

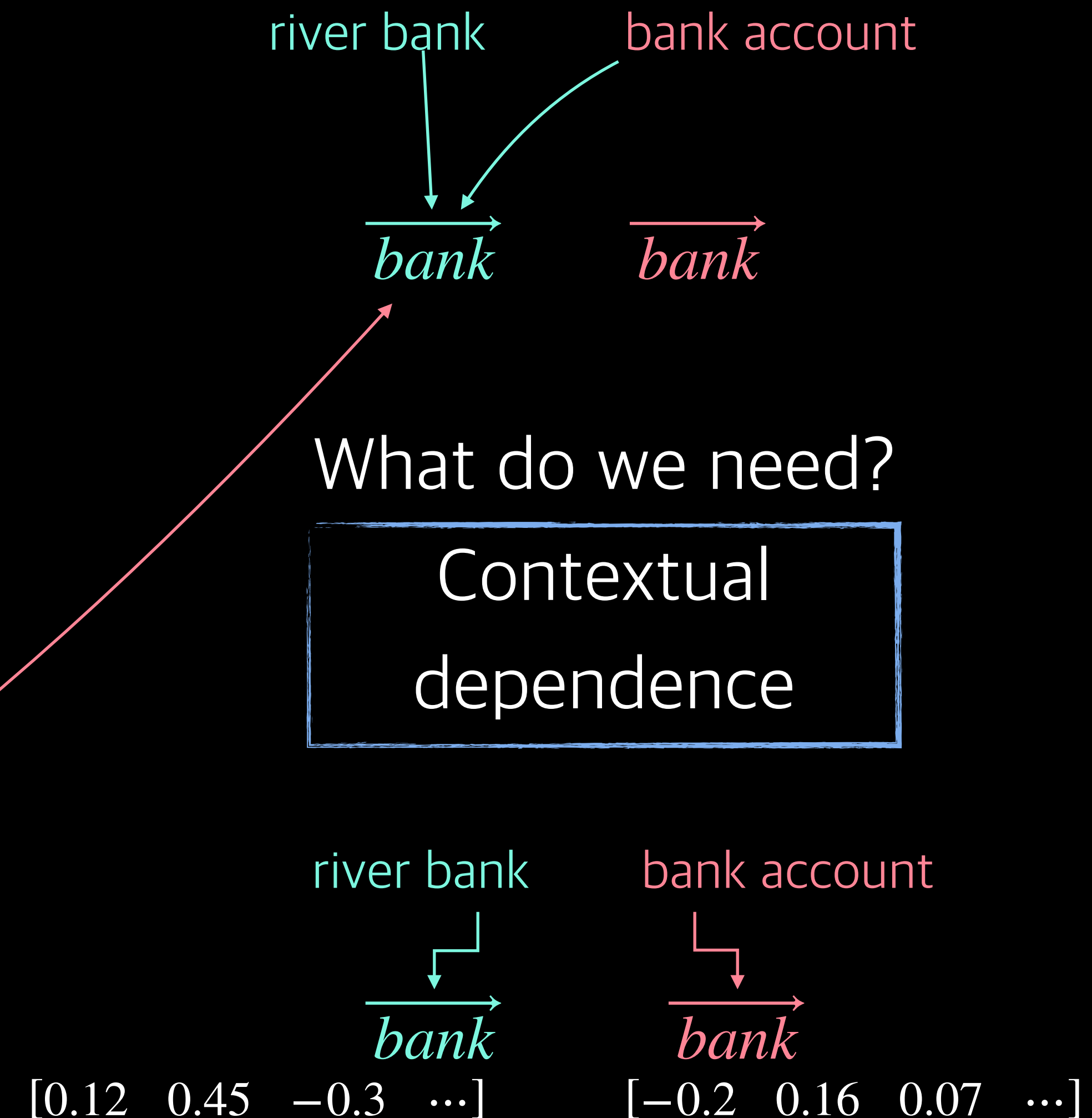
Text Analytics

Transformers – BERT and BART

Ramaseshan Ramachandran

WORD EMBEDDINGS

- ★ Word embeddings are the most fundamental to NLP
- ★ **Capturing meaning**: Represents meaning of a word numerically
- ★ **Dimensionality reduction**: Projects words onto a much lower-dimensional space
- ★ **Semantic relationships**: Captures semantic relationships like synonymy, antonymy and semantically similar words
- ★ **Context free**: Captures the meaning based on co-occurrence statistics



