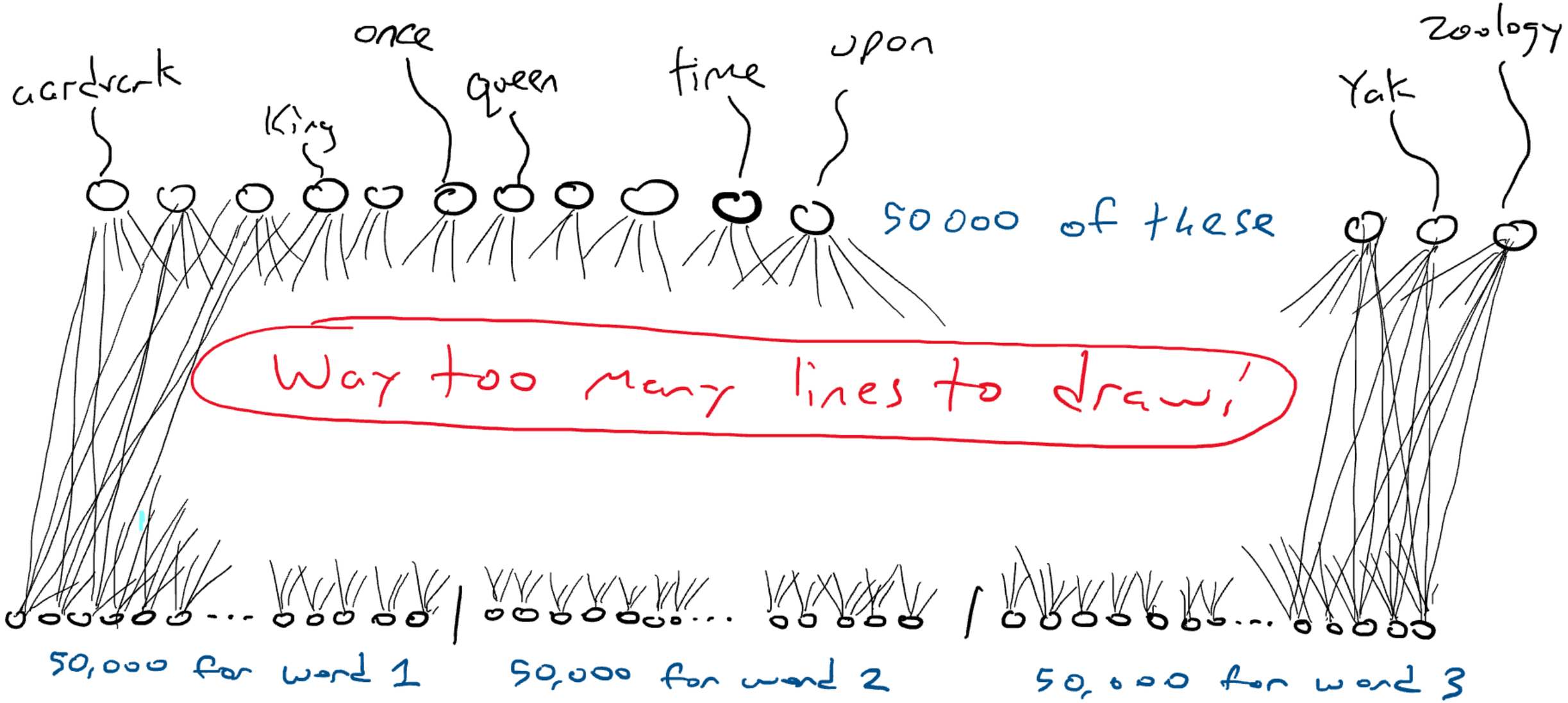
https://blogs.rstudio.com/ai/posts/2023-05-25-llama-tensorflow-keras/

Encoding/embedding: importance of dimension reduction



https://mark-riedl.medium.com/a-very-gentle-introduction-to-large-language-models-without-the-hype-5f67941fa59e

