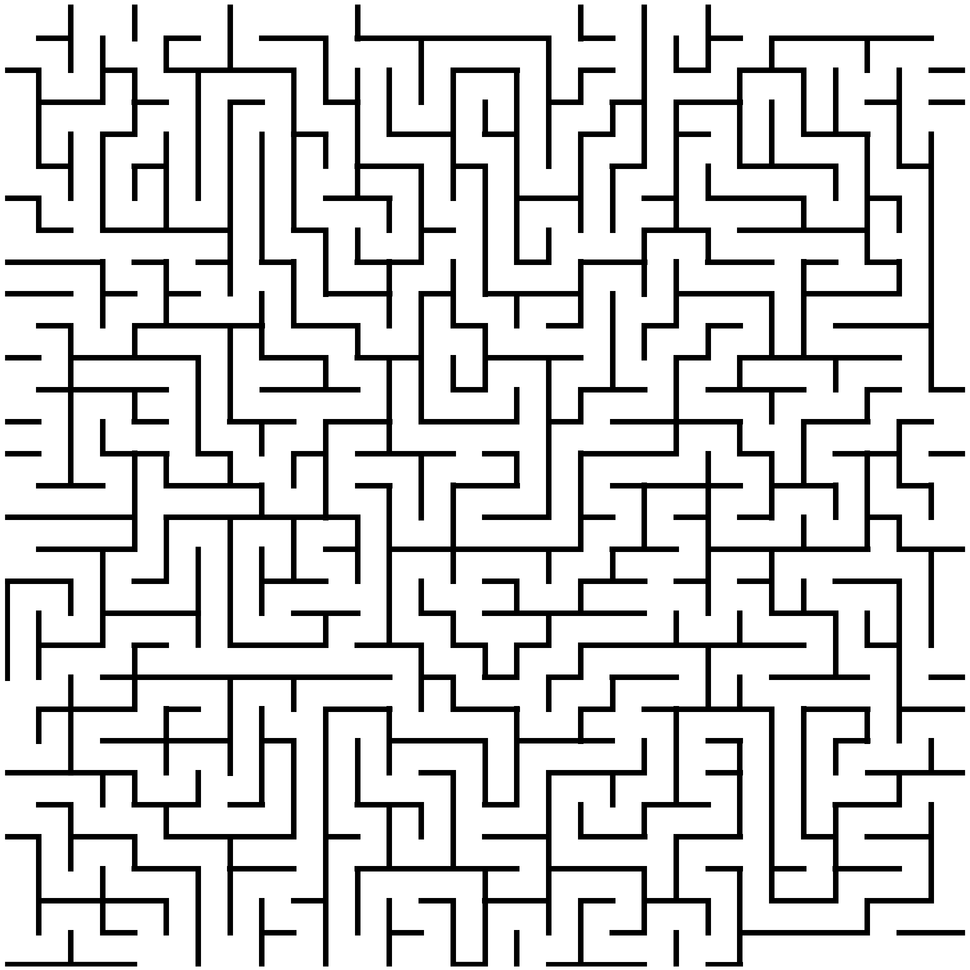
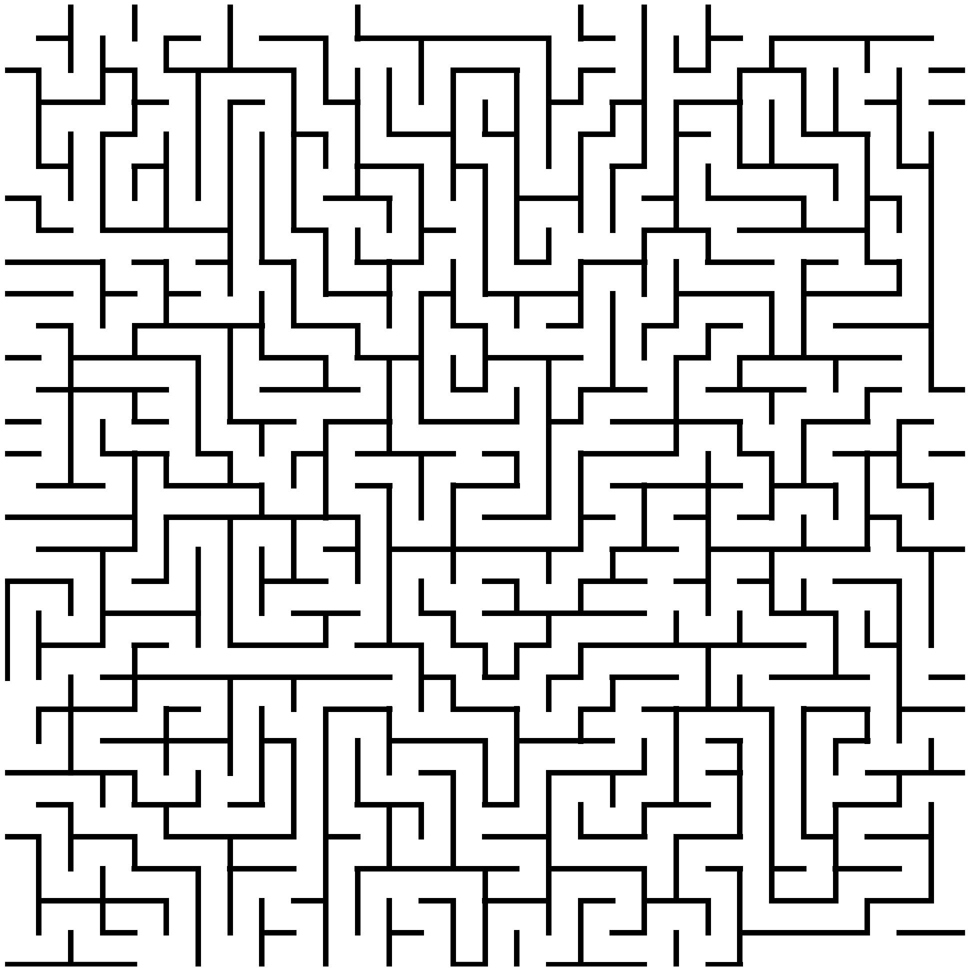
Generative AI

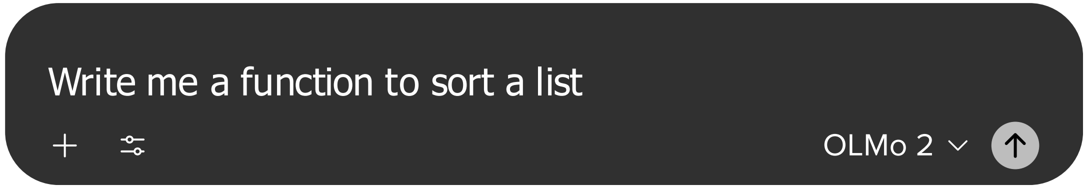
2025 / 10 / 15

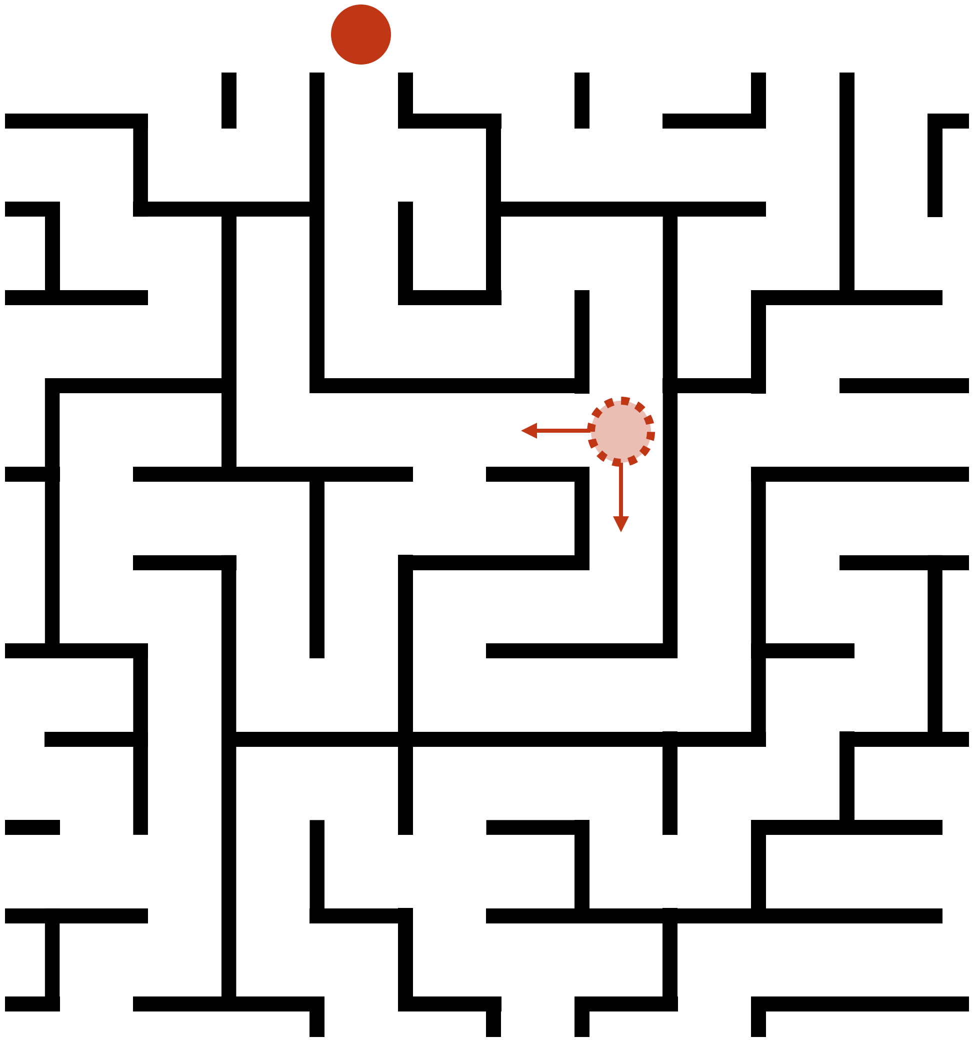


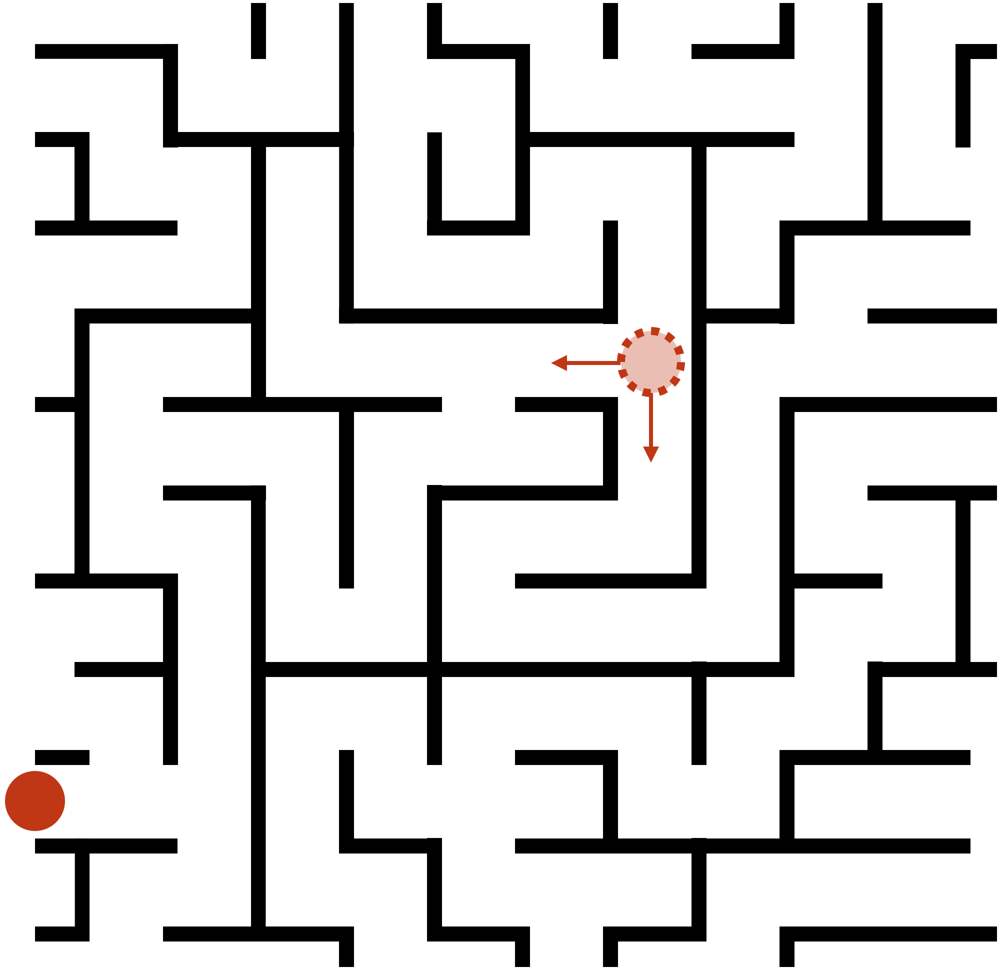
Prompting is a a **collaborative effort** between the LLM and you to **navigate through** the maze of **possibilities**.

Anything you add to the prompt will explicitly affect the output. Anything you don’t add to the prompt will implicitly affect the output.[1]




Write me a function to sort a list
OLMo 2



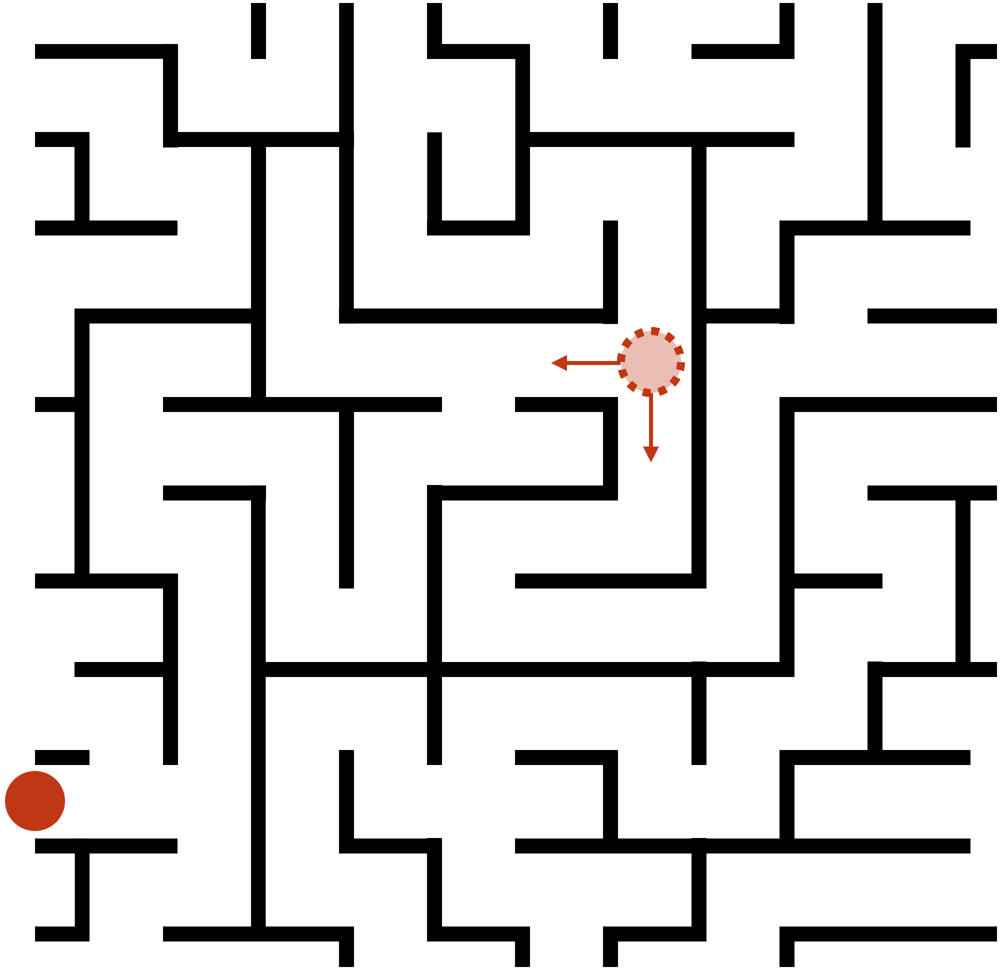


Quick Fire Prompting Tipps

Let the LLM explain its decisions.

Examples

- … What implicit decisions did you make?
- … What would I need to change in my prompt to achieve *XYZ.*



Quick Fire Prompting Tipps

Let the LLM explain its decisions.

Provide more **context**

- Your mental model / perspective / assumptions
- Previous solutions you tried
- Hard constraints and soft preferences

AGENDA

1. **What** is (non-)generative AI?
2. A bird's eye view of **generative sequence modeling**.
3. A bird‘s eye view of **diffusion modeling**.
4. A bird's eye view of the **transformer** architecture.