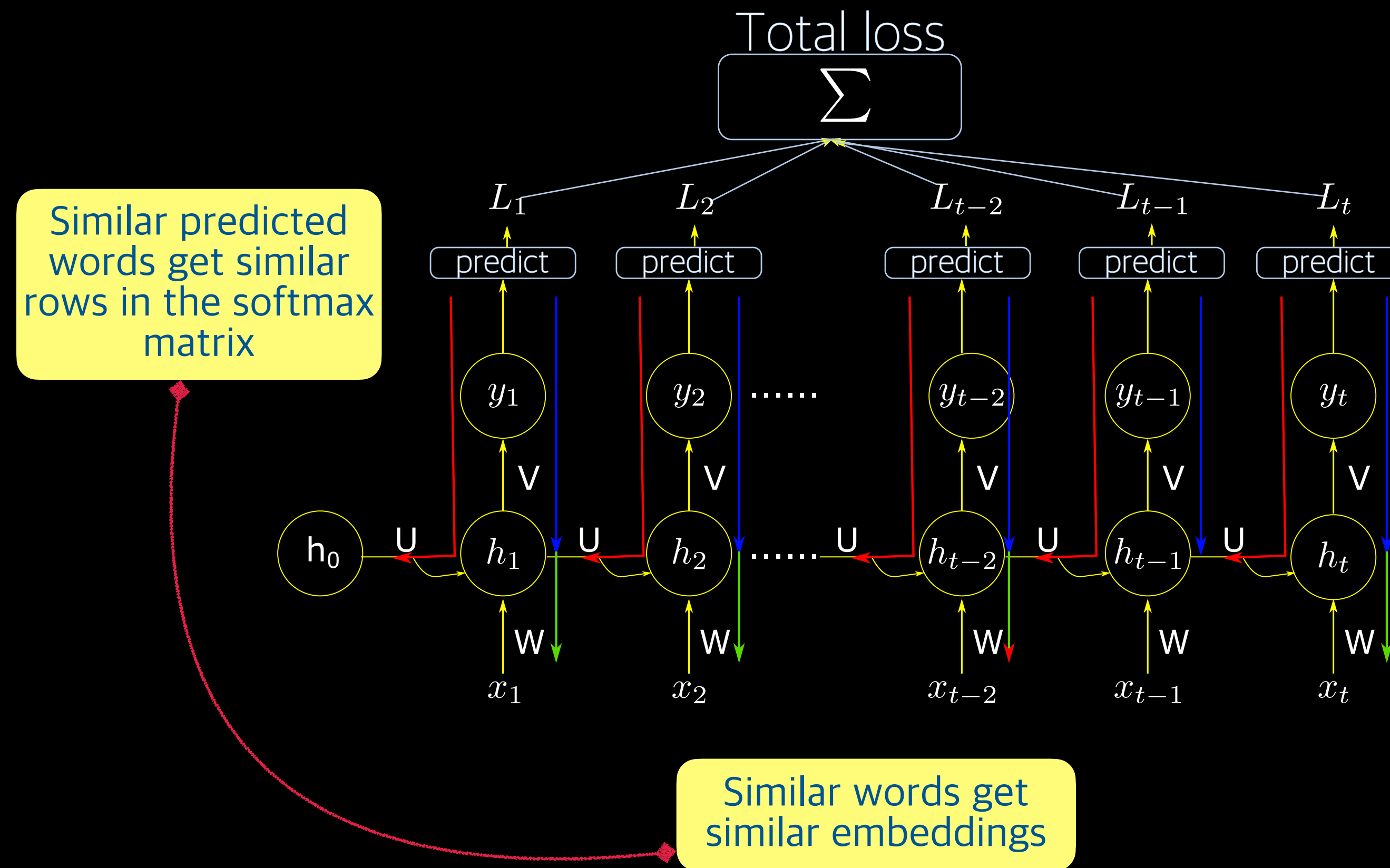
LANGUAGE MODELS

Dependency Type	Dependency Structure	Models	
Independent Prediction	Does not depend on any token	Unigram	$P(X) = \prod_{i=1}^{ V } P(x_i)$
Markov (Left2Right)	Depends on a fixed number of previous tokens	N-gram	$P(X) = \prod_{i=1}^{ V } P(x_i x_{i-n+1}, \dots, x_{i-1})$