Even worse for longer input sequences



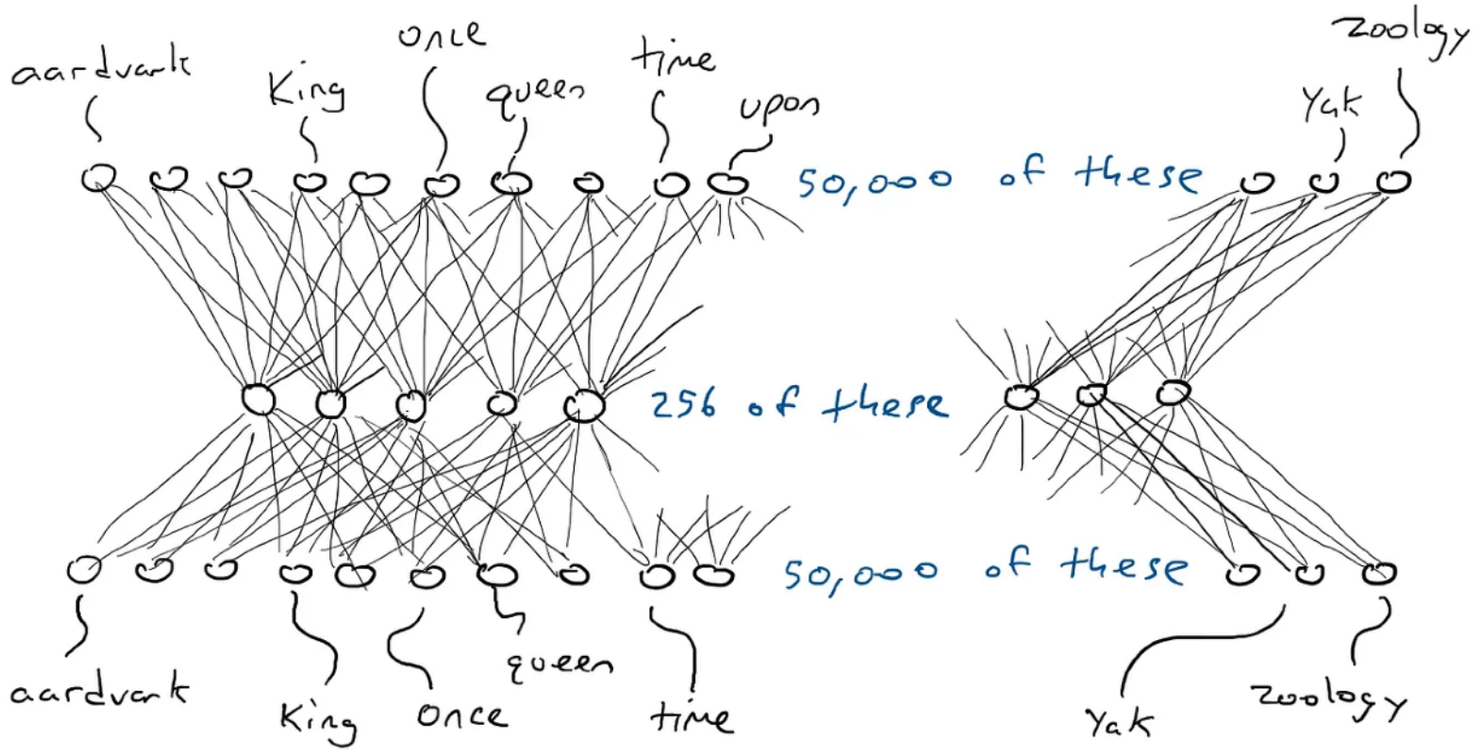
https://mark-riedl.medium.com/a-very-gentle-introduction-to-large-language-models-without-the-hype-5f67941fa59e

But many things functionally act the same in human language—we don't need such high dimension

- Anything that means royalty can probably lead to the same word (e.g. throne vs. chair/toilet/horse/etc)

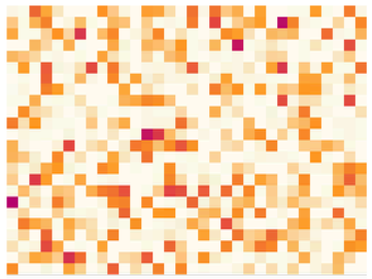
The king sat on the ___
The queen sat on the ___
The princess sat on the ___
The regent sat on the ___