What is

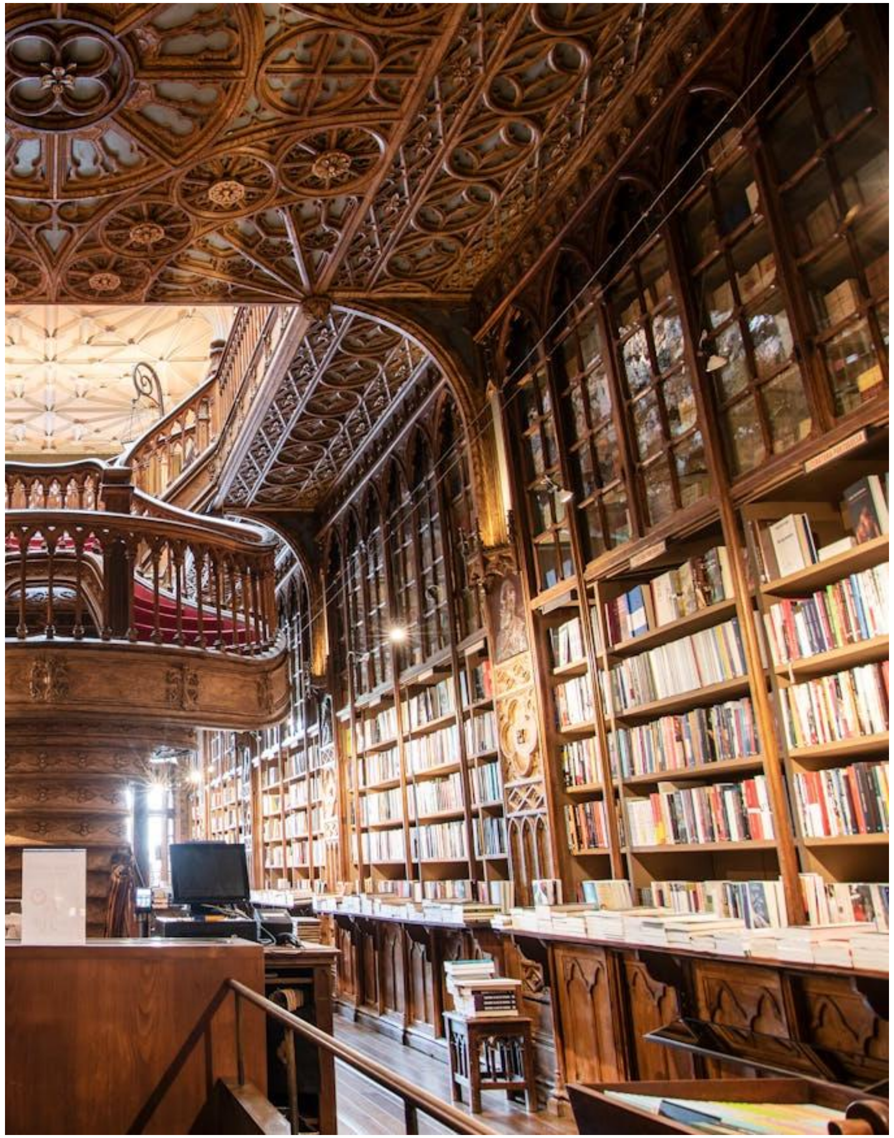
Generative Aritificial Intelligence

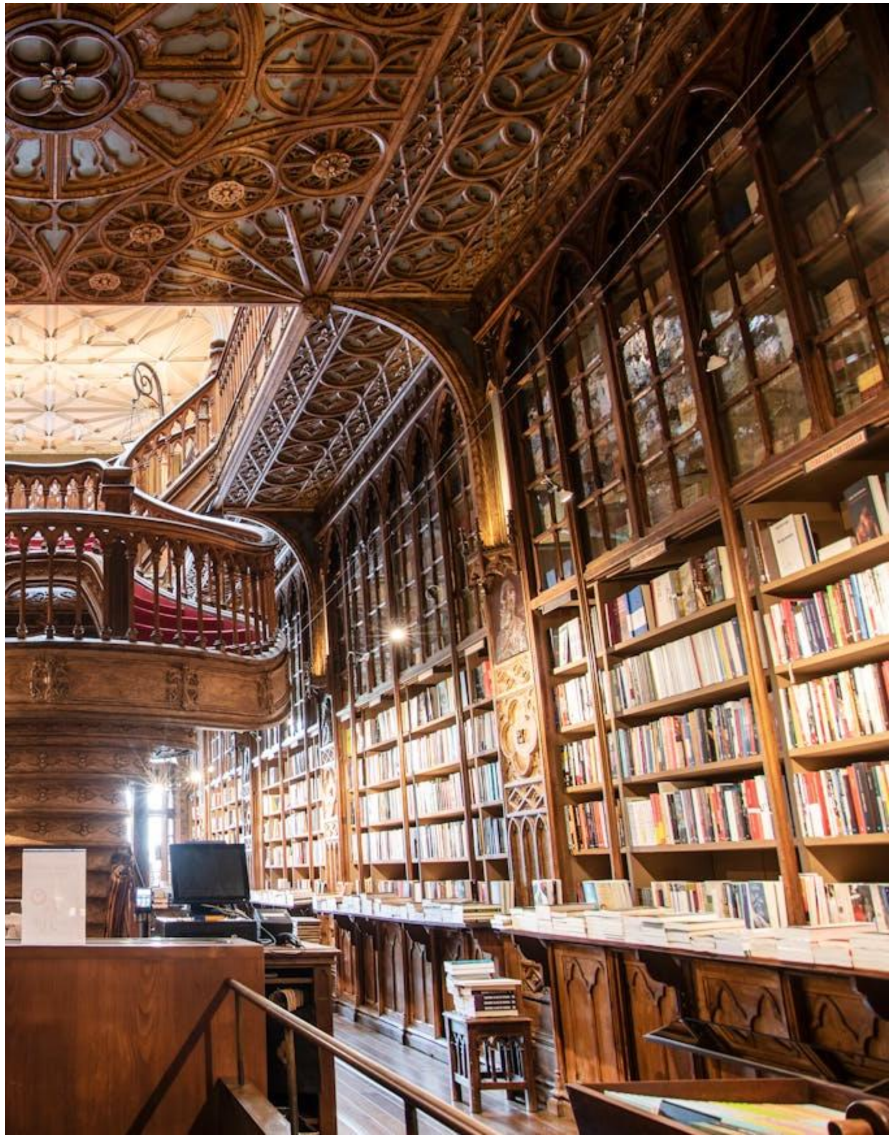
Artificial Intelligence

Symbolic AI

Sub**symbolic AI**

Symbolic AI





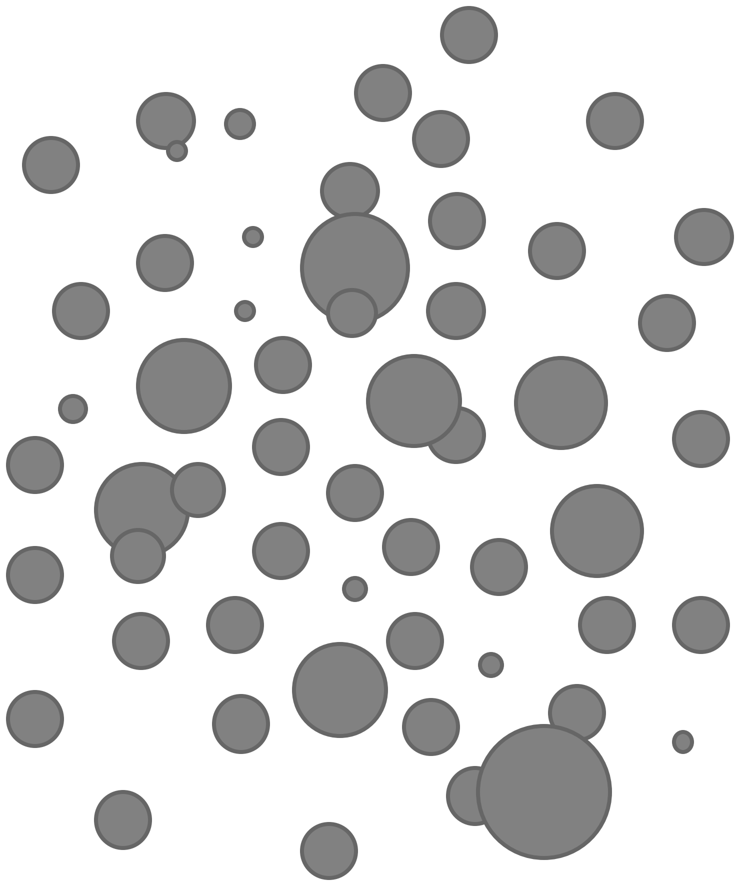
"A book is either available or a member has borrowed it."

$\forall x\, (Book(x) \rightarrow (Available(x) \lor \exists y\, (Member(y) \land Borrowed(y, x))))$

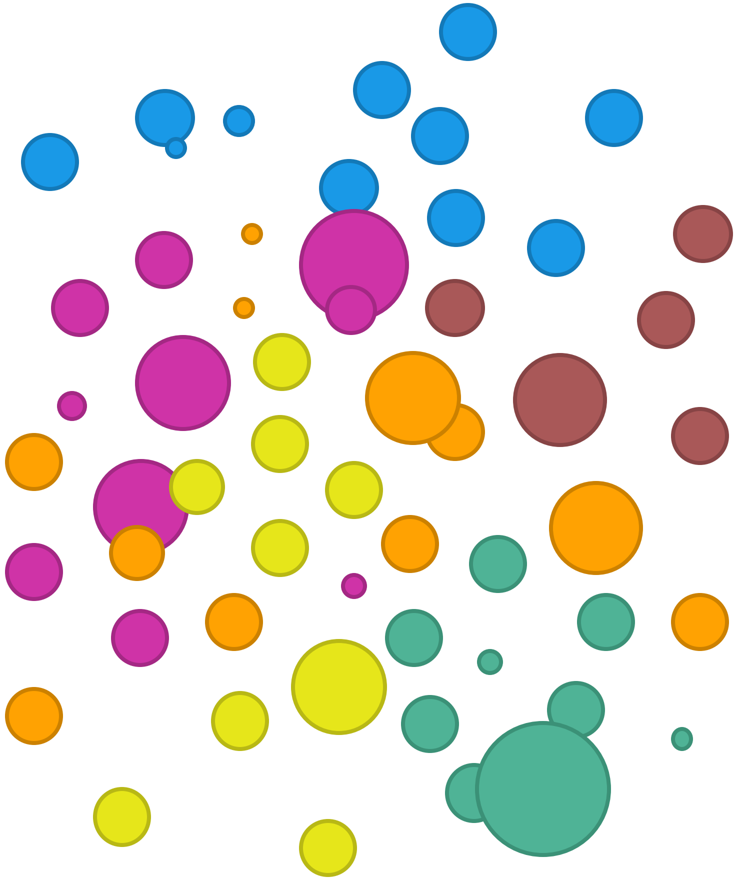
"Is a book x available?"

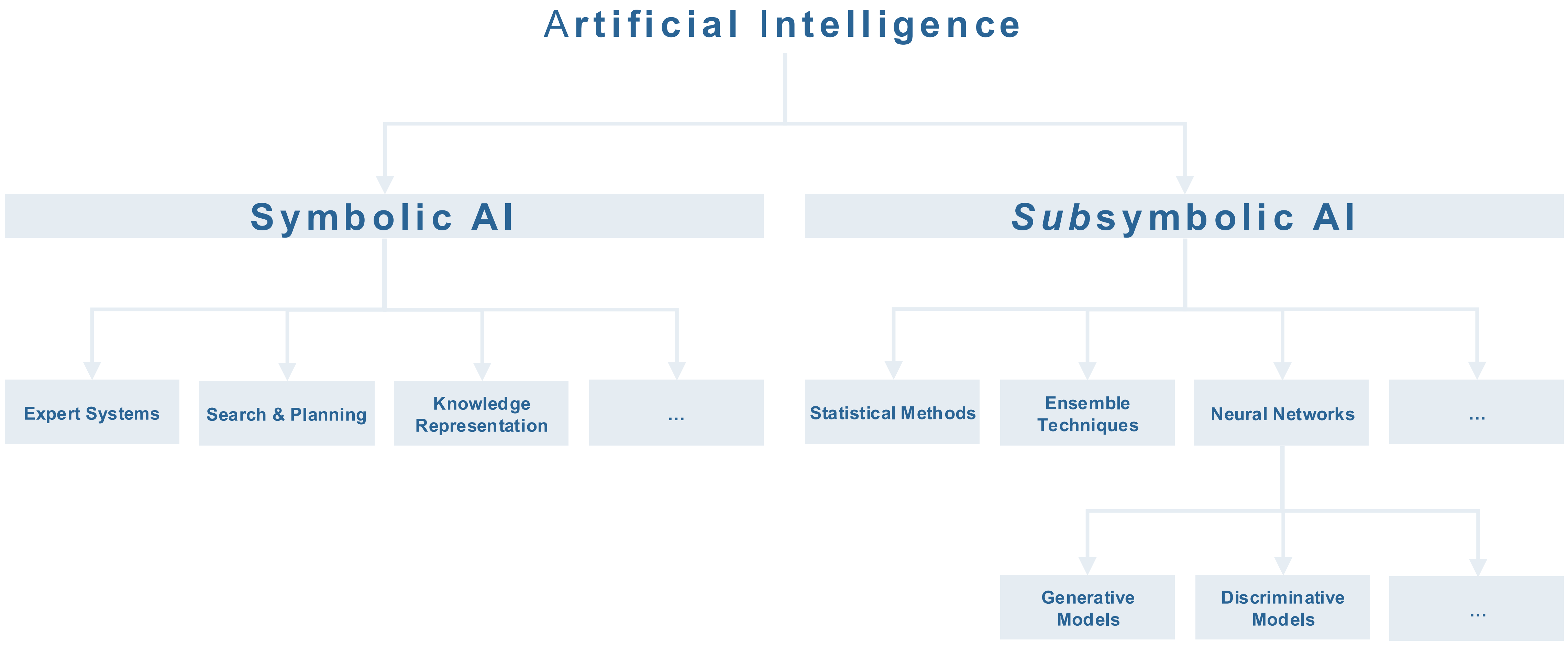
$\neg \exists y\, (Member(y) \land Borrowed(y, x))$

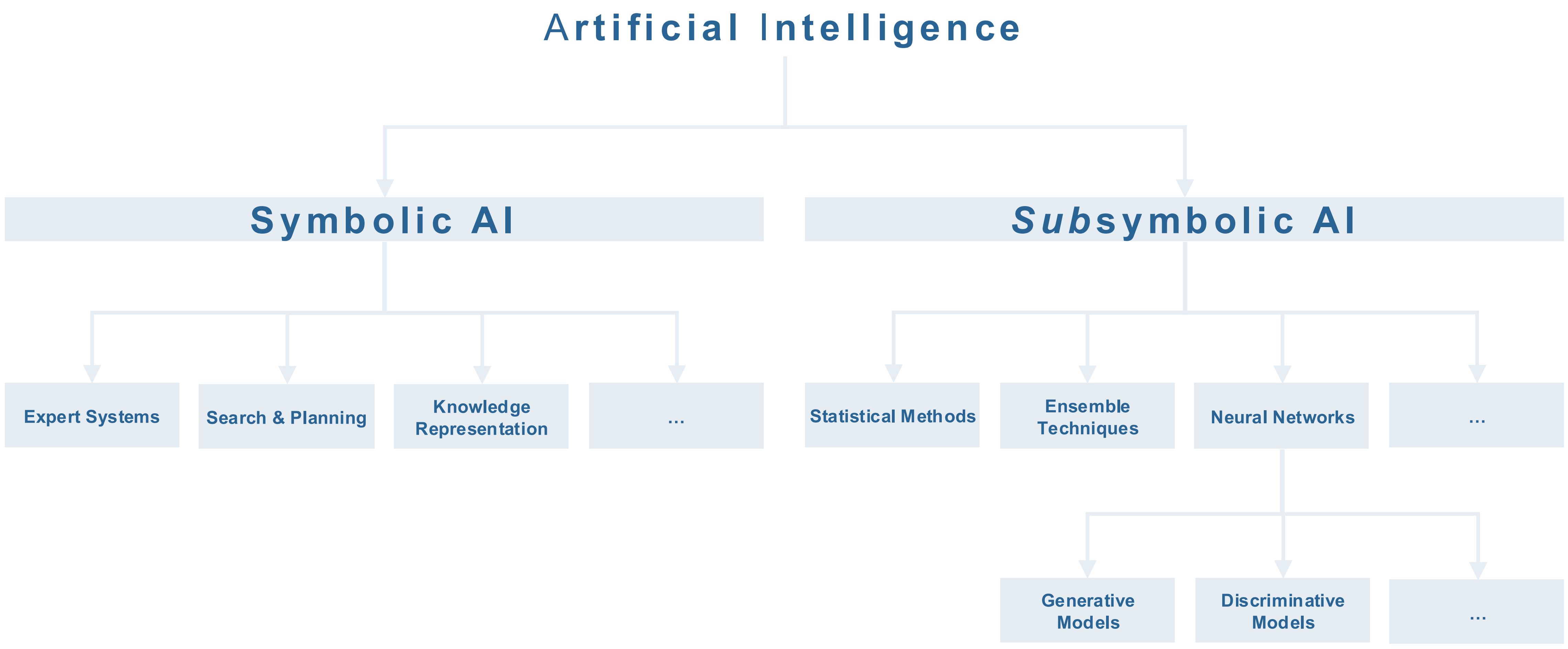
Subsymbolic AI



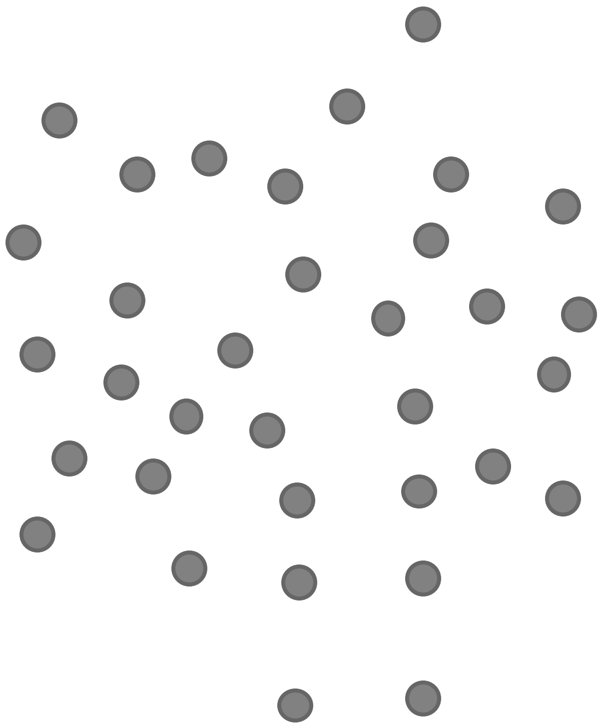
Subsymbolic AI

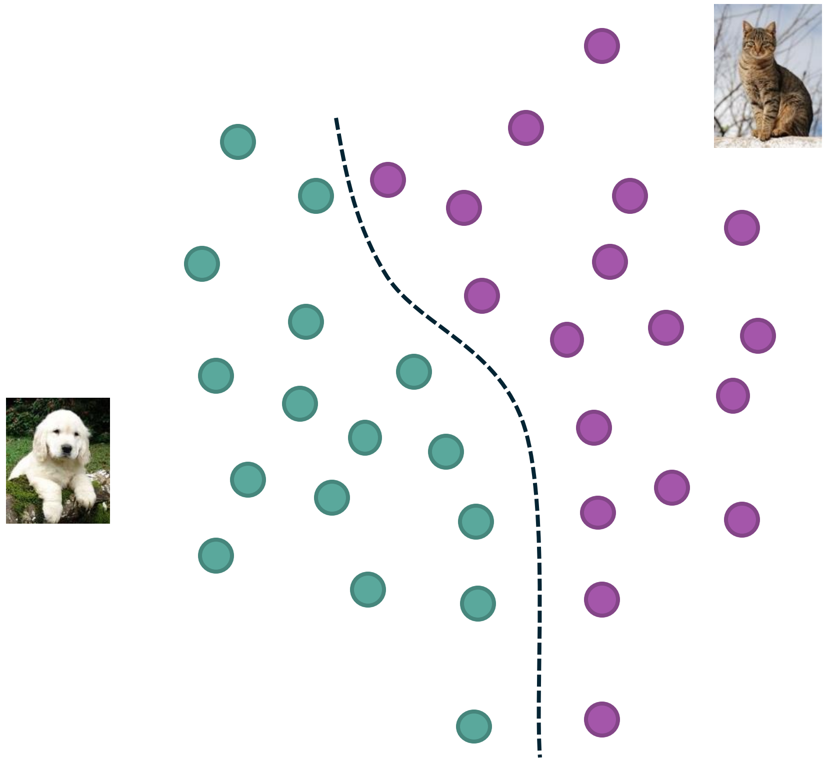



Artificial Intelligence
Symbolic AI
Subsymbolic AI
Expert Systems
Search & Planning
Knowledge Representation
...
Statistical Methods
Ensemble Techniques
Neural Networks
...
Generative Models
Discriminative Models
...


Artificial Intelligence
Symbolic AI
Subsymbolic AI
Expert Systems
Search & Planning
Knowledge Representation
...
Statistical Methods
Ensemble Techniques
Neural Networks
...
Generative Models
Discriminative Models
...

Discriminative Models

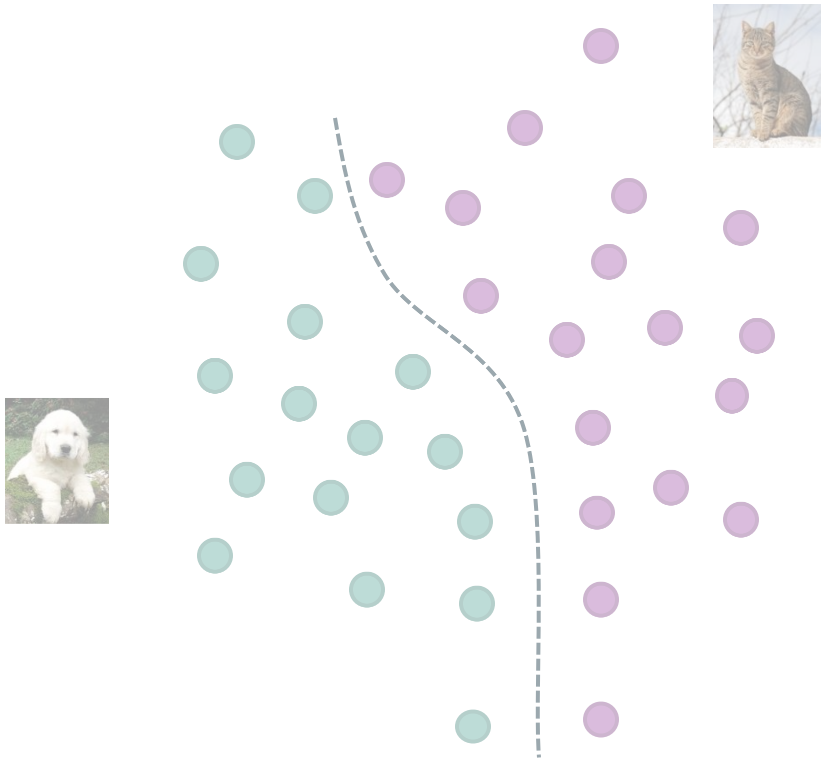




Discriminative Models

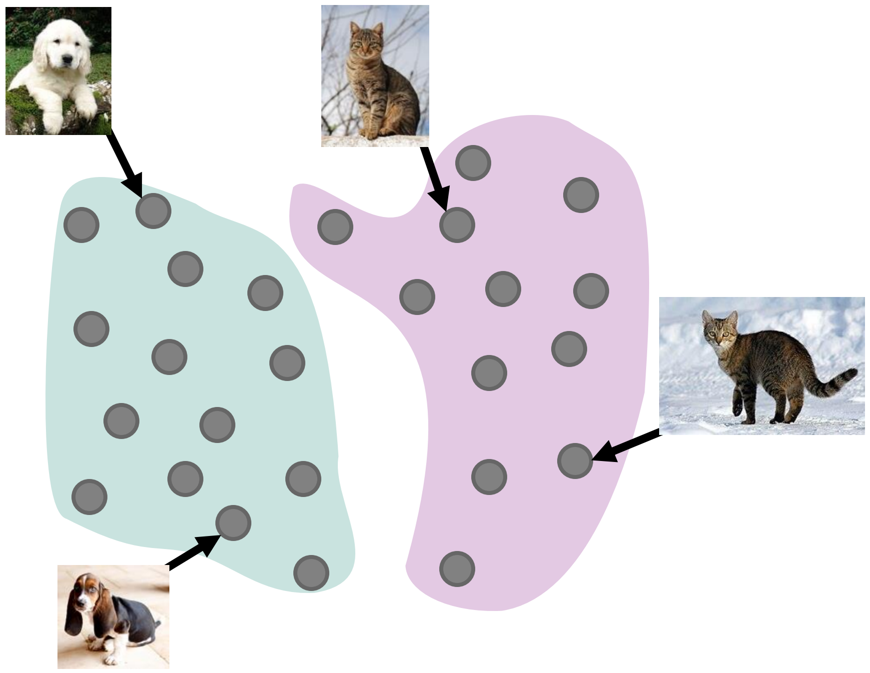
$p(Y|X)$

Generative Models



Discriminative Models

p(Y|X)