Unidirectional autoregressive	Depends on the previous tokens	RNN, GPT	$P(X) = \prod_{i=1}^{ X } P(x_i x_1, \dots, x_{i-1})$
Bidirectional context	Left2Right Autoregressive	BERT	$P(X) = \prod_{i=1}^{ X } P(x_i x_{\neq i})$

BIDIRECTIONAL LANGUAGE MODEL

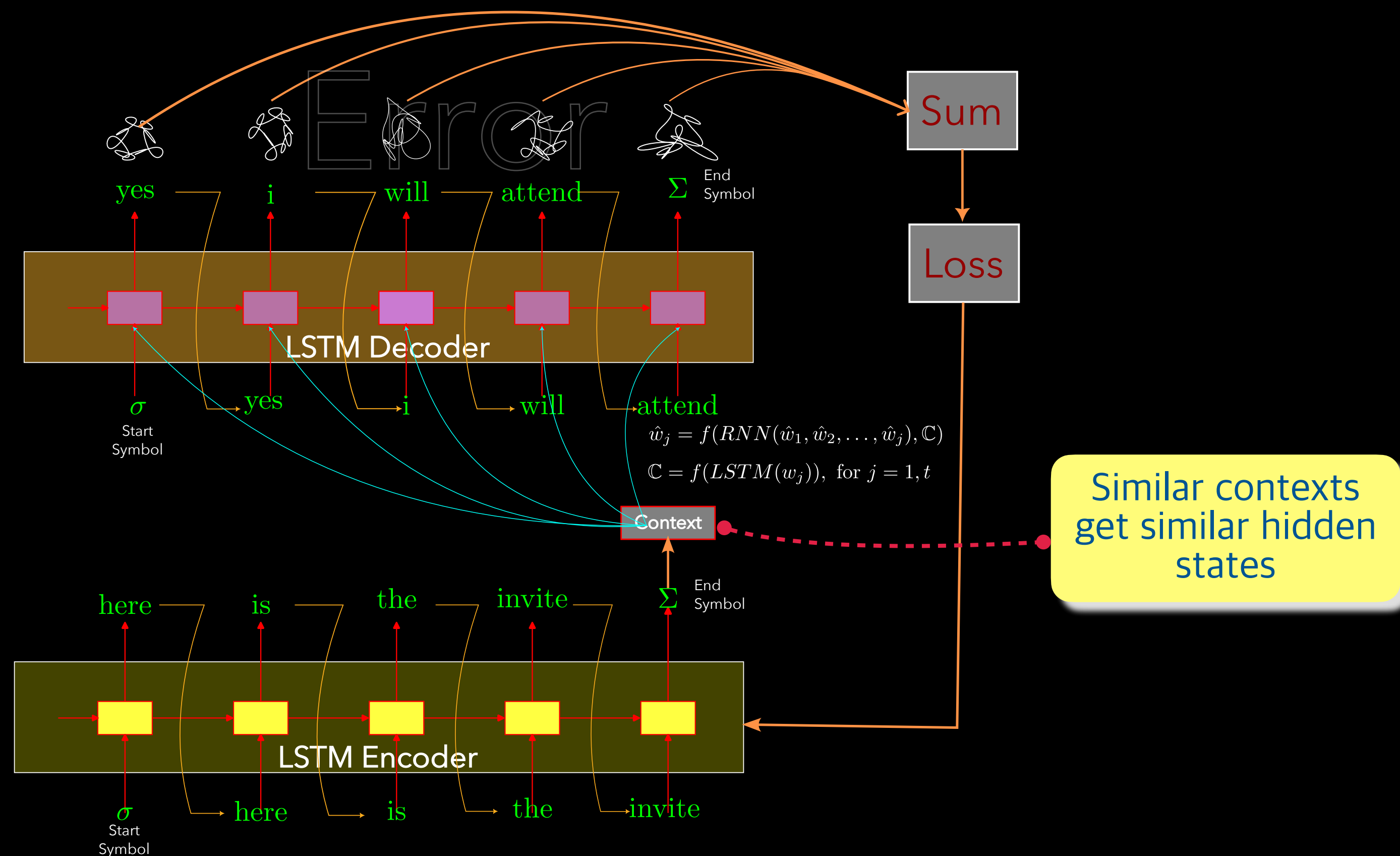
- ✦ In the forward LSTM with L layers, $\vec{x}_{k,L}$ is the L^{th} layer output is the representation of the word embedding \vec{x}_k
- ✦ In the reverse LSTM with L layers, $\overleftarrow{x}_{k,L}$ is the L^{th} layer output presentation of the word embedding \vec{x}_k
- ✦ $f(\vec{x}_{k,L}, \overleftarrow{x}_{k,L})$ at the L^{th} layer is used to predict the next word
- ✦ $\vec{x}_{k,0}$ is the word embedding at the token layer