- This is part of how LLMs learn to understand language!
- Forcing the model to reduce to a lower dimension means we learn common structures

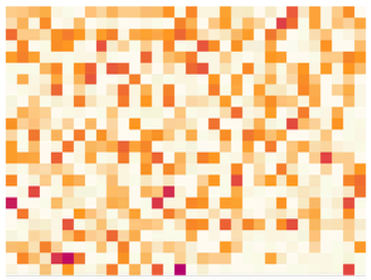


https://mark-riedl.medium.com/a-very-gentle-introduction-to-large-language-models-without-the-hype-5f67941fa59e

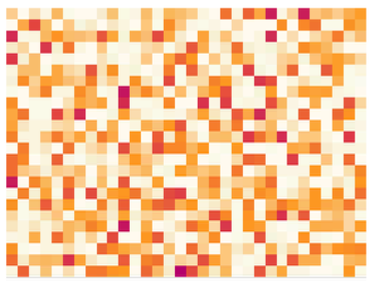
Embedding helps capture meaning



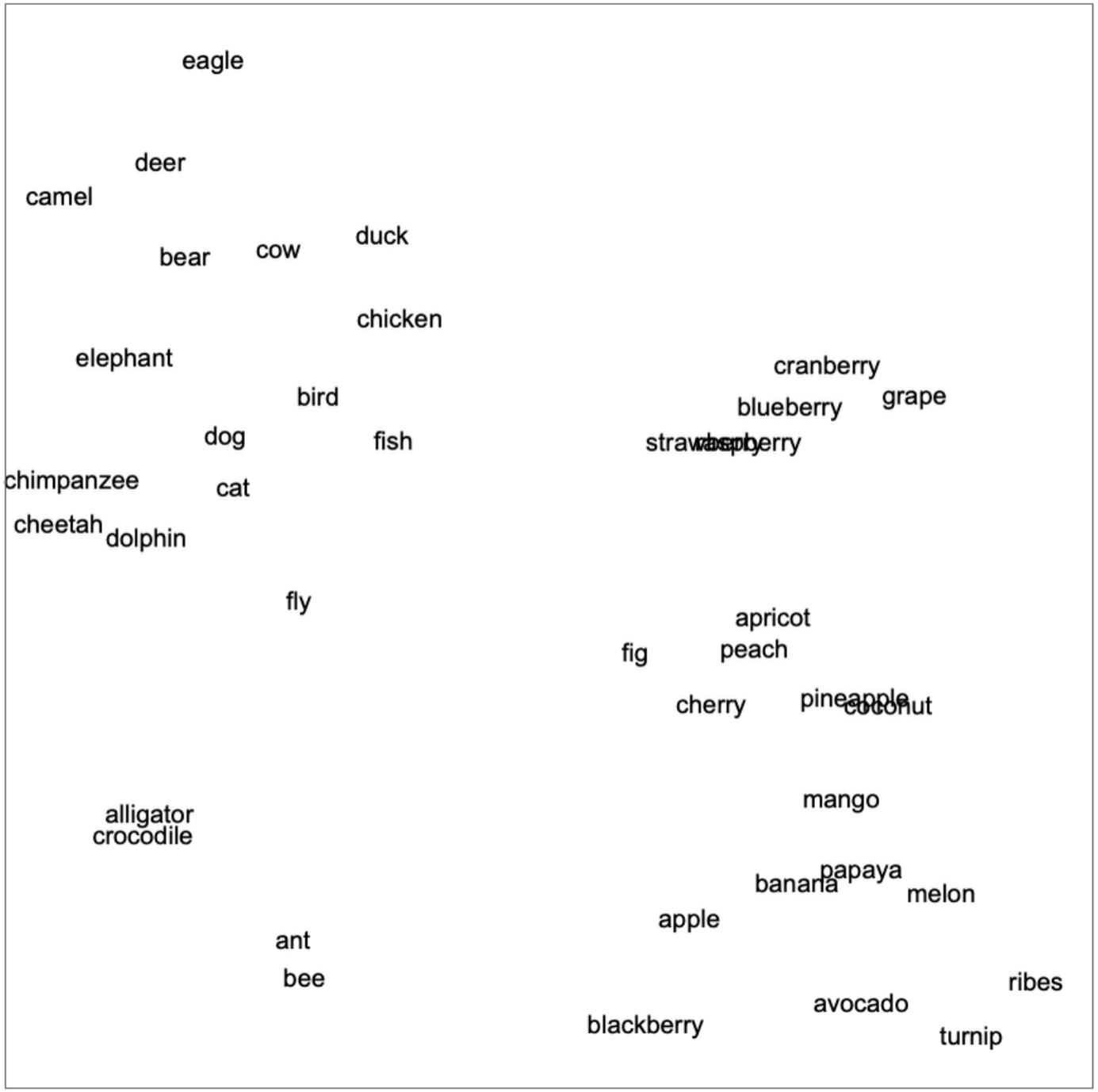
cat



dog



chair



https://writings.stephenwolfram.com/2023/02/what-is-chatgpt-doing-and-why-does-it-work/

Embedding

- How much did we crunch things down?

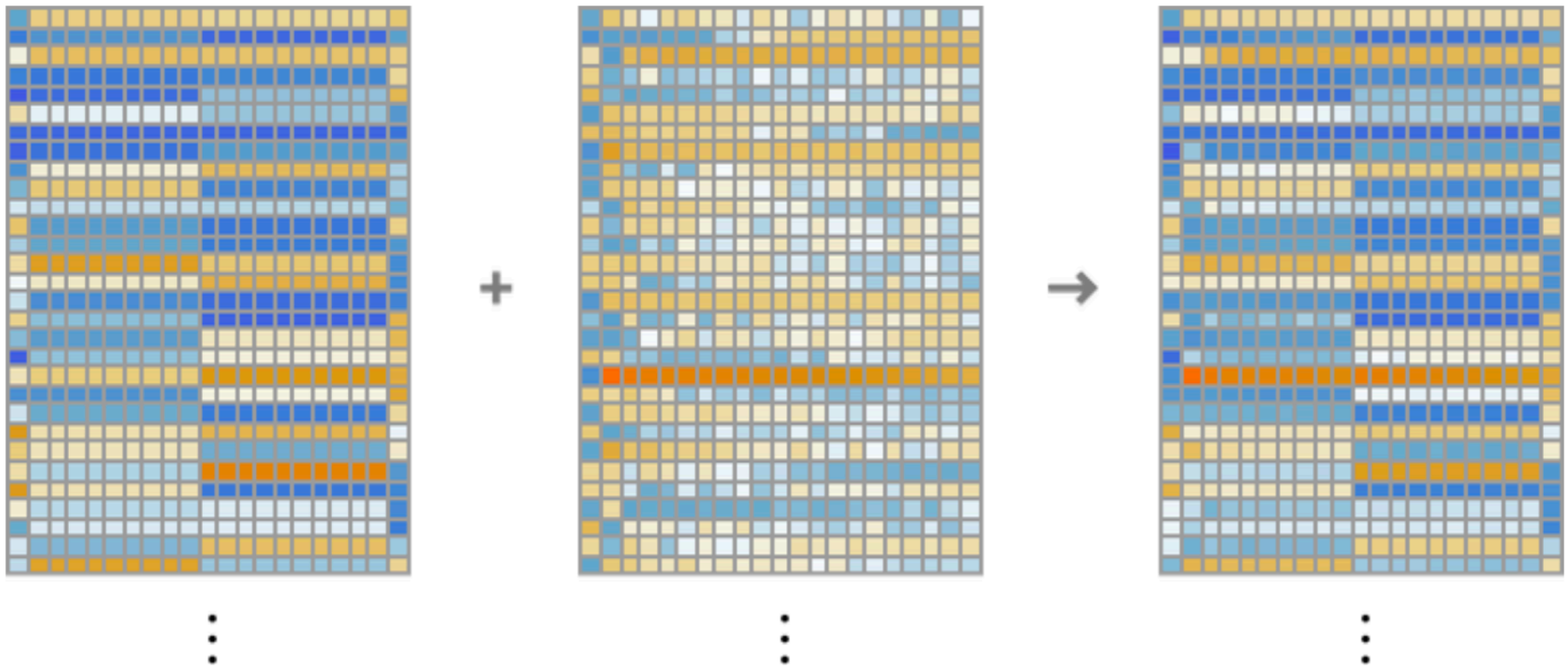
- 50K token options are converted (by a single-layer neural net) into an embedding vector of length 768 for GPT-2 and 12,288 for ChatGPT's GPT-3