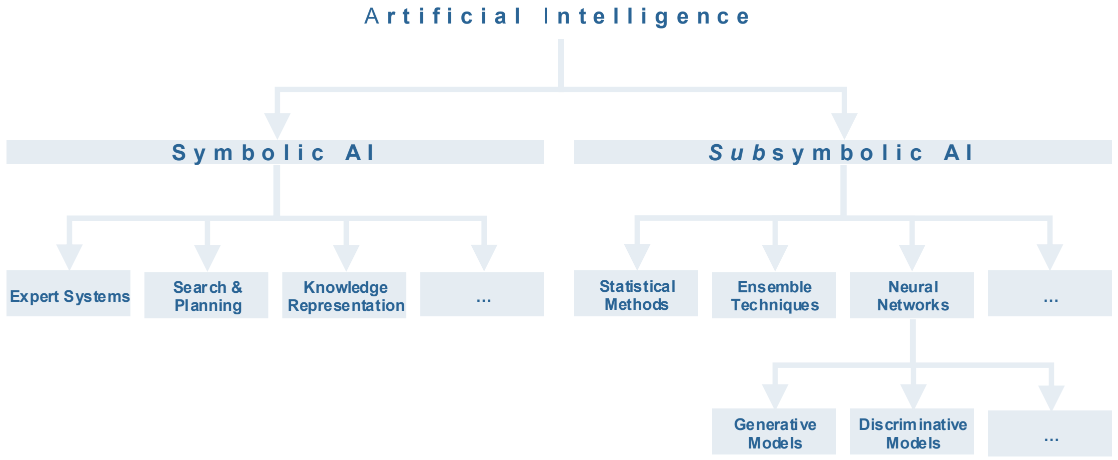


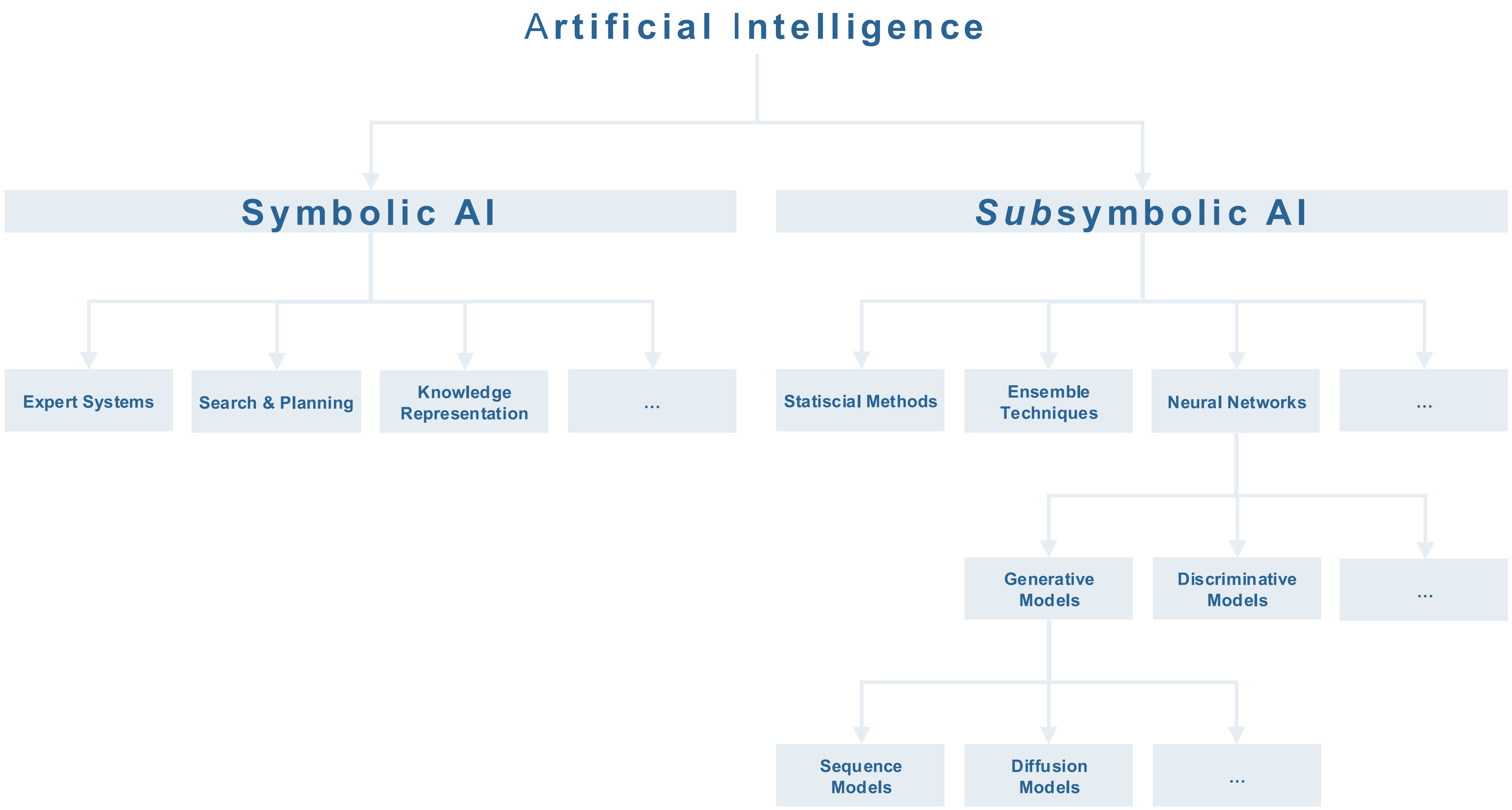
Generative Models

p(X,Y) or p(X)

What is (non-)generative AI?

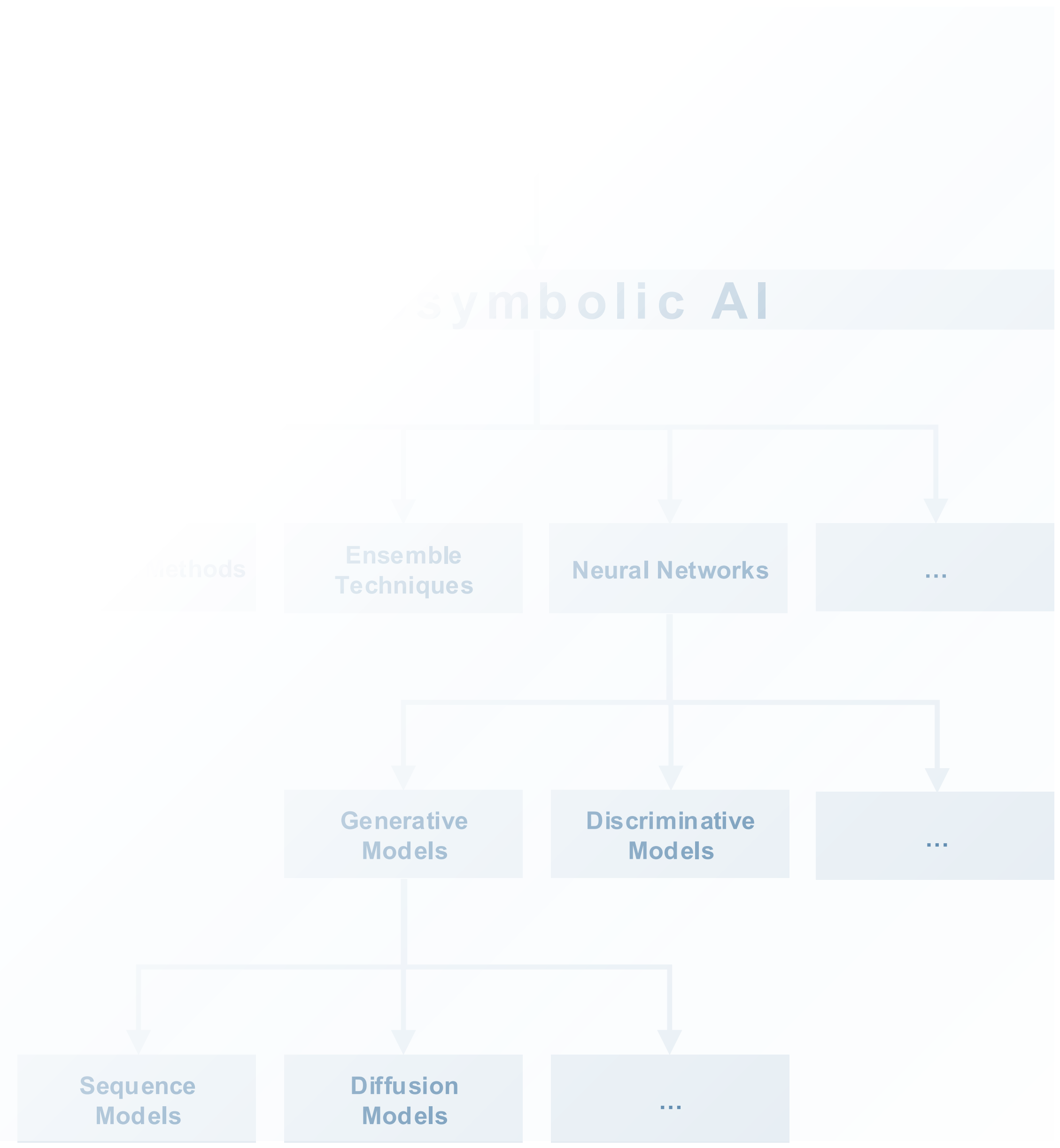
Recap




Artificial Intelligence
Symbolic AI
Subsymbolic AI
Expert Systems
Search & Planning
Knowledge Representation
...
Statiscial Methods
Ensemble Techniques
Neural Networks
...
Generative Models
Discriminative Models
...
Sequence Models
Diffusion Models
...

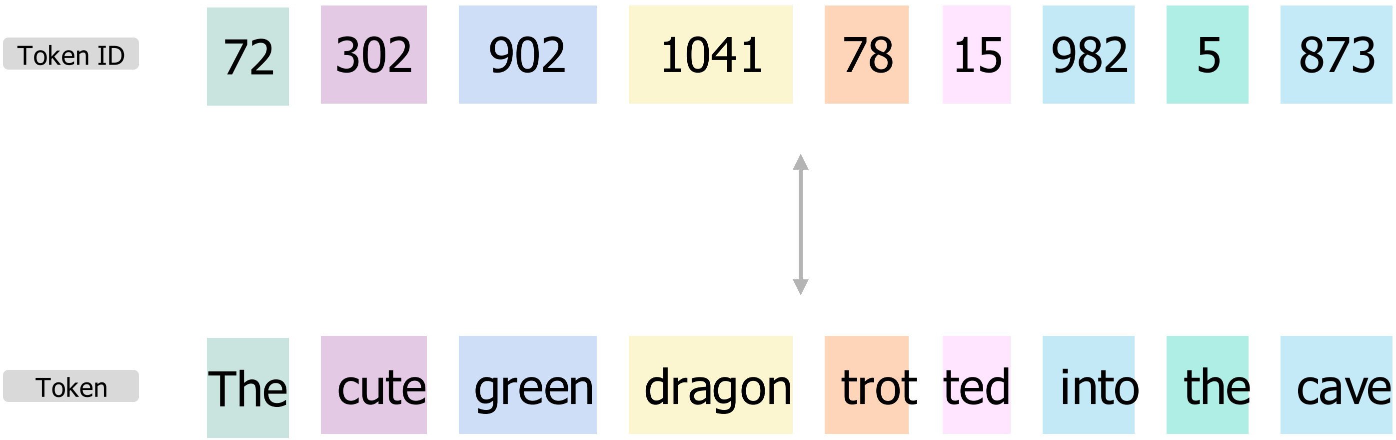
Sequence Modeling

Tokens



The cute green dragon trotted into the cave

The cute green dragon trot ted into the cave


Token ID
72
302
902
1041
78
15
982
5
873
Token
The
cute
green
dragon
trot
ted
into
the
cave

The goal of generative language modelling is to capture the "true distribution of language": $p(X)$

A few definitions:

- V … vocabulary of unique tokens $s_i \in V$
- n … maximum sequence length
- X … space of all possible token sequences

$$X = \{(s_1, s_2, \ldots, s_k) | s_i \in V, 0 \le k \le n, k \in \mathbb{Z}\} \text{ where } k \text{ is the sequence length}$$

- $x \in X$ … a particular sequence of tokens
- The probability of a given sequence x

$$p(x) = p(s_1) \cdot p(s_2 | s_1) \cdot p(s_3 | s_1, s_2) \cdot \ldots \cdot p(s_k | s_1, \ldots, sk_{-1})$$
$$= p(s_1) \cdot \prod_{i=2}^{k} p(si | s_1, \ldots, si_{-1})$$

The probability of a sequence is the product of the probability of the first token and the conditional probabilities of each subsequent token given all previous tokens.

The probability of a sequence is the product of the probability of the first token and the conditional probabilities of each subsequent token given all previous tokens.

A few more definitions:

- V_i … the random variable for a token at position i, and s_i is the actual token observed at that position.
- $p(V_i \,|s_1, \dots, si_{-1})$ … the probability distribution over all possible tokens that will come next.
- y_i … a one-hot encoded vector of dimension equal to the vocabulary size holding 1 for the correct token at position i and 0 otherwise. This acts as the ground truth.

$$L = -\sum_i \mathrm{y}_i \log\big(p(V_i|s_1, \dots, si_{-1})\big) = -\sum_i \log\big(p(s_i|s_1, \dots, si_{-1})\big)$$

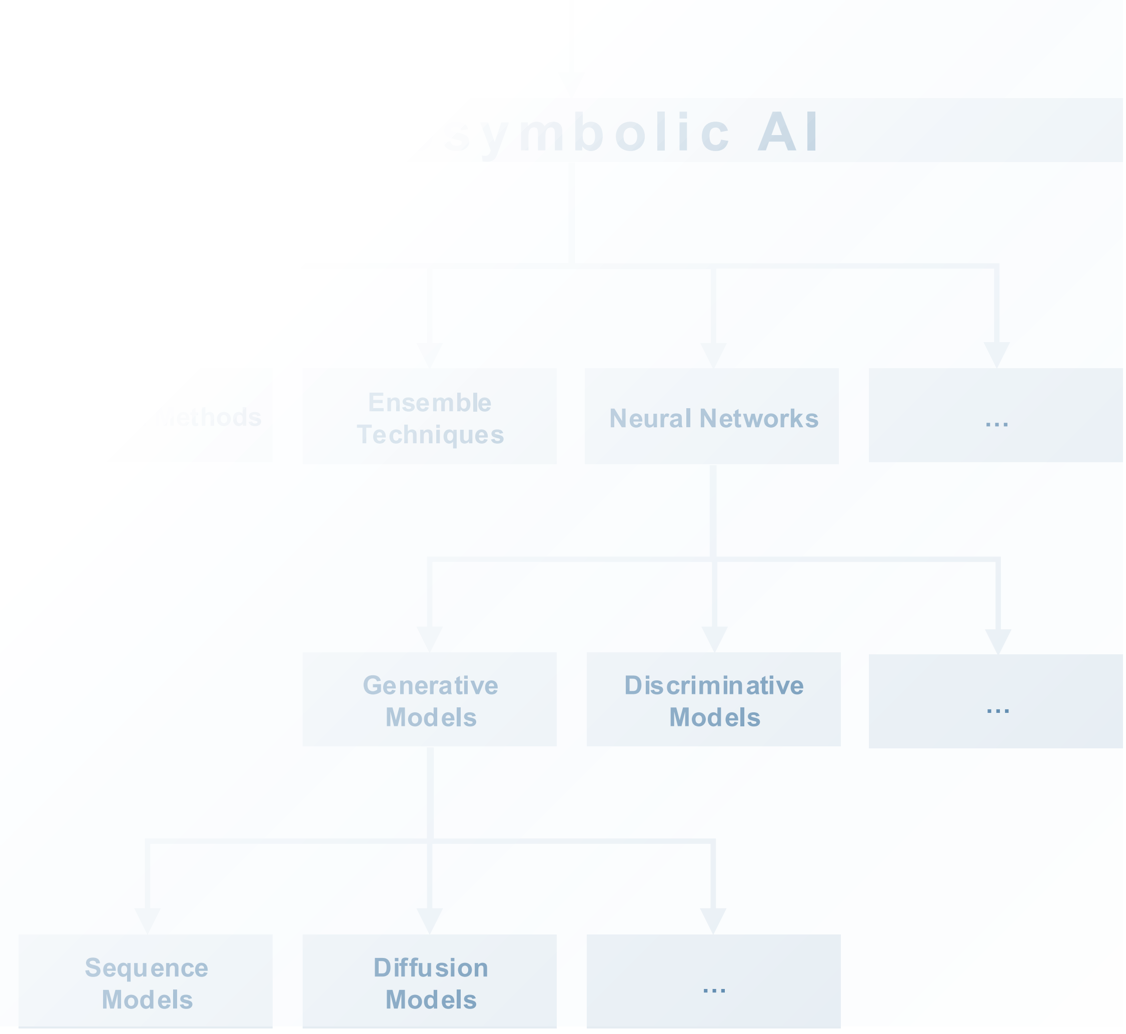
The loss function is defined as the negative log-likelihood of the correct tokens across all positions in the sequence, given the preceding tokens.