RNN



Parameters are shared, while derivatives are accumulated.

SEQUENCE2SEQUENCE



ELMO

The weights, s^{task} , in the linear combination are learned for each task and normalized by softmax

Context sensitive

- ELMo word representations are functions of the entire input sentence

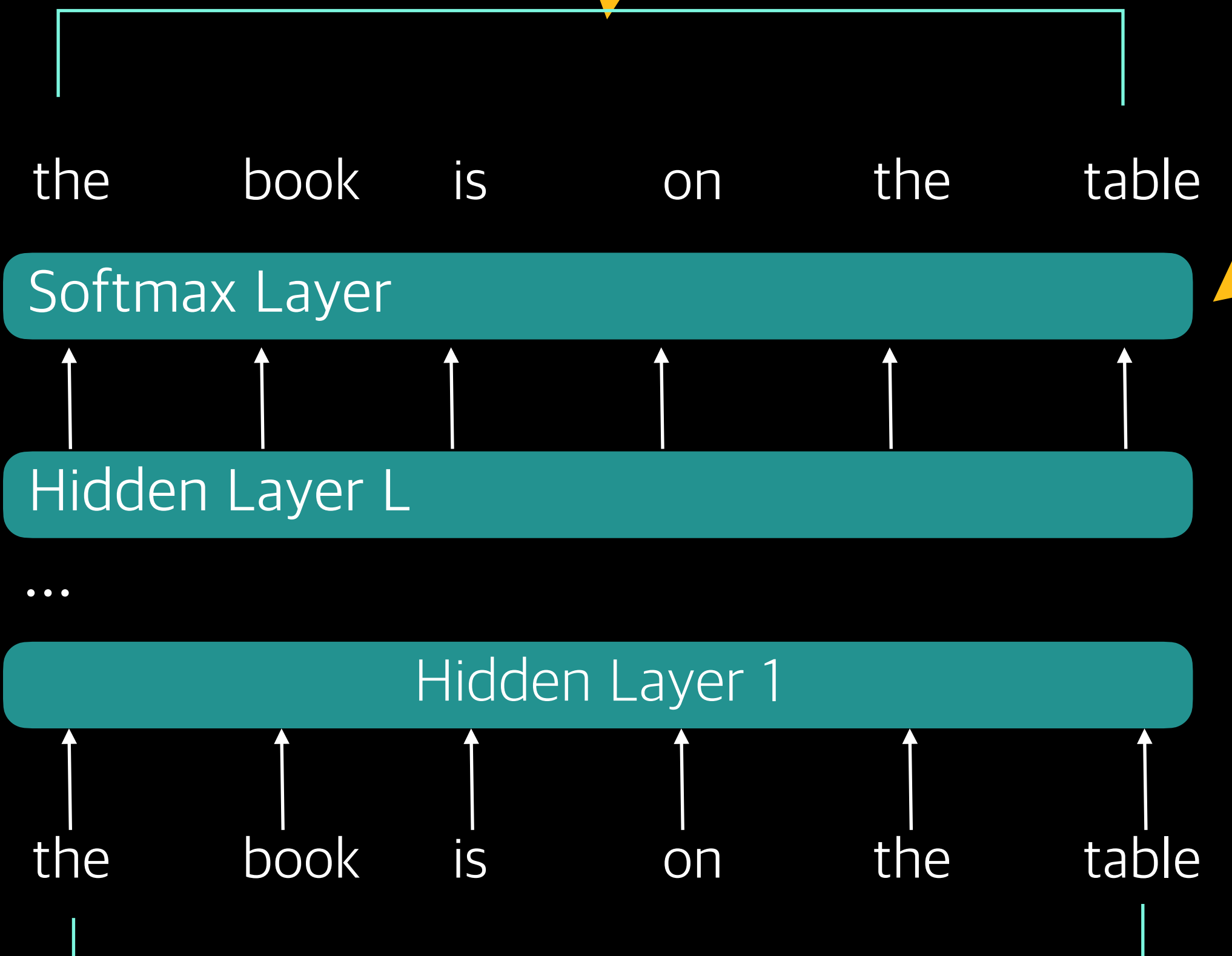
$$\forall k \in V, R_k = \vec{x}_{k,0}, \vec{x}_{k,j}, \vec{x}_{k,j}^{\leftarrow}, \text{ for } j = 1, \dots, L \quad \Phi_s$$