- Knowing when to reduce dimension and when not to is part of the art that makes it not just a universal approximator

  - E.g. compare to digits example in http://neuralnetworksanddeeplearning.com/chap1.html (could do four output neurons but works much better with 10)

Positional encoding

- Keep track of both the what the words are and their position relative to each other
- Process those two pieces of information together
- Example: hello hello hello hello hello hello hello hello hello hello bye bye bye bye bye bye bye bye bye bye



https://writings.stephenwolfram.com/2023/02/what-is-chatgpt-doing-and-why-does-it-work/

Transformer blocks

- Process the embedded text input
- Made up of several sub-layers
  - Attention heads alternating with feedforward networks

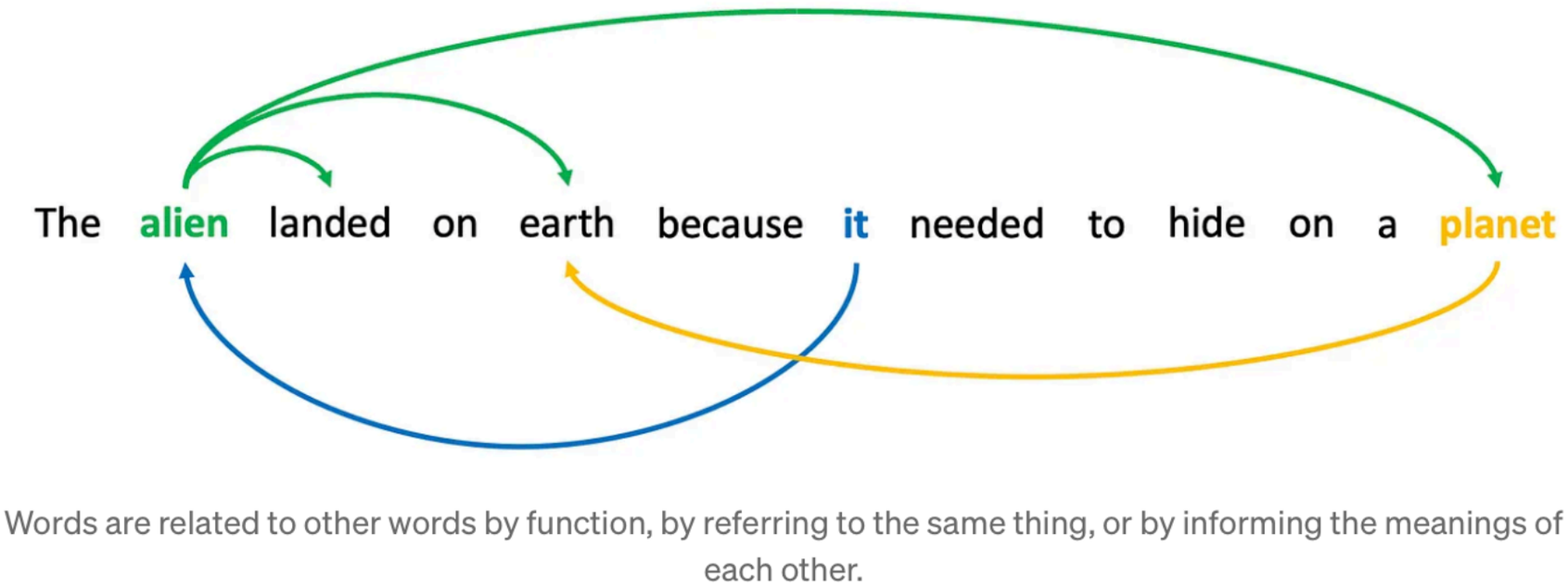
Transformer blocks

- Multi-head attention: Self-attention operates in multiple "attention heads" to capture different types of relationships between tokens.