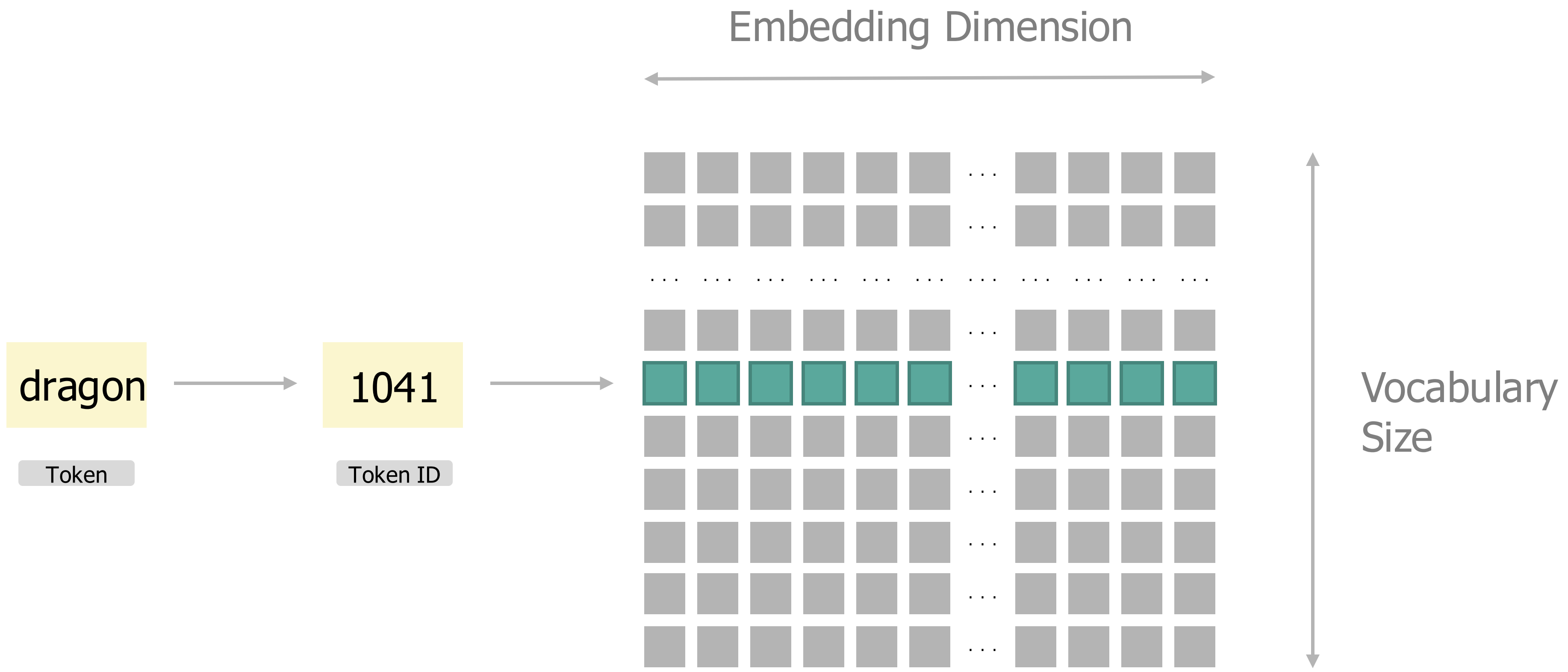
Tokens in Sequence Modeling

Recap

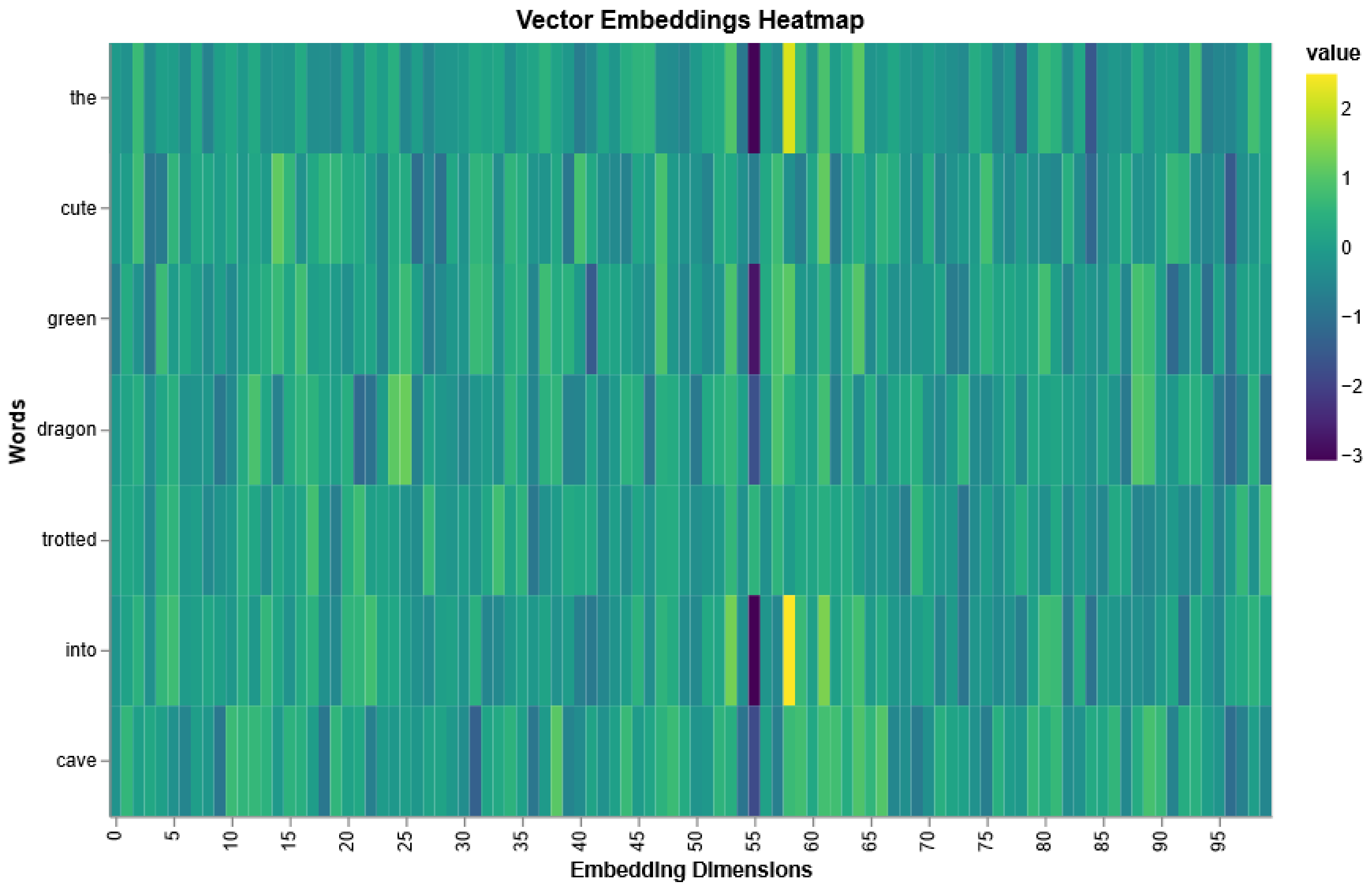
Sequence Modelling

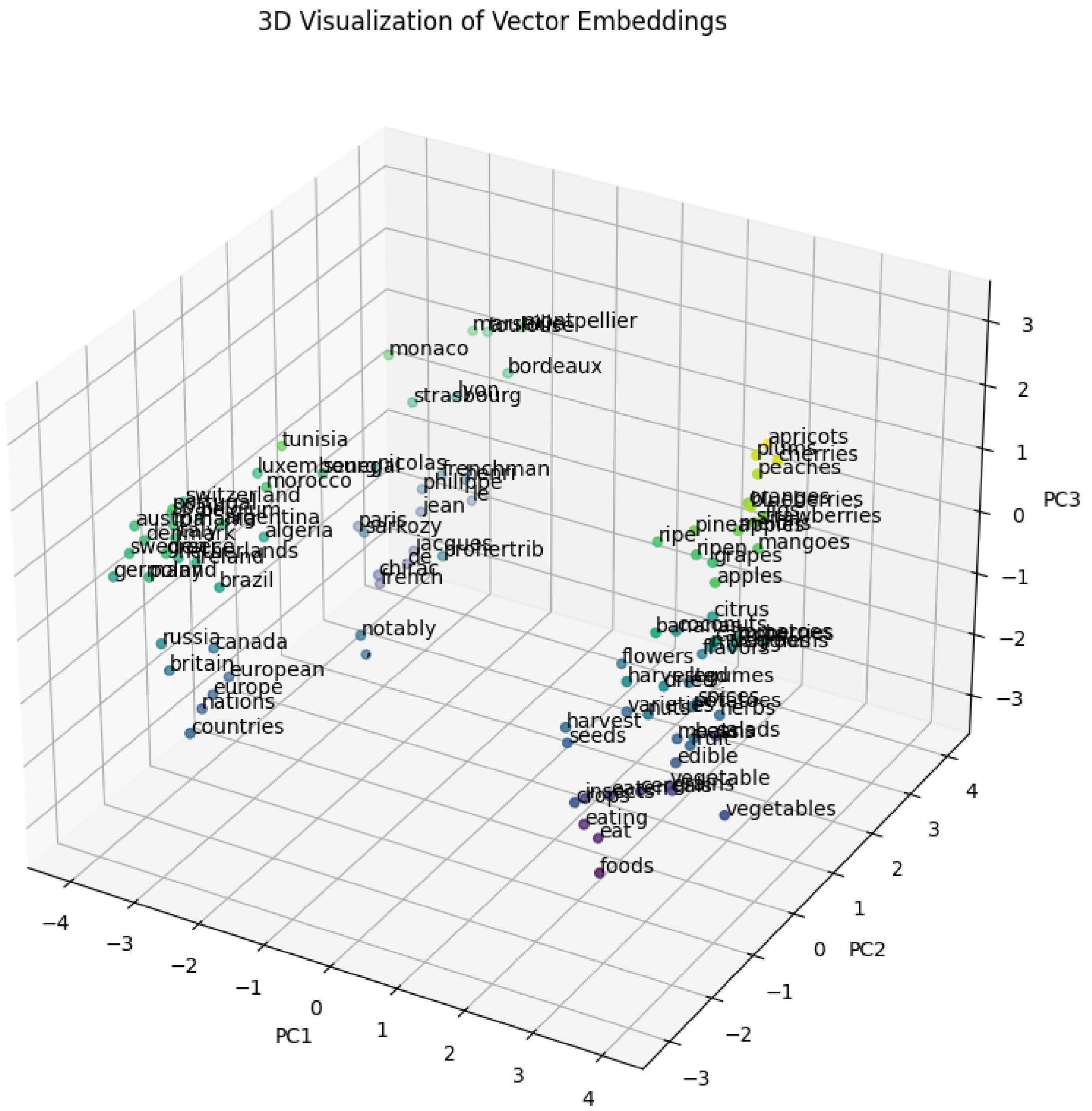
Embeddings

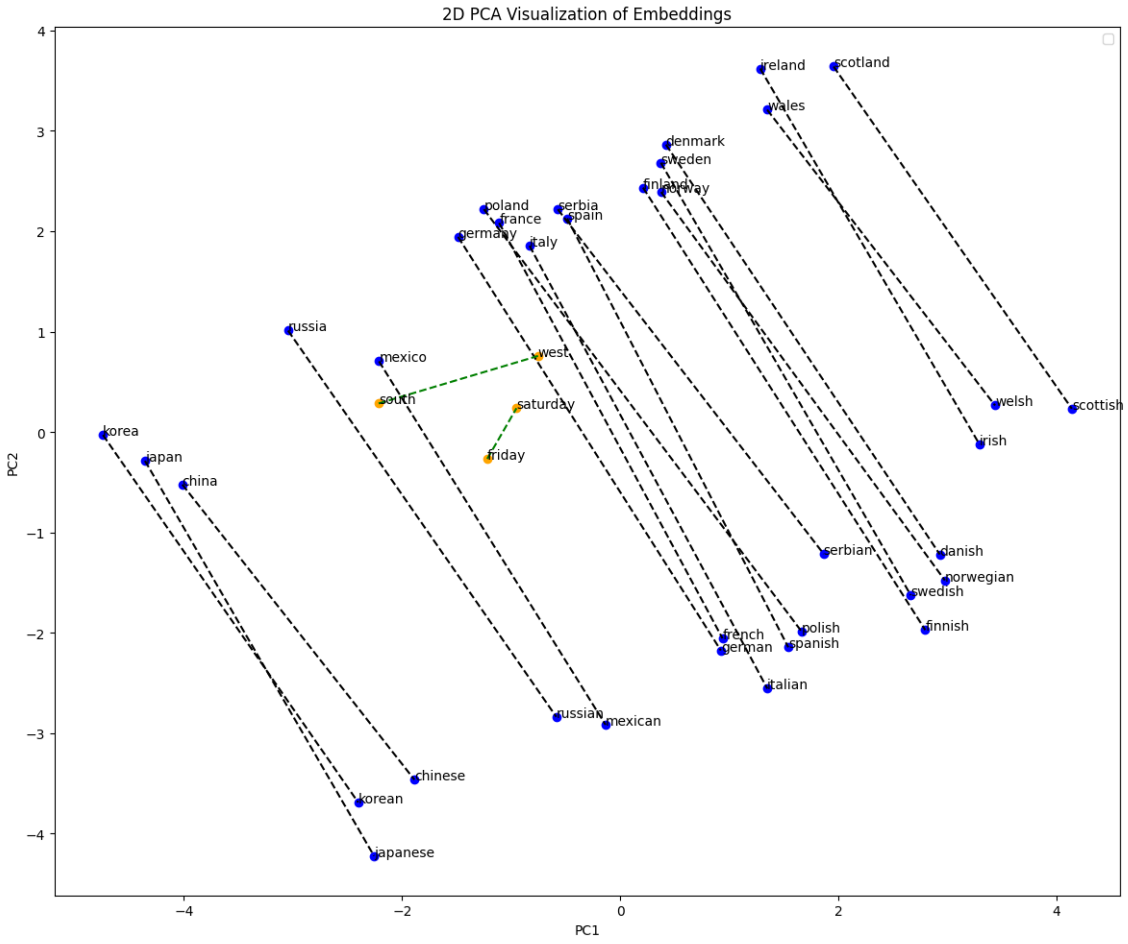



Embedding Dimension
dragon
Token
1041
Token ID
Vocabulary Size

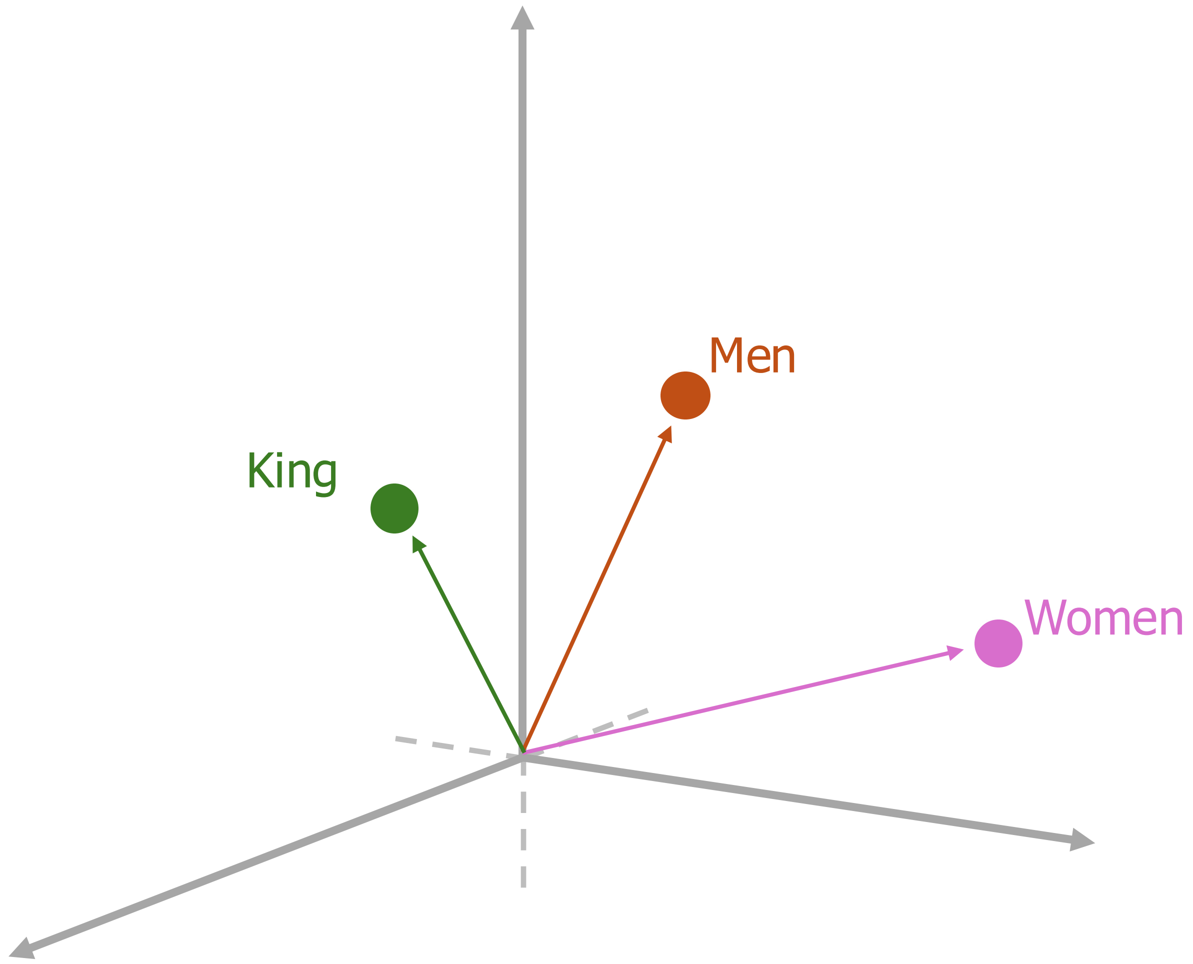
The	cute	green	dragon	trot
0.8	1.3	4.5	1.5	5.0
5.4	4.1	9.9	0.3	2.3
1.2	2.0	1.2	1.1	3.8
0.9	0.3	1.1	1.5	8.6
0.2	1.3	1.9	5.2	7.1
1.1	7.9	9.9	7.7	1.1
7.3	0.3	2.6	5.2	7.3
4.4	0.1	4.4	3.2	4.4
8.9	0.1	7.0	1.7	3.5
0.4	0.8	0.3	0.4	4.2
1.5	5.3	3.3	0.2	5.3
2.2	1.1	3.3	0.6	1.5
⋮	⋮	⋮	⋮	⋮
0.3	8.8	3.2	0.8	9.8

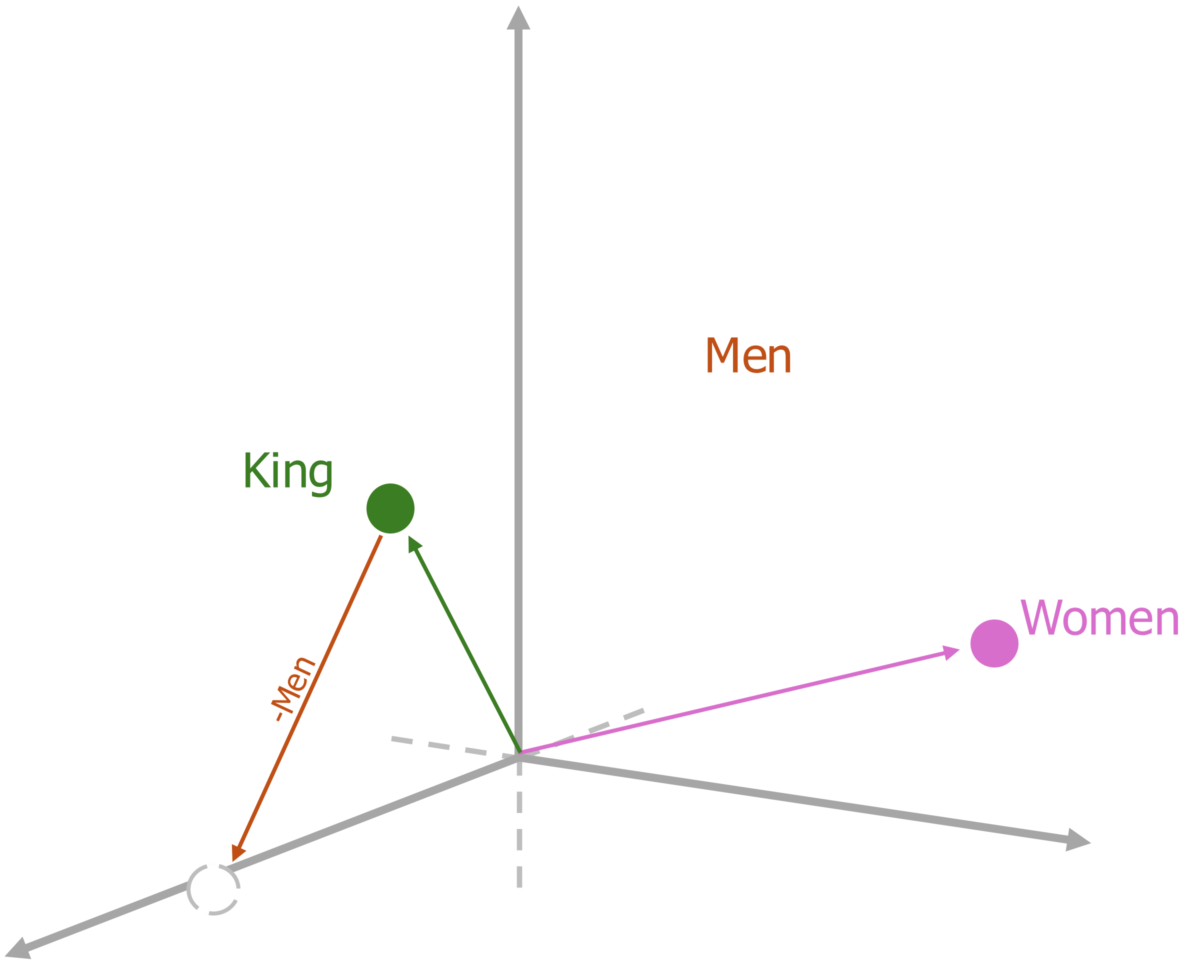

Vector Embeddings Heatmap
value
2
1
0
−1
−2
−3
Words
the
cute
green
dragon
trotted
into
cave
0
5
10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90
95
Embedding Dimensions