$$= \vec{x}_{k,j} \text{ for } j = 0, \dots, L$$

- For every token, there is a new representation

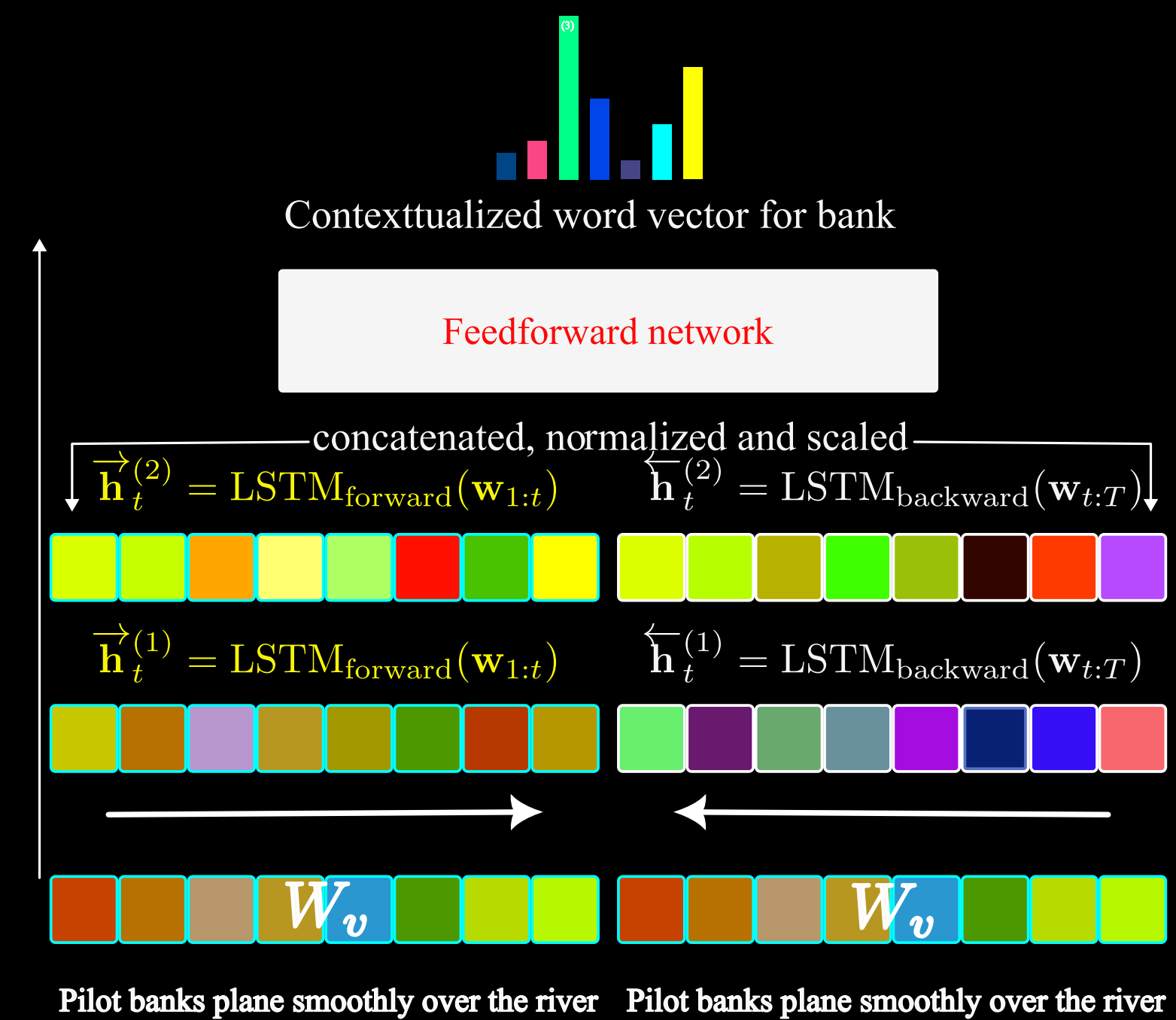
Context independent

$$ELMO_k^{task} = E(R_k; \Phi^{task}) = \gamma^{task} \sum_{j=0}^L s_j^{task} h_{k,j}^{LM}$$



Pre-trained

EMBEDDING FROM LANGUAGE MODEL(ELMO)



Component	Description
Input	Raw tokens
Layer 1	Input Embedding
Layer 2	Forward
Layer 3	Backward
Output	Contextualized word embedding

$$\mathbf{ELMO}_t = \gamma \sum_{l=0}^L s^{(l)} \cdot \left[\vec{\mathbf{h}}_t^{(l)}; \overleftarrow{\mathbf{h}}_t^{(l)} \right]$$

where $s^{(l)}$ are Layer-wise normalized weights using softmax, learned as task-specific parameters, γ is the scaling parameter to adjust the overall magnitude and $\left[\vec{\mathbf{h}}_t^{(l)}; \overleftarrow{\mathbf{h}}_t^{(l)} \right]$ are the concatenated hidden states from forward and backward LSTMs.