- Feedforward neural networks: The output of the self-attention layer is passed through feedforward layers. These networks apply non-linear transformations to the token representations, allowing the model to capture complex patterns and relationships in the data.

https://www.ibm.com/topics/transformer-model

Attention heads



- Query, Key, Value, Residual
- Input a query and this acts as the key —> value is which words that word “attends” to (well, a weighted combination but yep)

https://mark-riedl.medium.com/a-very-gentle-introduction-to-large-language-models-without-the-hype-5f67941fa59e

LLMs

- In the end, they are almost sort of like really good search on billions of examples of data—weighting what we want next based on where we are

- We think of things like writing an essay as hard and creative, but in some sense they are just borrowing from us for ‘the hard part’ (the creativity, etc)

- Autoregressive (errors accumulate)

How do we train these?

- How do we train these? 175 billion parameters in chat GPT
  - Running out of data

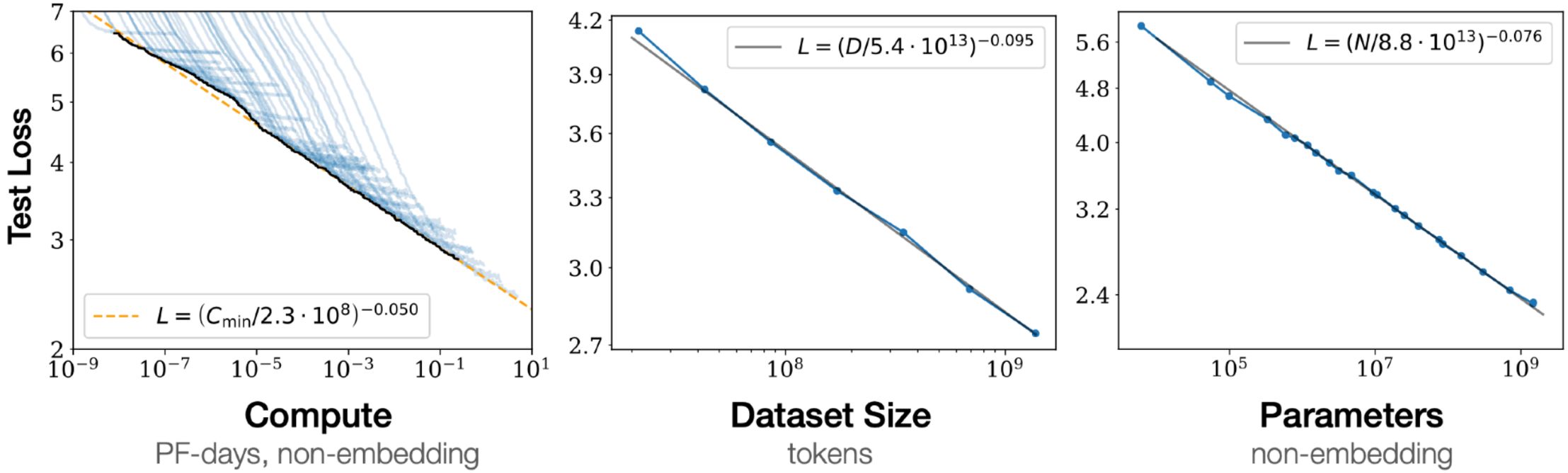


Figure 1 Language modeling performance improves smoothly as we increase the model size, datasetset size, and amount of compute[2] used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

https://arxiv.org/pdf/2001.08361.pdf

How do we train these?

- How do we train these? 175 billion parameters in chat GPT
  - Memorization
  - Identifiability