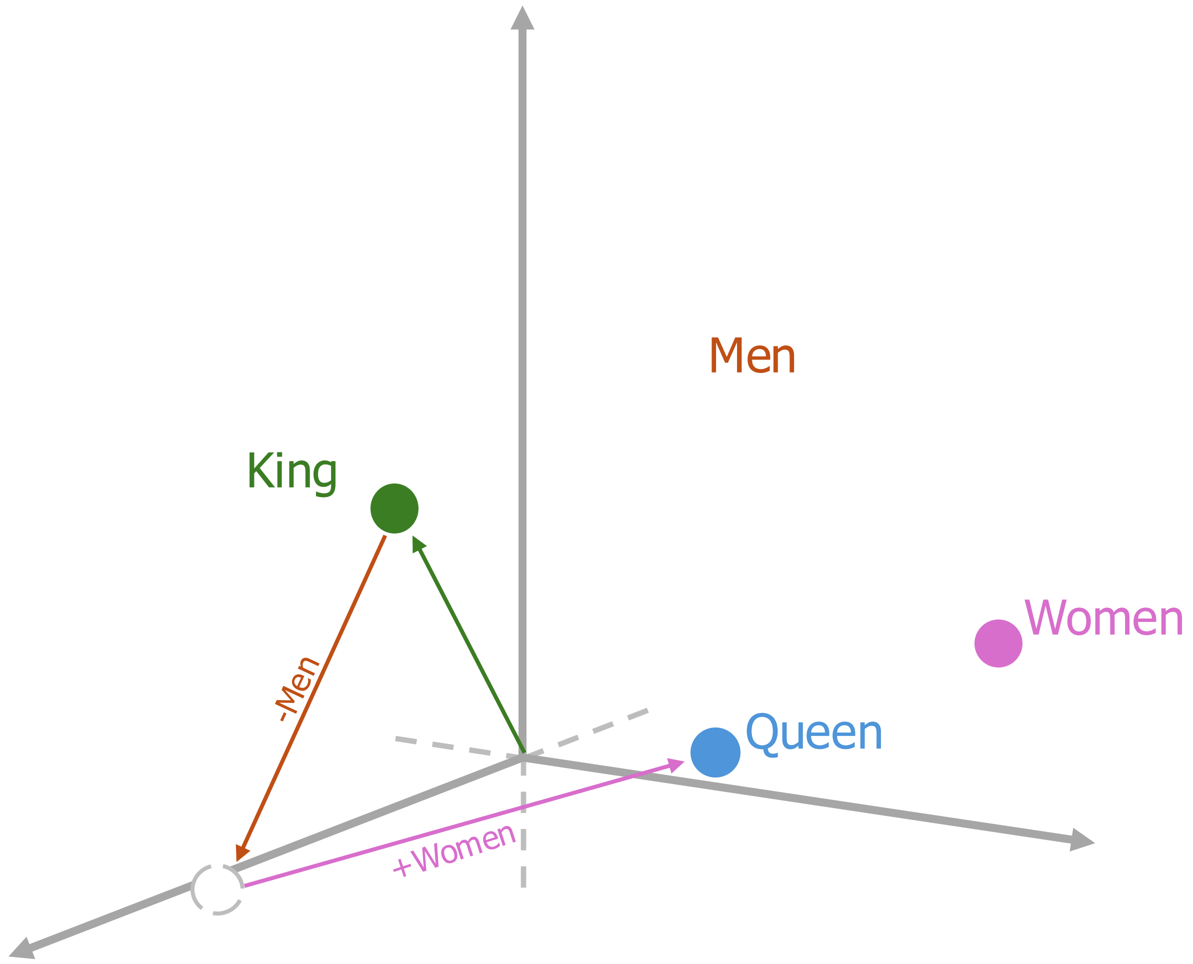

3D Visualization of Vector Embeddings
PC1
PC2
PC3
monaco
bordeaux
strasbourg
tunisia
morocco
switzerland
algeria
brazil
russia
canada
britain
european
europe
nations
countries
notably
paris
sarkozy
jean
philippe
jacques
chirac
french
nicolas
apricots
cherries
peaches
strawberries
mangoes
grapes
apples
ripe
citrus
flowers
harvest
seeds
herbs
edible
vegetable
vegetables
eating
eat
foods

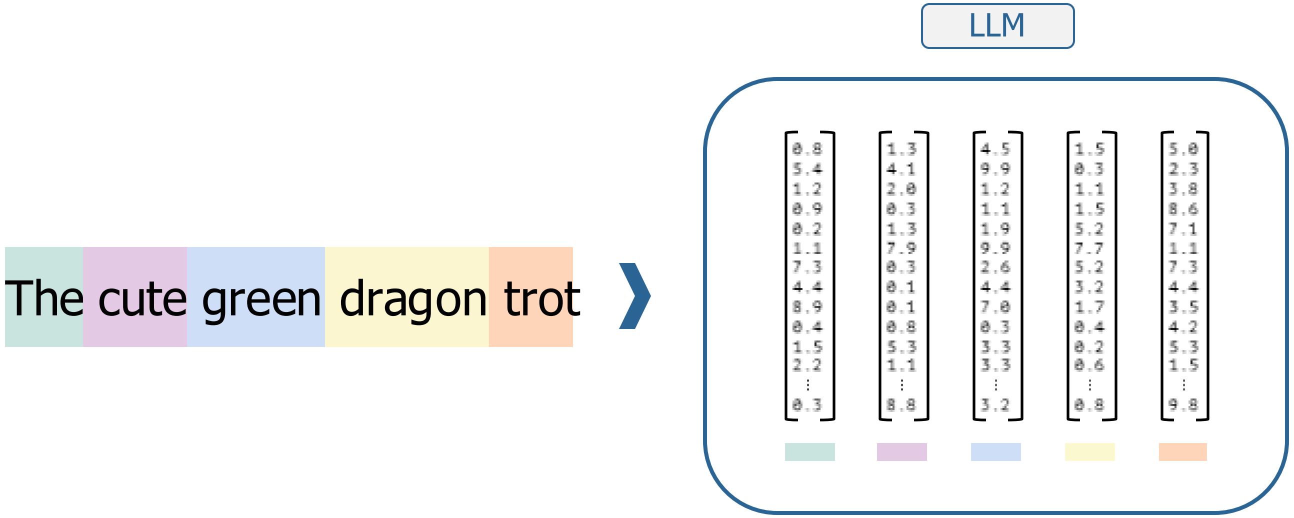


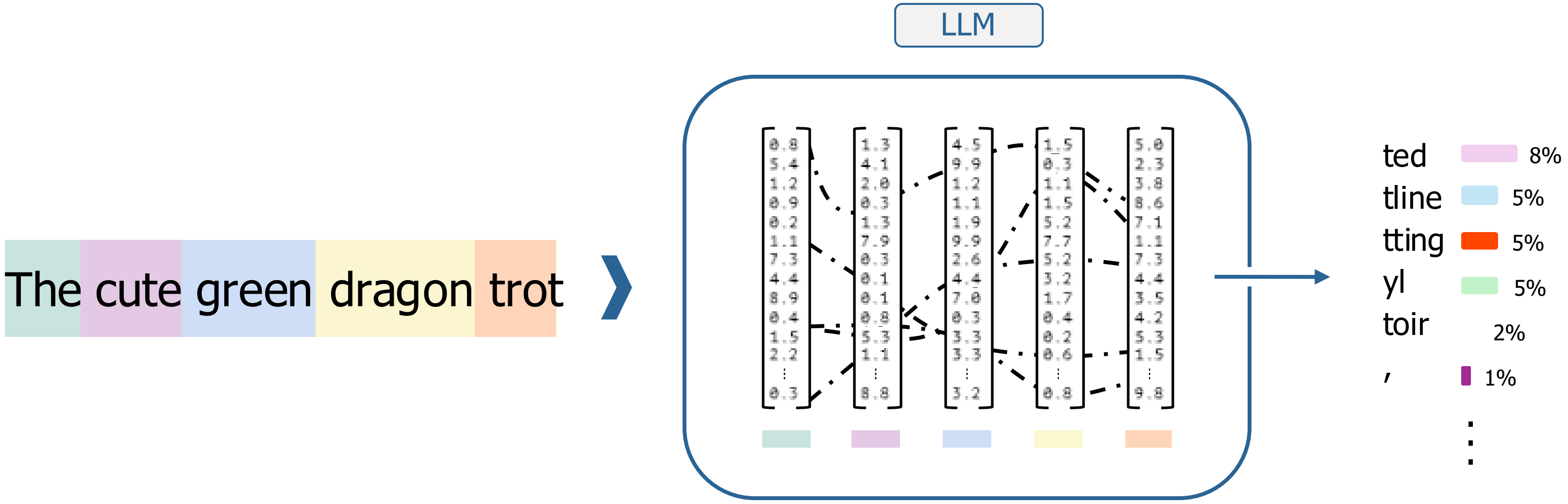

```python
import gensim.downloader

w2v = gensim.downloader.load('glove-wiki-gigaword-300’)

# Quantify semantic relationship between
# "germany" and "german"
relation = w2v["germany"] - w2v["german"]

# Apply relation to different embedding
prediction = w2v["austrian"] + relation
word, similarity = w2v.most_similar(
          positive=[prediction],
          topn=1)[0]

print(f'Word: "{word}" with {similarity} similarity’)
# -> Word: "austria" with 0.8964902758598328 similarity
```