Pilot banks plane to smoothly over the river
The forward LSTM processes "bank" in the context of "Pilot," while the backward LSTM processes it in the context of "plane smoothly over the river." This dual context helps disambiguate "bank" as a plane maneuver rather than a financial institution

Reference: Deep Contextualized Word Representations

SA USING ELMO

Step 1: Data Preparation

Tokenization: Convert raw text into tokens (words or subwords).
Labeling: Assign sentiment labels (e.g., positive, negative, neutral) to each sentence or document

Step 2: ELMo Embeddings

Word Embedding: Convert each token into a fixed-size vector using a traditional word embedding model (e.g., Word2Vec, GloVe)
Bidirectional LSTM:
 Forward LSTM: Processes text left-to-right, capturing context from preceding words
 Backward LSTM: Processes text right-to-left, capturing context from following words

For each token, compute: $\mathbf{ELMO}_t = \gamma \sum_{l=0}^L s^{(l)} \cdot [\vec{\mathbf{h}}_t^{(l)}; \overleftarrow{\mathbf{h}}_t^{(l)}]$

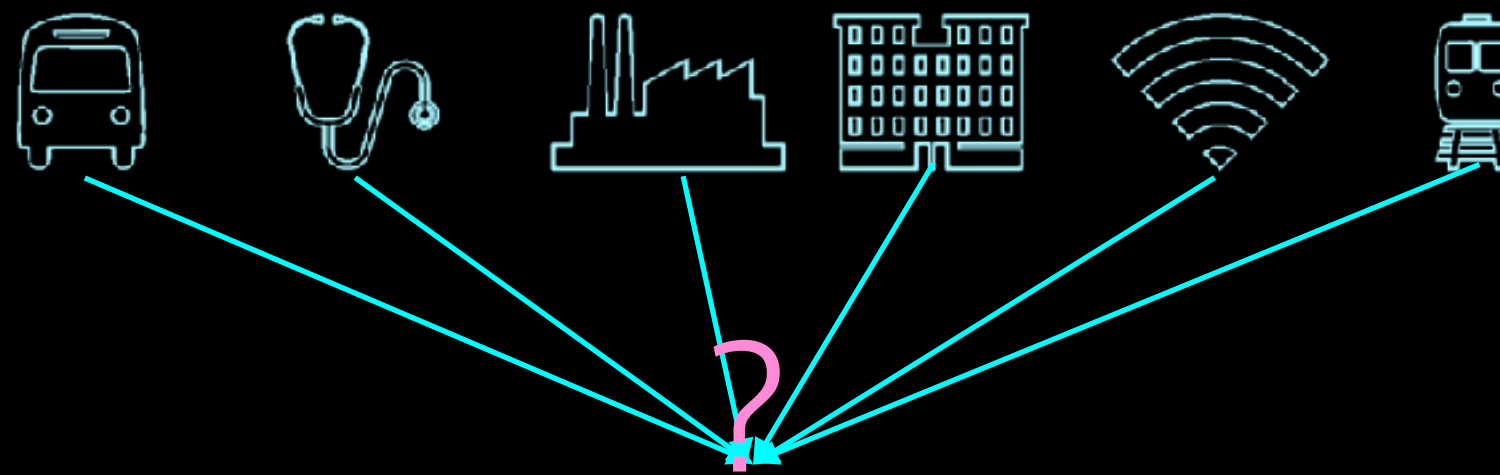
Step 3: Model Architecture

Input: ELMo embeddings for each token in the sentence.
Pooling: Aggregate token embeddings (e.g., mean pooling) to create a fixed-size representation for the entire sentence.
Dense Layer: A fully connected layer to transform the pooled embeddings into a lower-dimensional space.
Softmax: Convert the dense layer output into probabilities for each sentiment class.

Training Process

Input: “The movie was phenomenal!”
ELMo Embeddings: Generate embeddings for each token.
Pooling: Average the embeddings to get a sentence-level representation.
Dense Layer: Transform the pooled embeddings into a 2-dimensional vector (e.g., positive/negative).
Softmax: Convert the dense layer output into probabilities: 0.02, 0.98.
Loss: Compute cross-entropy loss between predicted probabilities and true label.
Update: Backpropagate the loss to update s_j and γ

ATTENTION



“She is employed at a facility that produces iPhones, and it is closer to her house.”