Men
King
Women

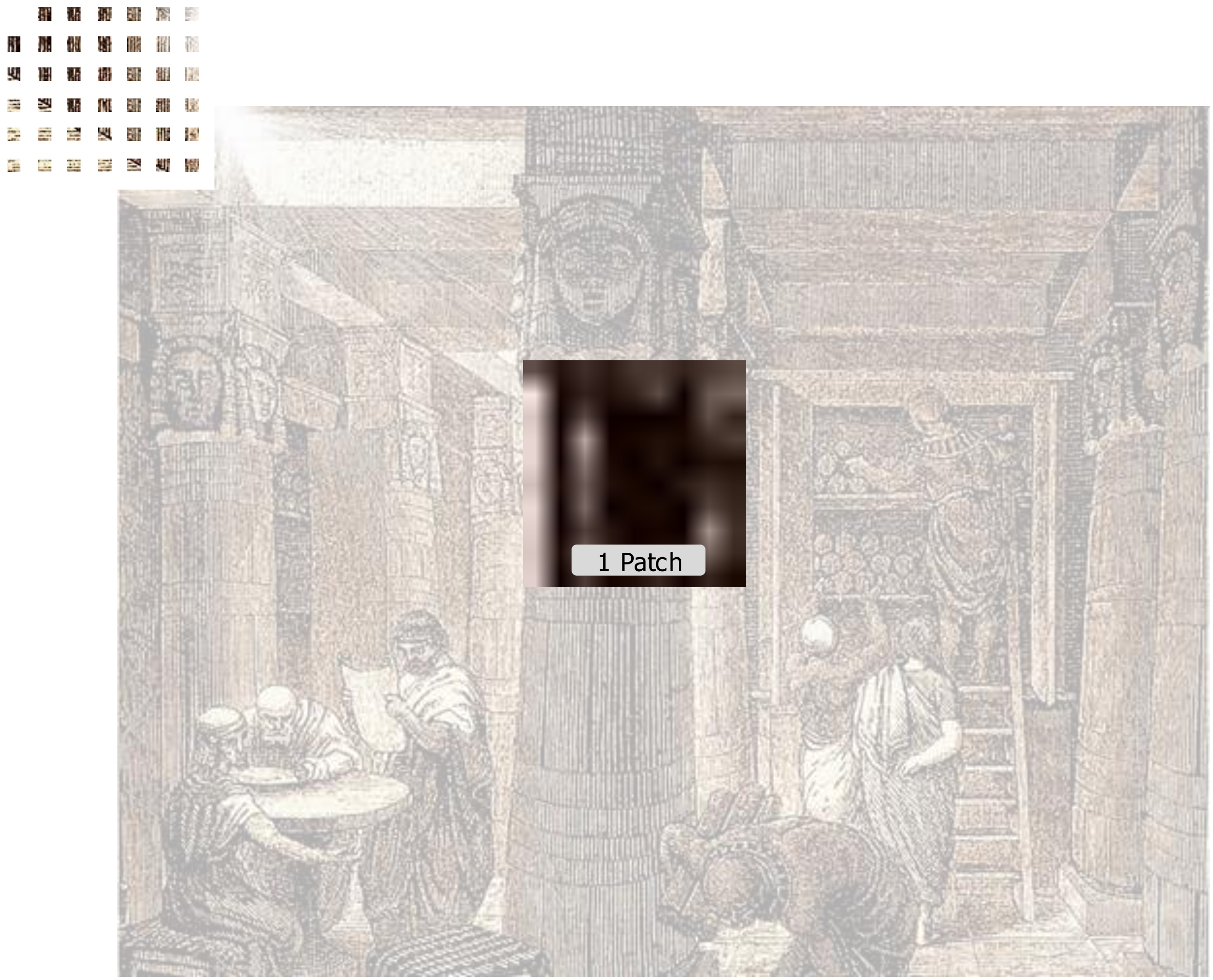

Men
King
-Men
Women

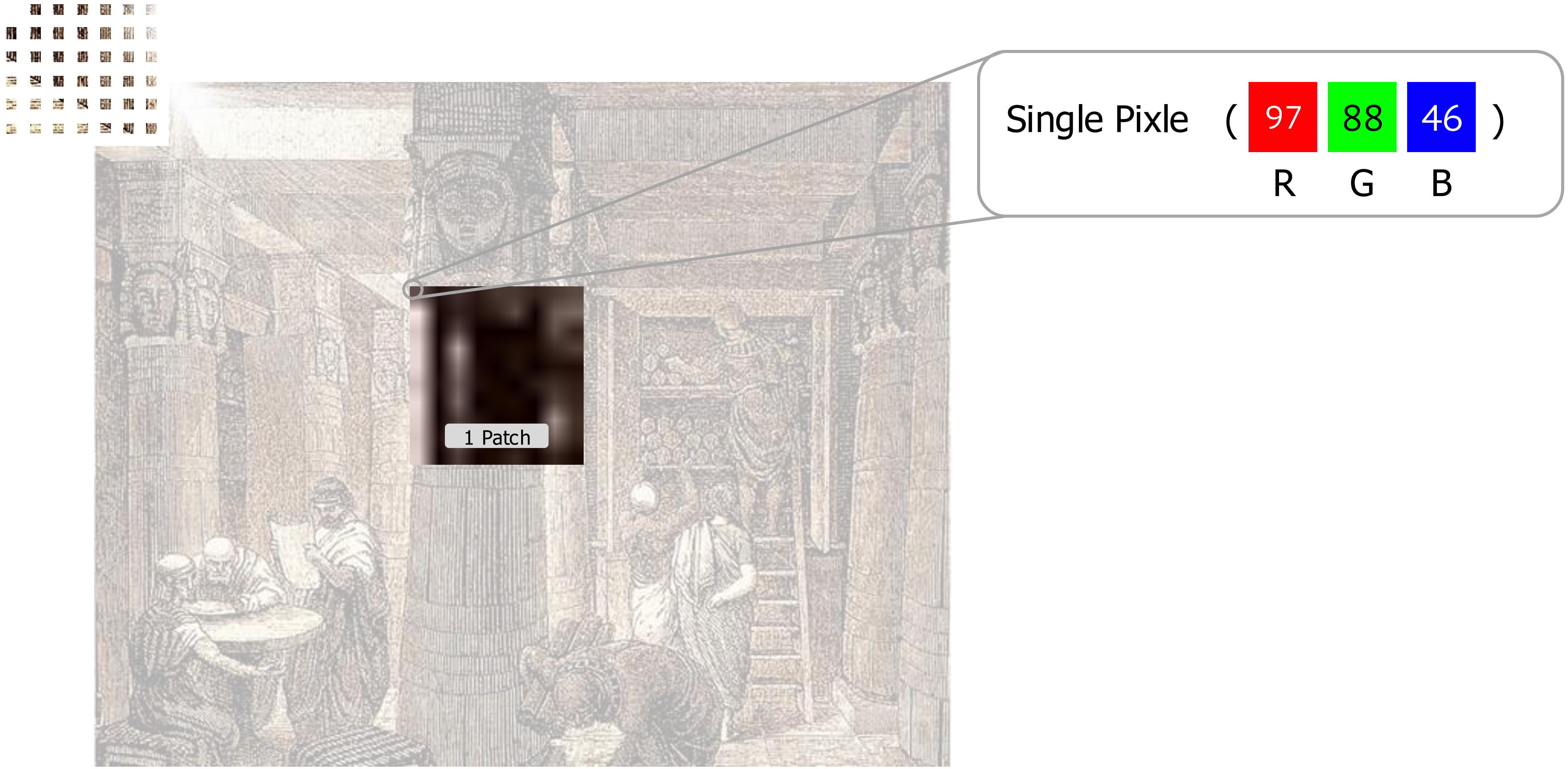

Men
King
-Men
Women
Queen
+Women

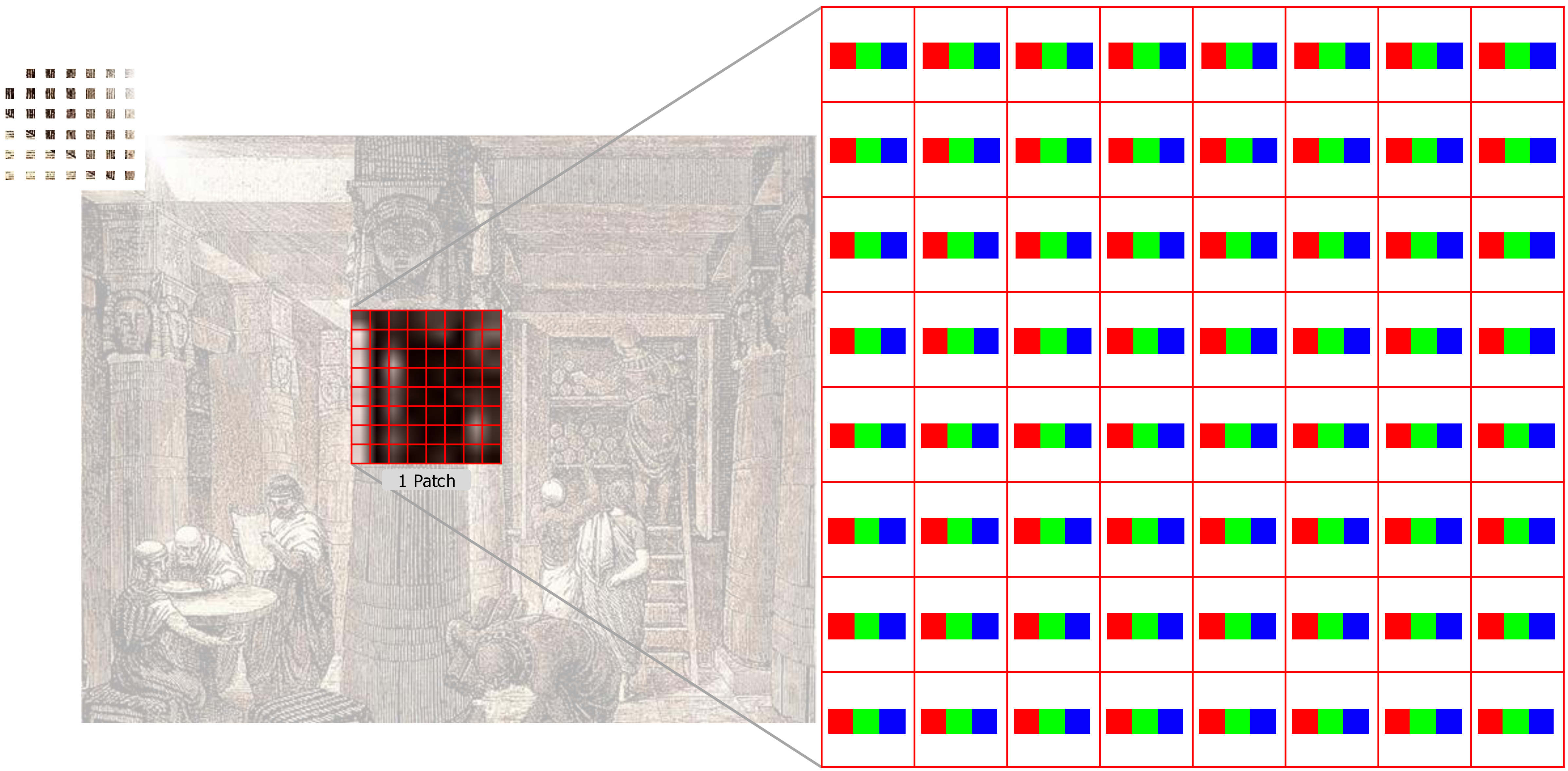

The cute green dragon trot
LLM

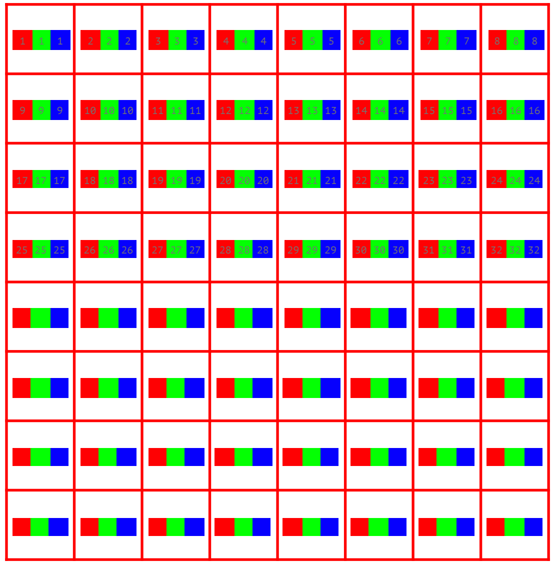


The cute green dragon trot
LLM
ted 8%
tline 5%
tting 5%
yl 5%
toir 2%
, 1%


1 Patch

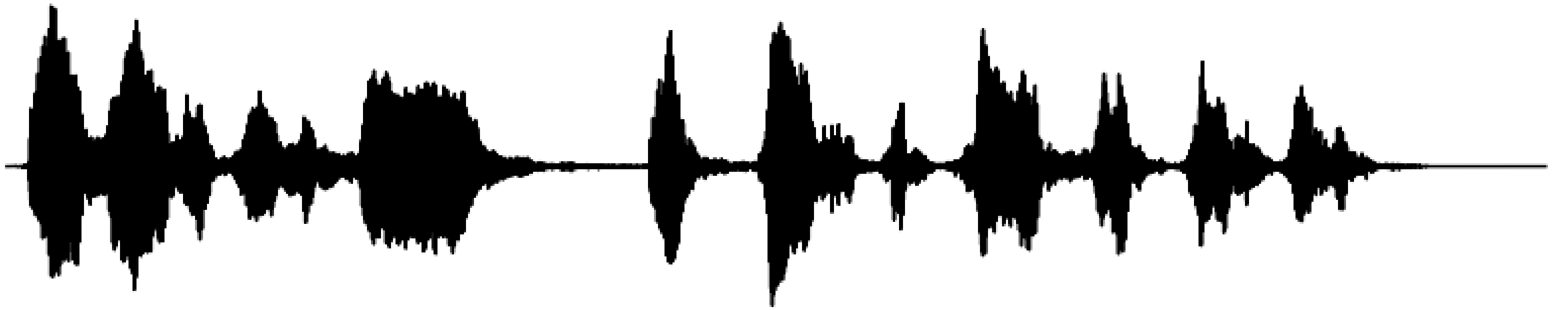

1 Patch
Single Pixle (97 88 46)
R G B


1 Patch

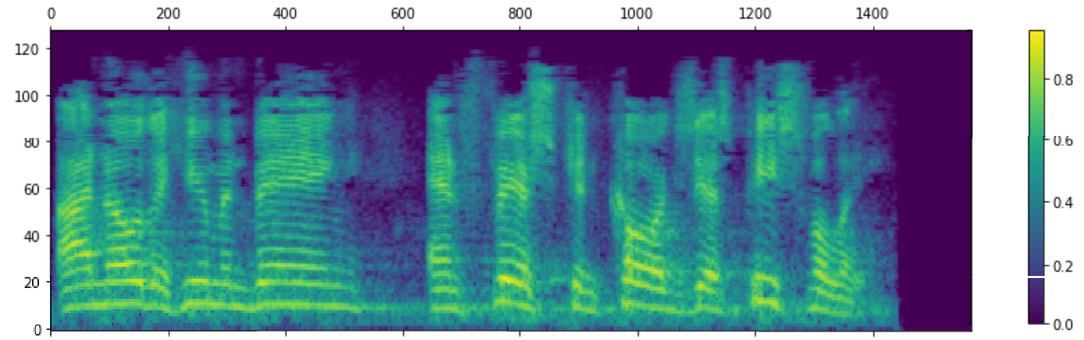


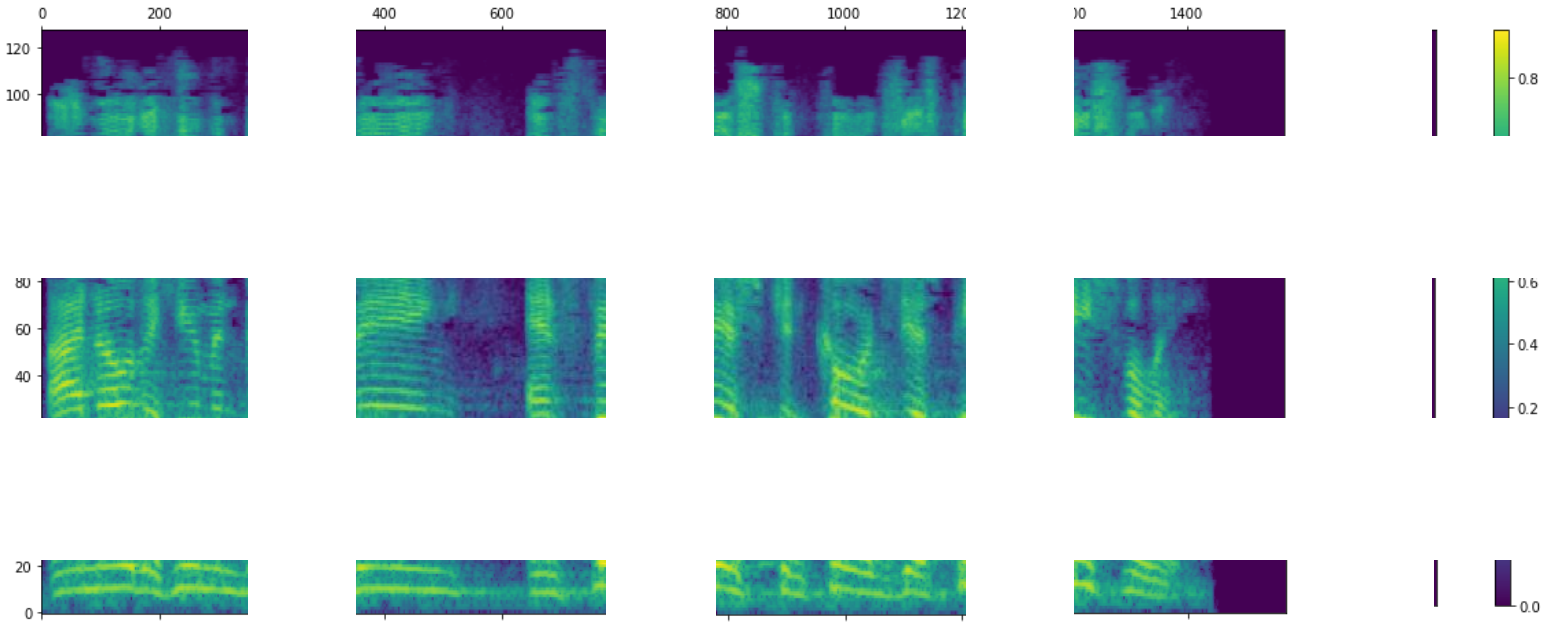
1 Patch


Embedding
1 Patch

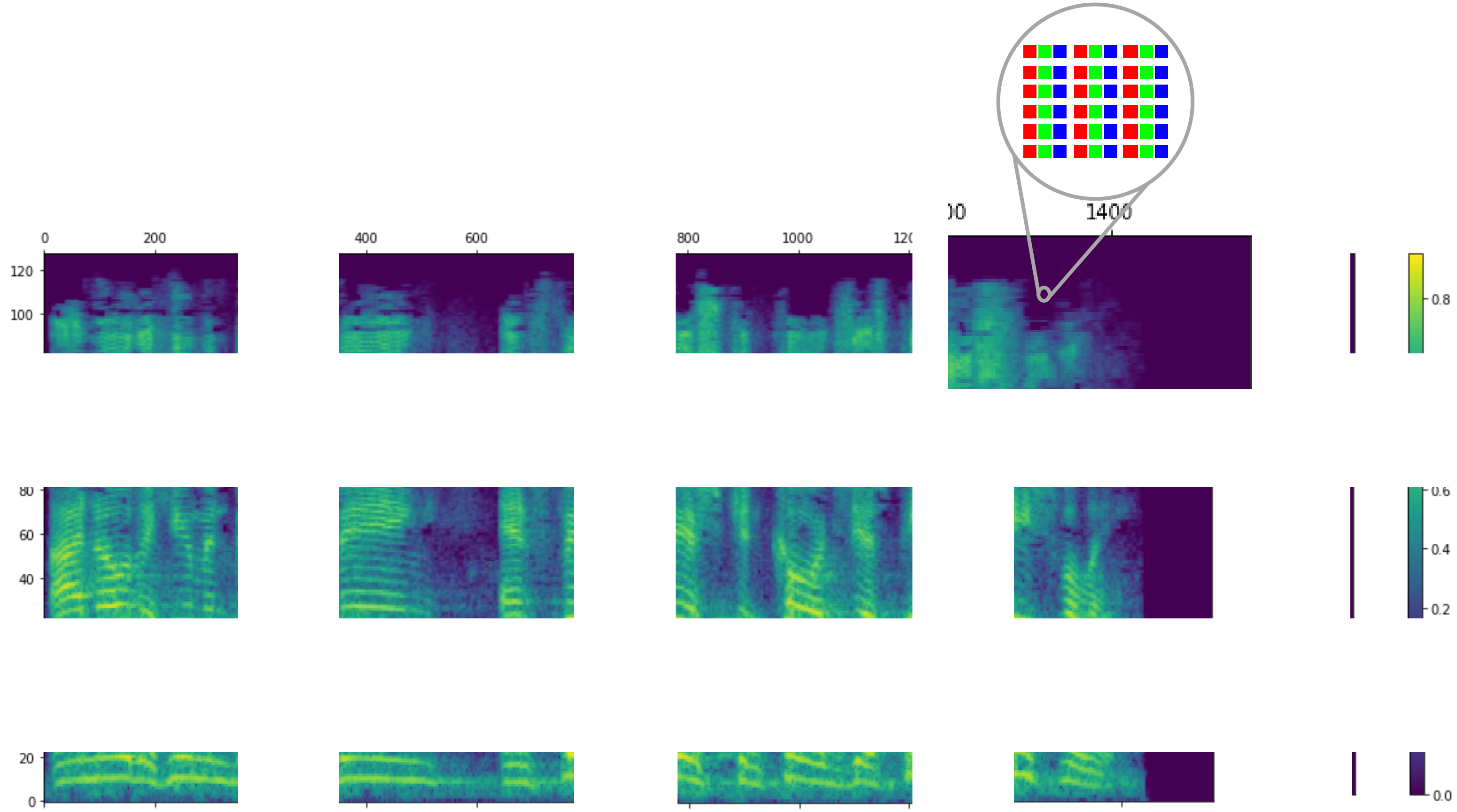


$$(\mathcal{F}f)(y) = \frac{1}{\sqrt{2\pi}^{\,n}} \int_{\mathbb{R}^n} f(x)\, \mathrm{e}^{-\mathrm{i}y\cdot x}\, \mathrm{d}x$$





$$(\mathcal{F}f)(y) = \frac{1}{\sqrt{2\pi}^{\,n}} \int_{\mathbb{R}^n} f(x)\, \mathrm{e}^{-\mathrm{i}y\cdot x}\, \mathrm{d}x$$



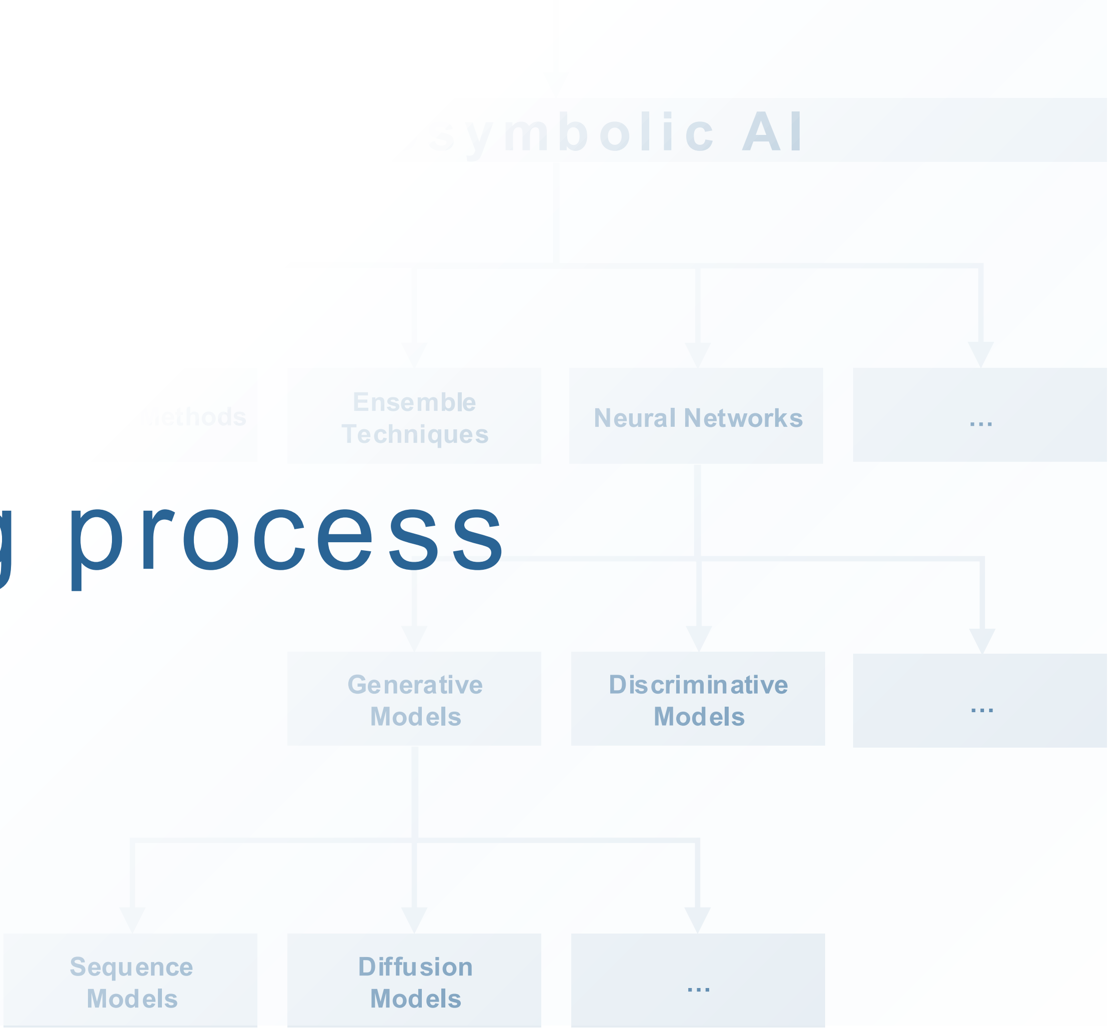
$$(\mathcal{F}f)(y) = \frac{1}{\sqrt{2\pi}^{\,n}} \int_{\mathbb{R}^n} f(x)\, \mathrm{e}^{-\mathrm{i}y\cdot x}\, \mathrm{d}x$$

Generative Sequence Modeling

Recap

Difussion Models

Stochastic denoising process



Diffusion Models

Diffusion in the context of diffusion models is a **gradual transformation** that **adds random noise** (e.g. Gaussian noise) **to data** (like images) eventually converting the original data into pure noise.

forward diffusion

$q(x_0)$ $q(x_1 \mid x_0)$ … $q(x_t \mid x_{t-1})$

backward diffusion

$q(x_t|x_{t-1}) = \mathrm{N}\left(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t \boldsymbol{I}\right)$
with $\beta_t \epsilon (0,1)$

"Create a normal distribution that centers around $\sqrt{1-\beta_t}x_{t-1}$ with variance $\beta_t I$"

Insight: $q(x_t|x_0) = \mathrm{N}\left(x_t; \sqrt{a_t}x_0, (1-a_t)\boldsymbol{I}\right)$
with $a_t = \prod 1 - \beta_i$

"Chaining multiple Gaussian distributions, allows us to directly sample from x_0 to any timestep t."

1. $q(x_t \mid x_{t-1})$ represents the noised image at step t conditioned by its previous step
2. Image src: https://cvpr2022-tutorial-diffusion-models.github.io/
3. Nice mathematical introduction to diffusion models: https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Introduction To Generative AI

Recap

Symbolic vs. Subsymbolic AI

- **Symbolic AI** defines a set of formal, humanly understandable symbols and employs explicit rules to draw logical conclusions.
- **Subsymbolic AI** leverages mathematical models to learn patterns from data without requiring hand-crafted rules.

Discriminative vs. Generative Models

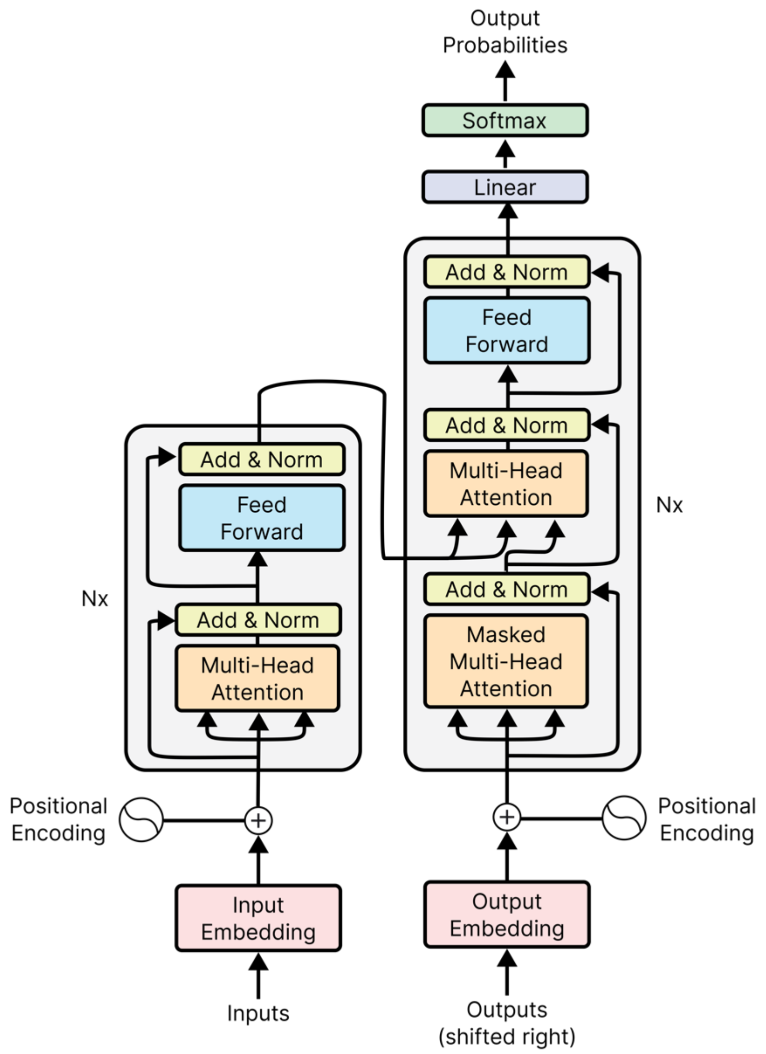
- **Discriminative Models** learn the conditional probability $p(Y|X)$, modeling the decision boundary between classes or the mapping from inputs to outputs.
- **Generative Models** learn the joint probability $p(X,Y)$ or, in the unsupervised case, the data distribution $p(X)$.

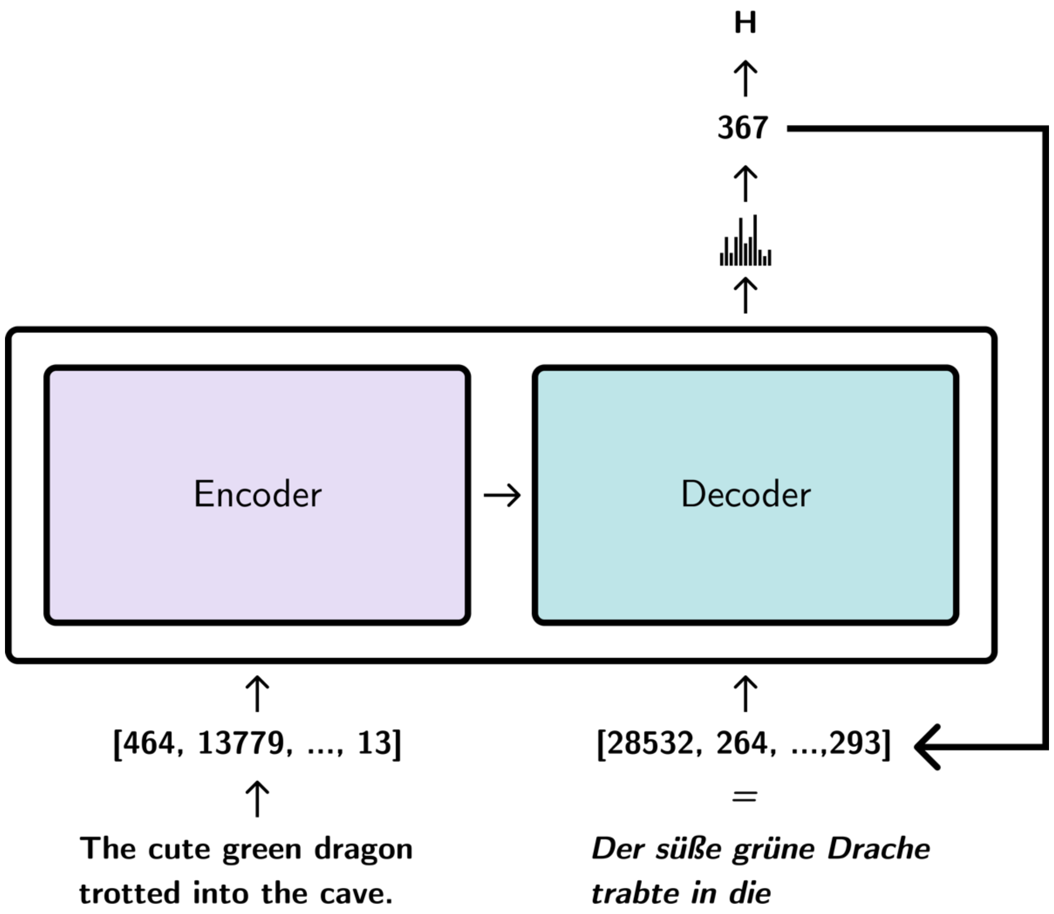
Sequence Modeling vs. Diffusion Modeling

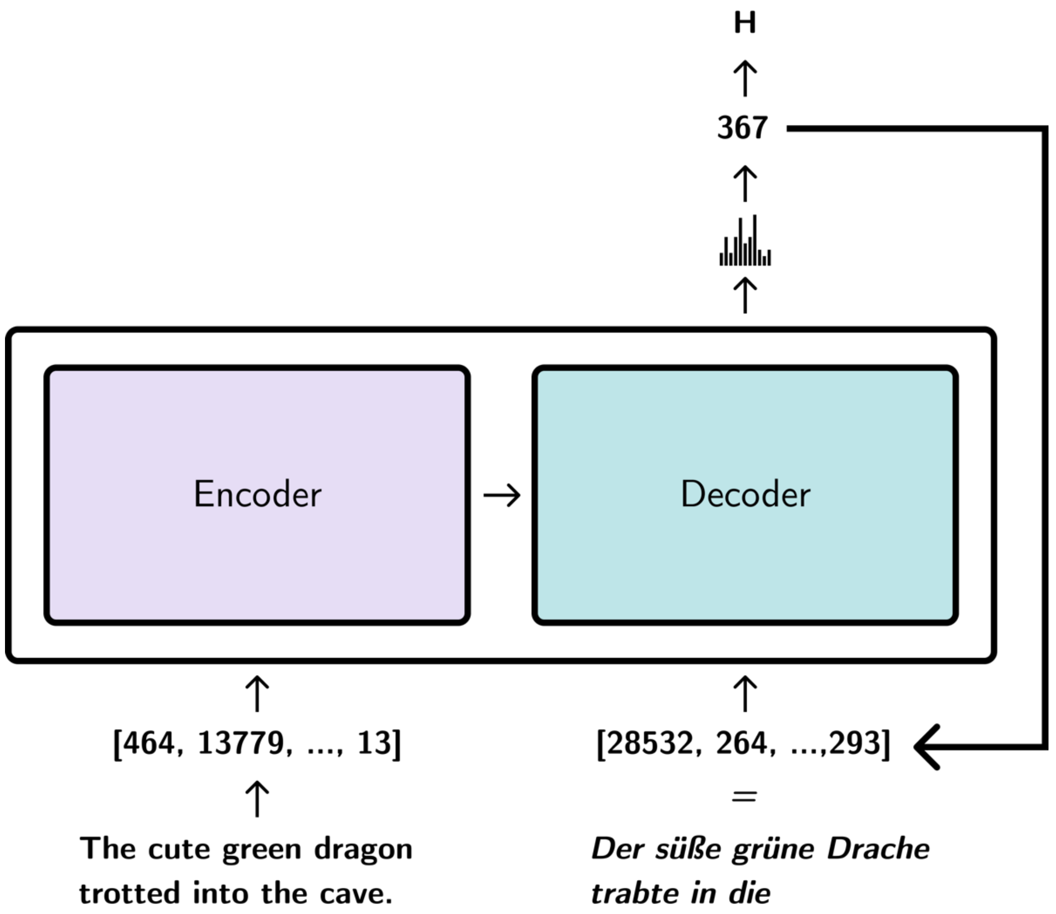
- **Sequence Models** process data as ordered sequences of discrete units (tokens in language models or patches in vision models). Those data atoms are mapped to continuous embeddings – dense vector representations capturing semantic or structural information.
- **Diffusion Models** learn the data distribution $p(X)$ by training a model to iteratively denoise data, reversing a gradual noising process applied during training.

A bird's eye view of the

Transformer Architecture.

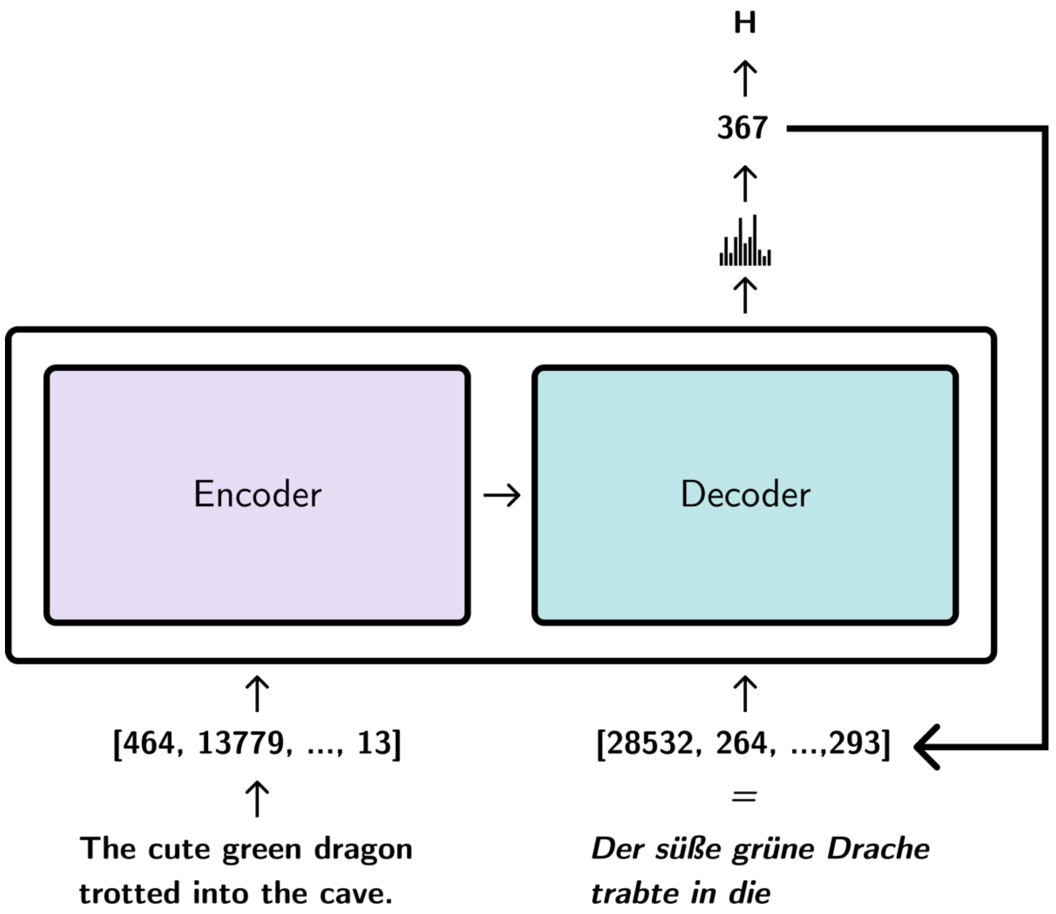



H
367
Encoder
Decoder
[464, 13779, ..., 13]
The cute green dragon trotted into the cave.
[28532, 264, ...,293]
=
Der süße grüne Drache trabte in die

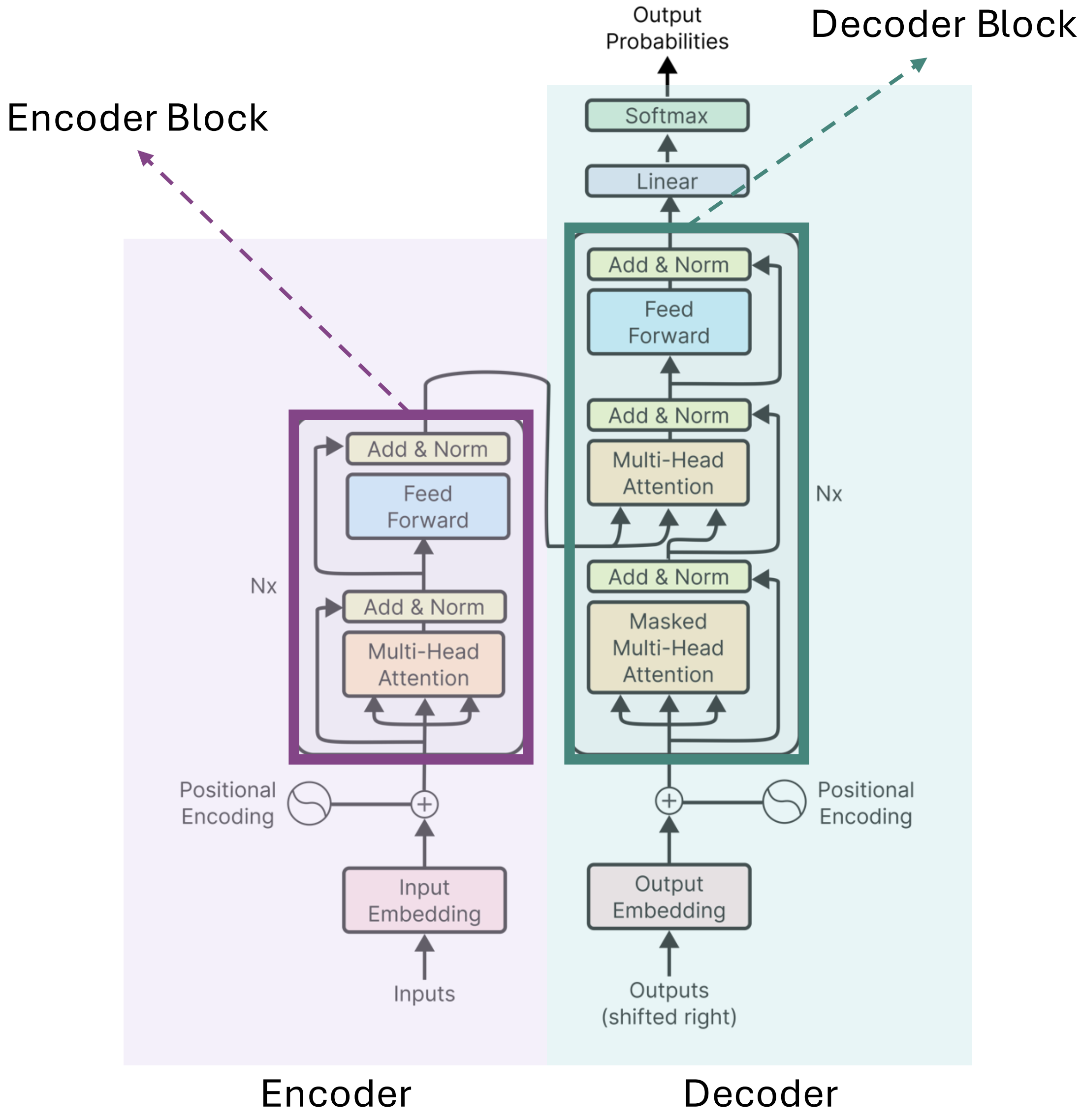


The **encoder** transforms the input sequence into a rich, contextual vector representation that captures the meaning of and relationships between elements.

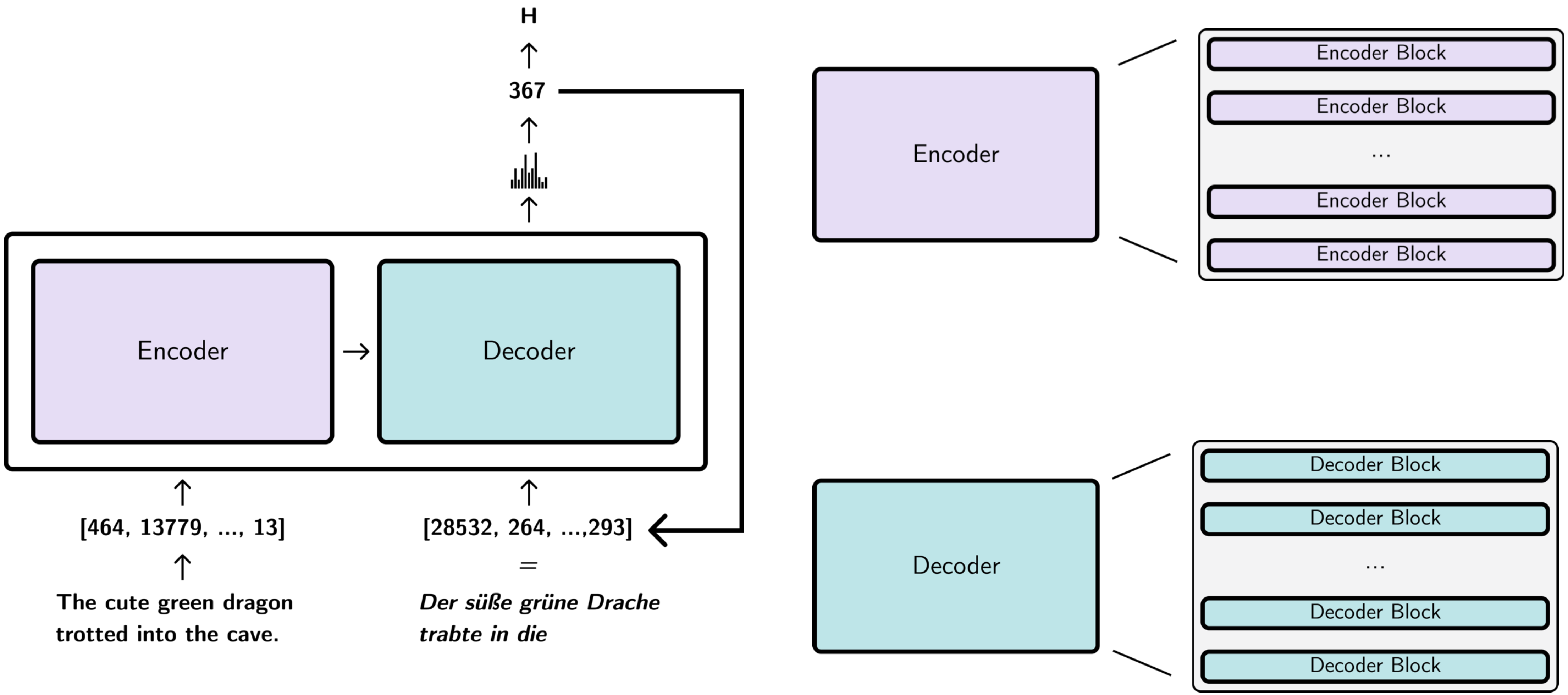
The **decoder** takes the encoded input from the encoder and autoregressively generates an output sequence token by token, using previously generated tokens and the context of the encoder to produce a transformed sequence (e.g. a translation or summary).

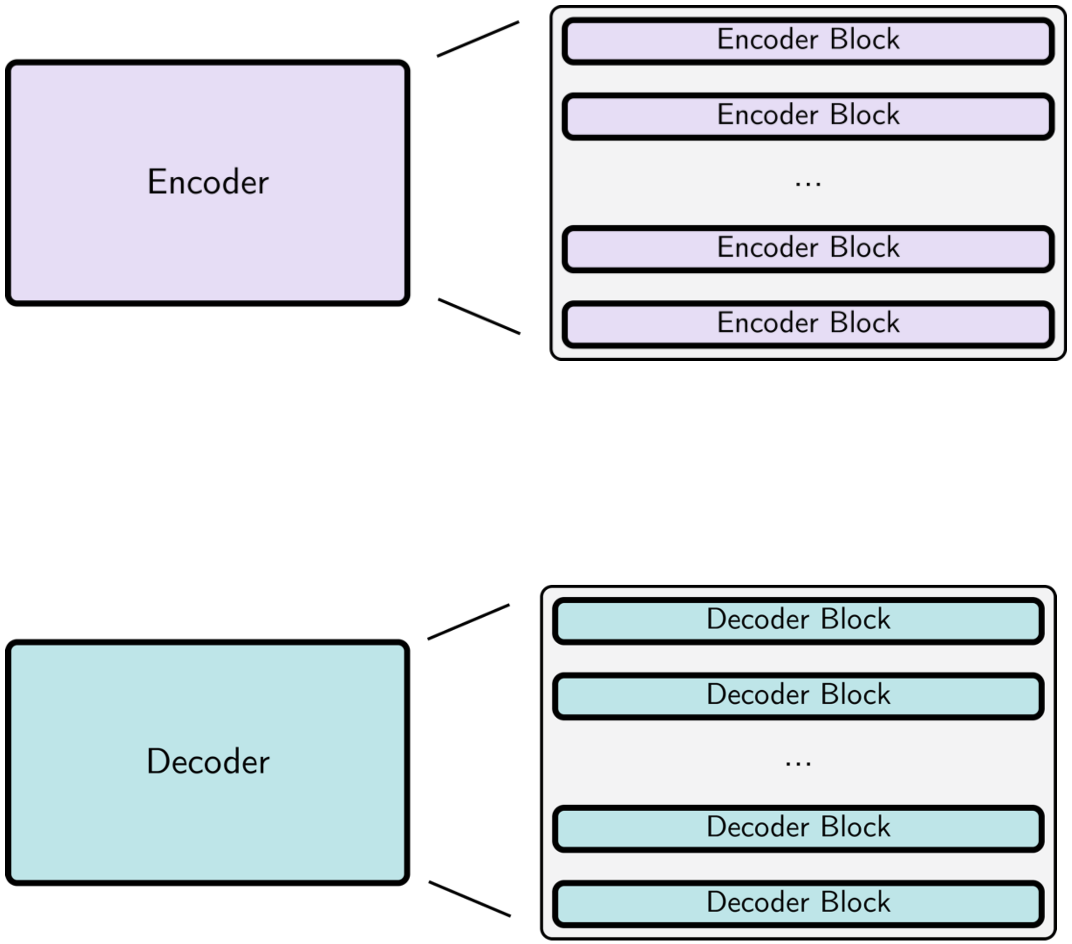


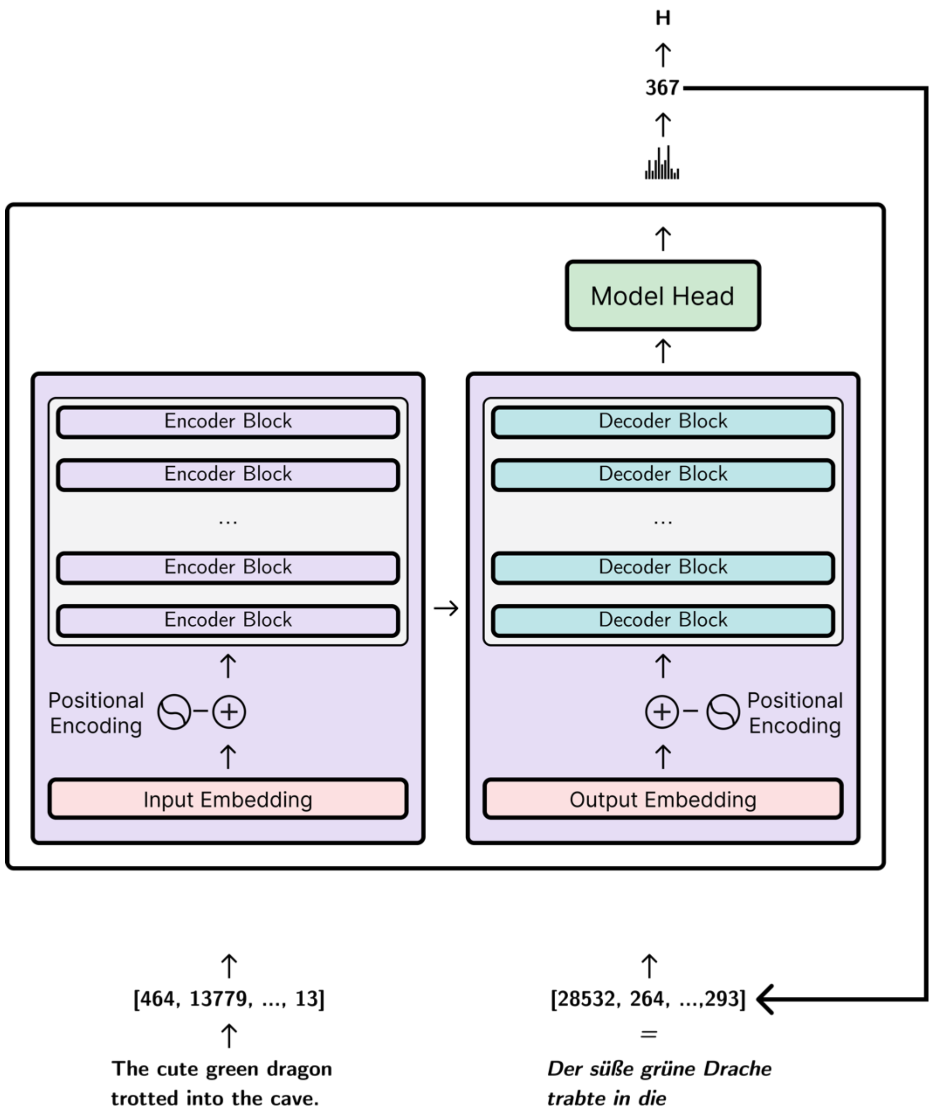
H
367
Encoder
Decoder
[464, 13779, ..., 13]
[28532, 264, ...,293]
=
The cute green dragon
trotted into the cave.
Der süße grüne Drache
trabte in die

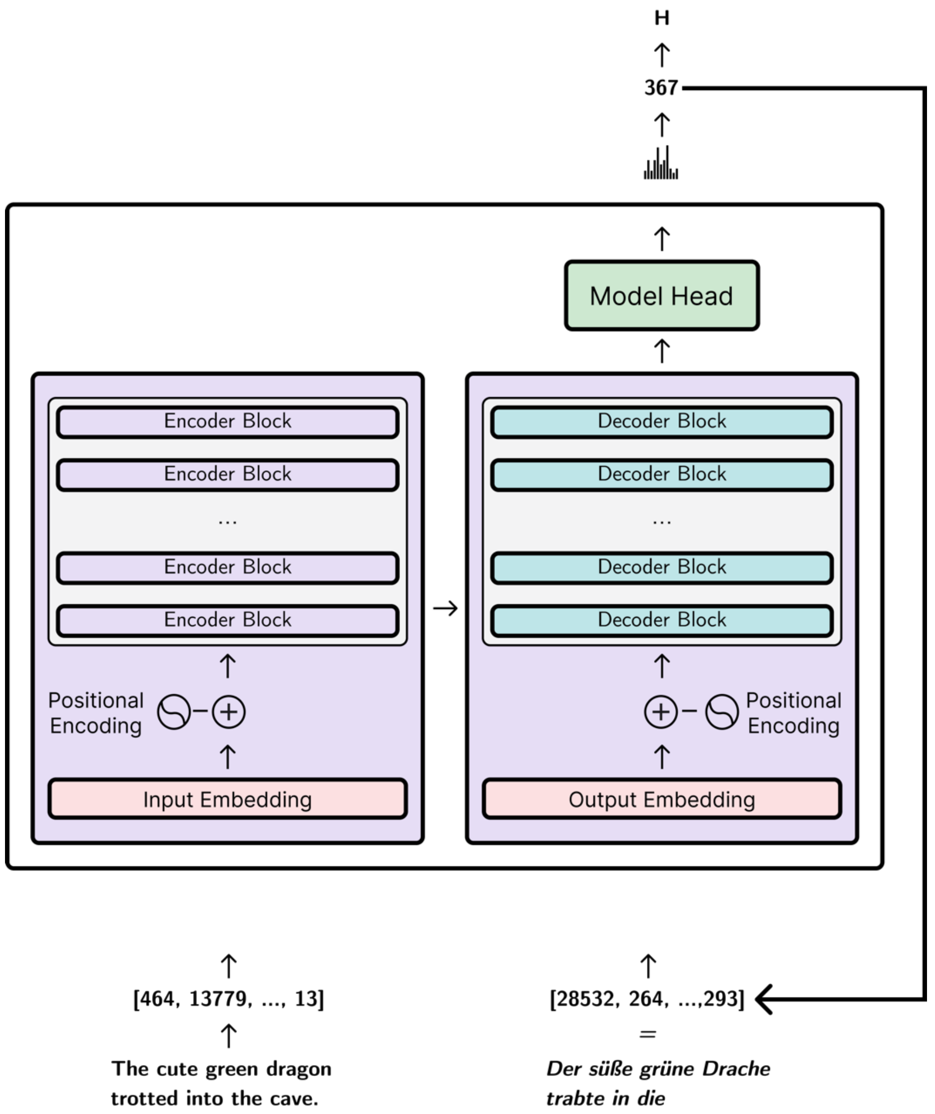


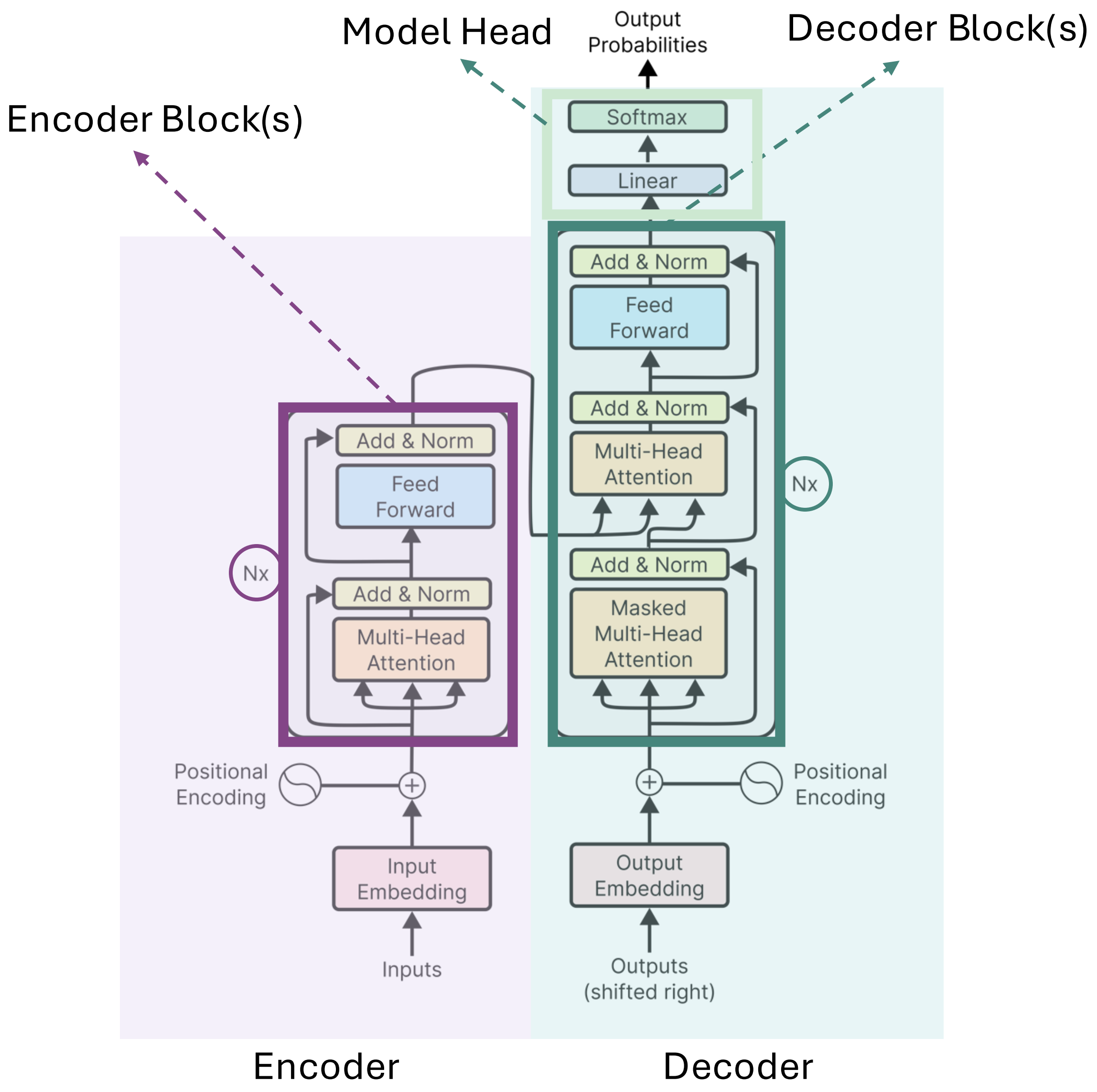
Output
Probabilities
Decoder Block
Encoder Block
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Add & Norm
Masked
Multi-Head
Attention
Nx
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Positional
Encoding
Positional
Encoding
Input
Embedding
Output
Embedding
Inputs
Outputs
(shifted right)
Encoder
Decoder


H
367
Encoder
Decoder
[464, 13779, ..., 13]
The cute green dragon trotted into the cave.
[28532, 264, ...,293]
=
Der süße grüne Drache trabte in die
Encoder
Encoder Block
Encoder Block
...
Encoder Block
Encoder Block
Decoder
Decoder Block
Decoder Block
...
Decoder Block
Decoder Block


Encoder
Encoder Block
Encoder Block
...
Encoder Block
Encoder Block
Decoder
Decoder Block
Decoder Block
...
Decoder Block
Decoder Block


H
367
Model Head
Encoder Block
Encoder Block
...
Encoder Block
Encoder Block
Positional Encoding
Input Embedding
Decoder Block
Decoder Block
...
Decoder Block
Decoder Block
Positional Encoding
Output Embedding
[464, 13779, ..., 13]
The cute green dragon trotted into the cave.
[28532, 264, ...,293]
=
Der süße grüne Drache trabte in die


H
367
Model Head
Encoder Block
Encoder Block
...
Encoder Block
Encoder Block
Decoder Block
Decoder Block
...
Decoder Block
Decoder Block
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
[464, 13779, ..., 13]
[28532, 264, ...,293]
=
The cute green dragon trotted into the cave.
Der süße grüne Drache trabte in die


Model Head
Output Probabilities
Decoder Block(s)
Encoder Block(s)
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)
Encoder
Decoder

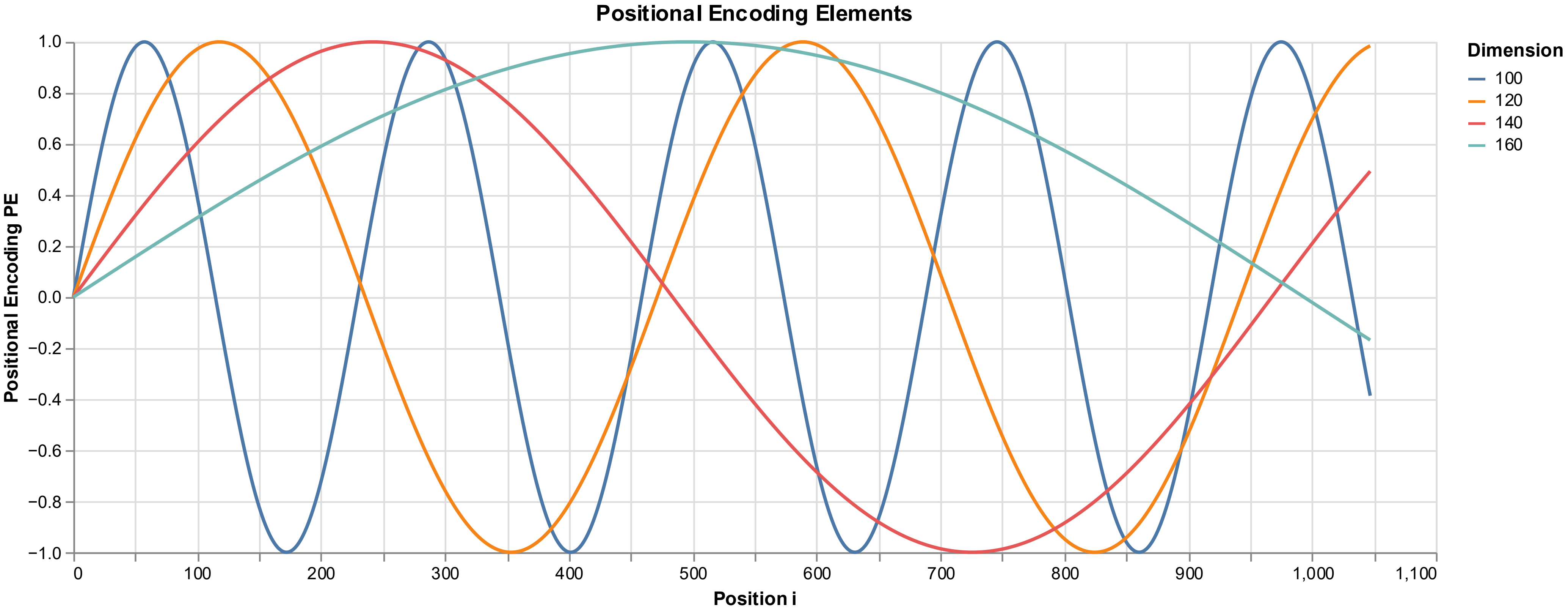
Positional Encoding

The embedding layer and subsequent layers are inherently permutation invariant, but the order plays a critical role in sequence processing tasks.

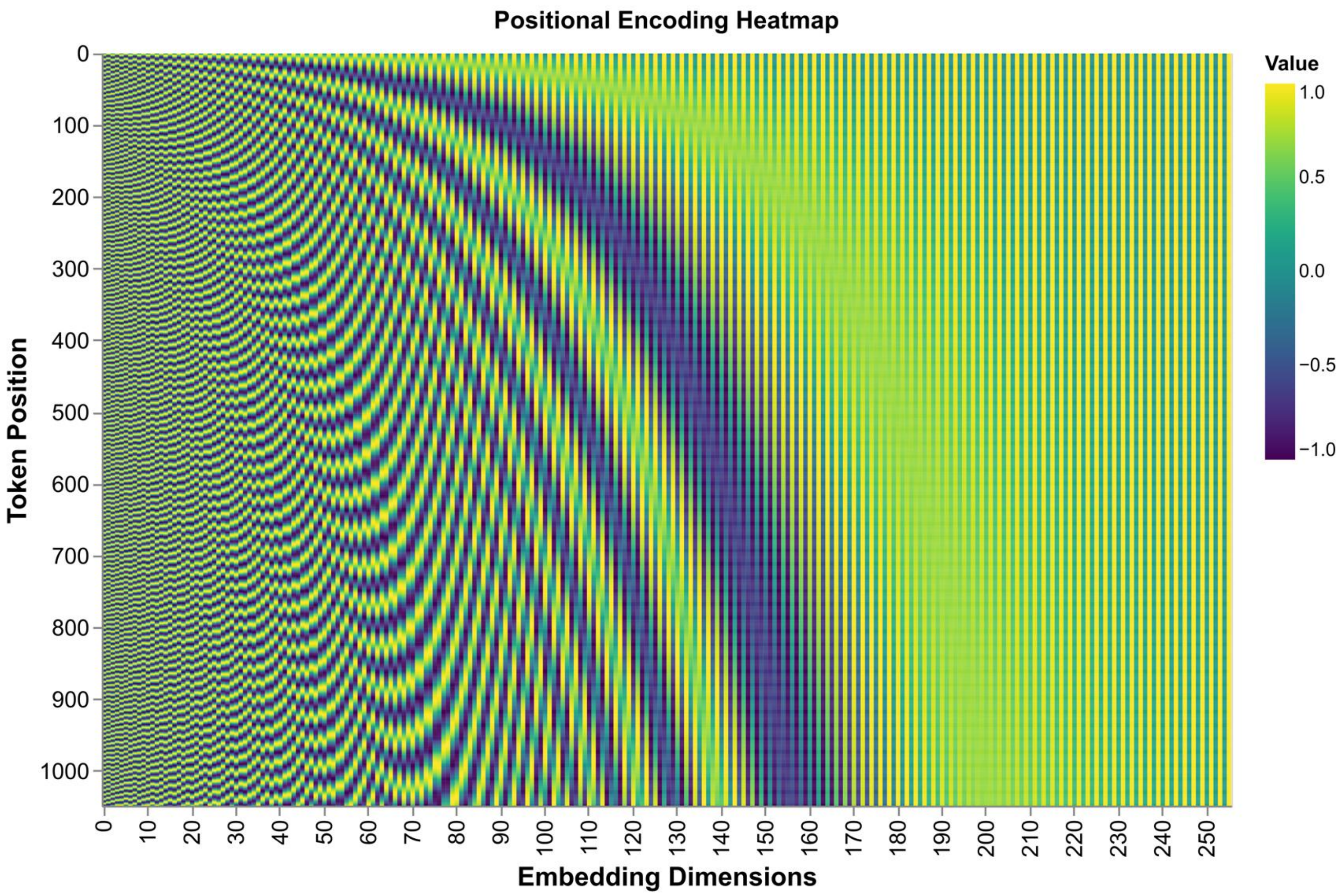
- $PE(pos, 2_i) = \sin(pos/10000^{2i/d_{model}})$
- $PE(pos, 2_{i+1}) = \cos(pos/10000^{2i/d_{model}})$

$$\overrightarrow{PE} = \begin{bmatrix} \sin(pos/10000^{0/d_{model}}) \\ \cos(pos/10000^{0/d_{model}}) \\ \sin(pos/10000^{2/d_{model}}) \\ \cos(pos/10000^{2/d_{model}}) \\ . \\ . \\ . \\ \sin(pos/10000^{d_{model}-2/d_{model}}) \\ \cos(pos/10000^{d_{model}-2/d_{model}}) \end{bmatrix}$$

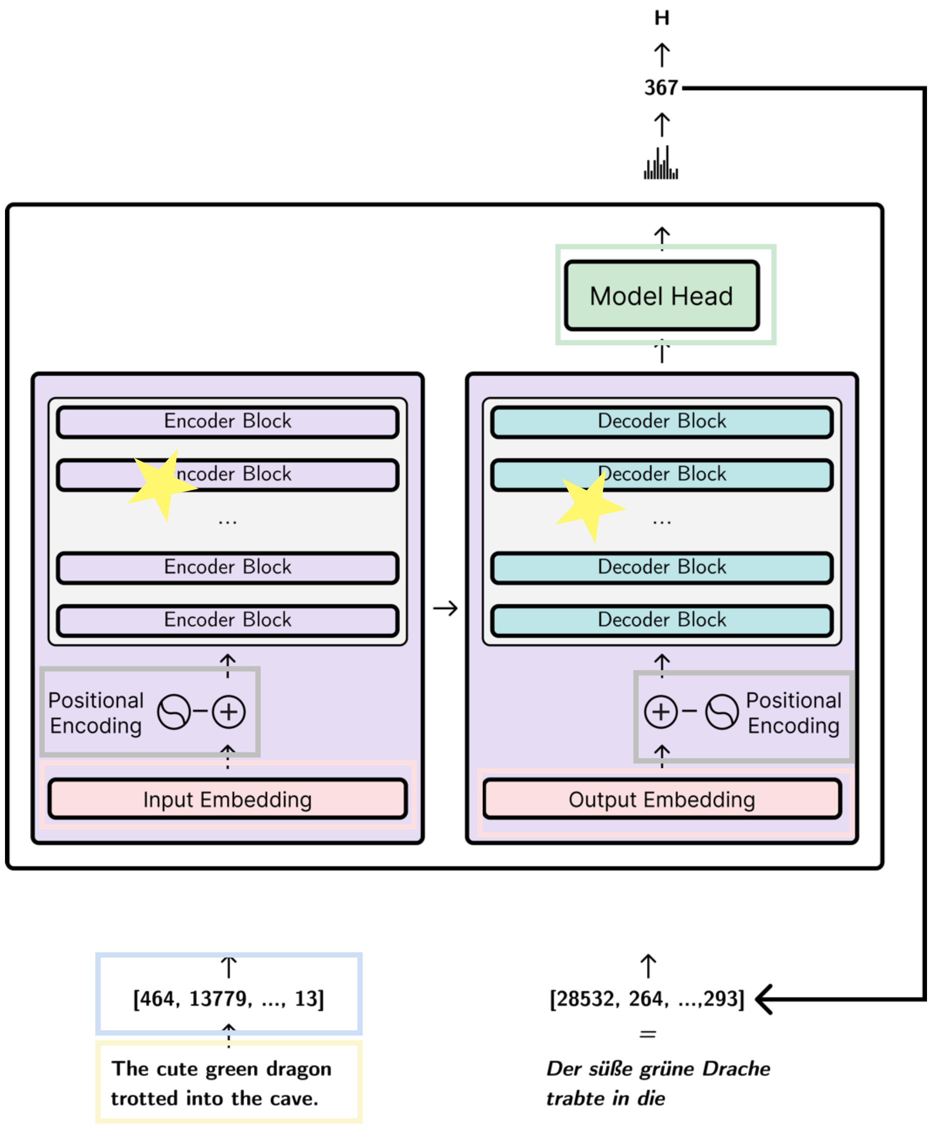
- $PE(pos, 2_i) = \sin(pos/10000^{2i/d_{model}})$
- $PE(pos, 2_{i+1}) = \cos(pos/10000^{2i/d_{model}})$



- $PE(pos, 2_i) = \sin(pos/10000^{2i/d_{model}})$
- $PE(pos, 2_{i+1}) = \cos(pos/10000^{2i/d_{model}})$



Recap



Tokenisation breaks down natural language sequences into atomic units that carry some semantic meaning (tokens).

Encoding represents tokens in a one-dimensional numeric space (token IDs)

Embeddings project token IDs in higher dimensional vector space

Positional encoding injects information about a tokens' sequence-position into the embedding vector

Some magic yet to be covered

The decoder autoregressively generates a probability distribution over the vocabulary.