Provide high weights to facility that implies factory
Focus on critical information from the context

“facility” gets attention weights based on the current context highlight it as a manufacturing unit even if it was created using a different context earlier. It enables models to focus on critical information while filtering out noise, mimicking human-like prioritization of salient details

Enables understanding of phrases like “it” which refers to “the facility”

Contextual understanding

Long range dependencies

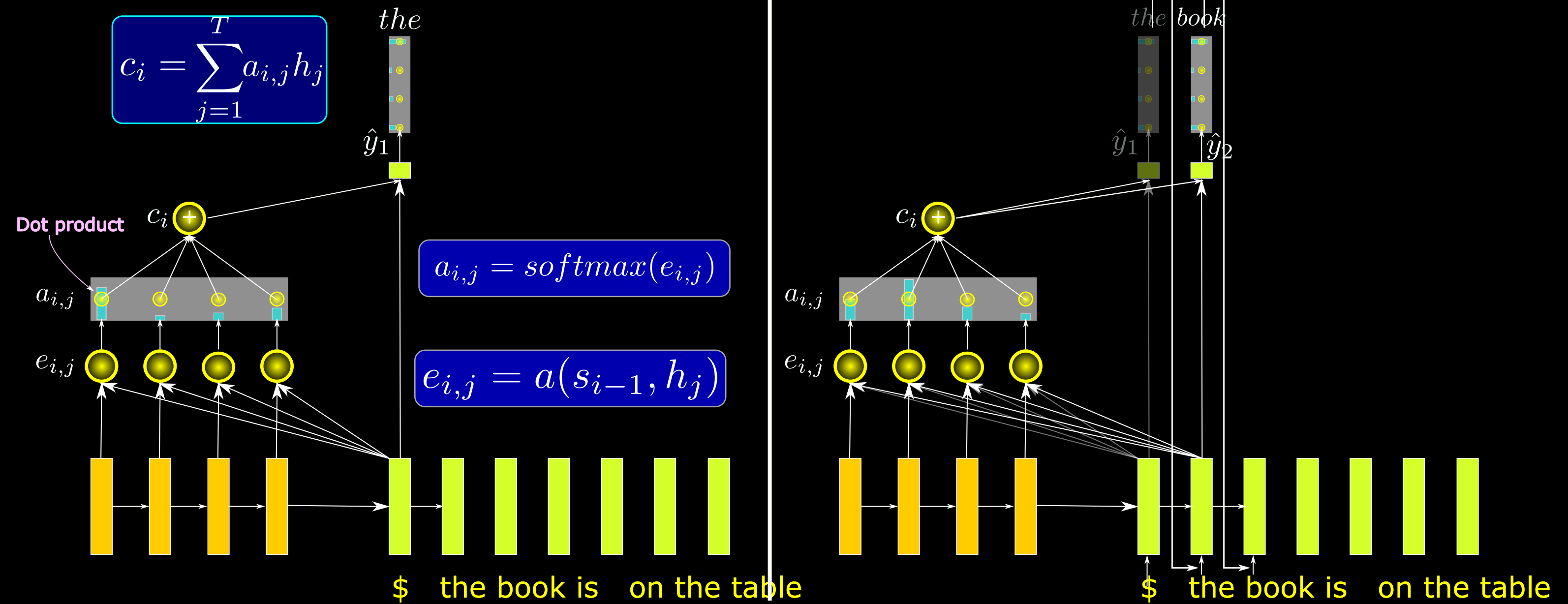
Efficiency

Bahdanau D. Neural machine translation by jointly learning to align and translate. arXiv preprint [arXiv:1409.0473](https://arxiv.org/abs/1409.0473). 2014.

ENCODING AND DECODING

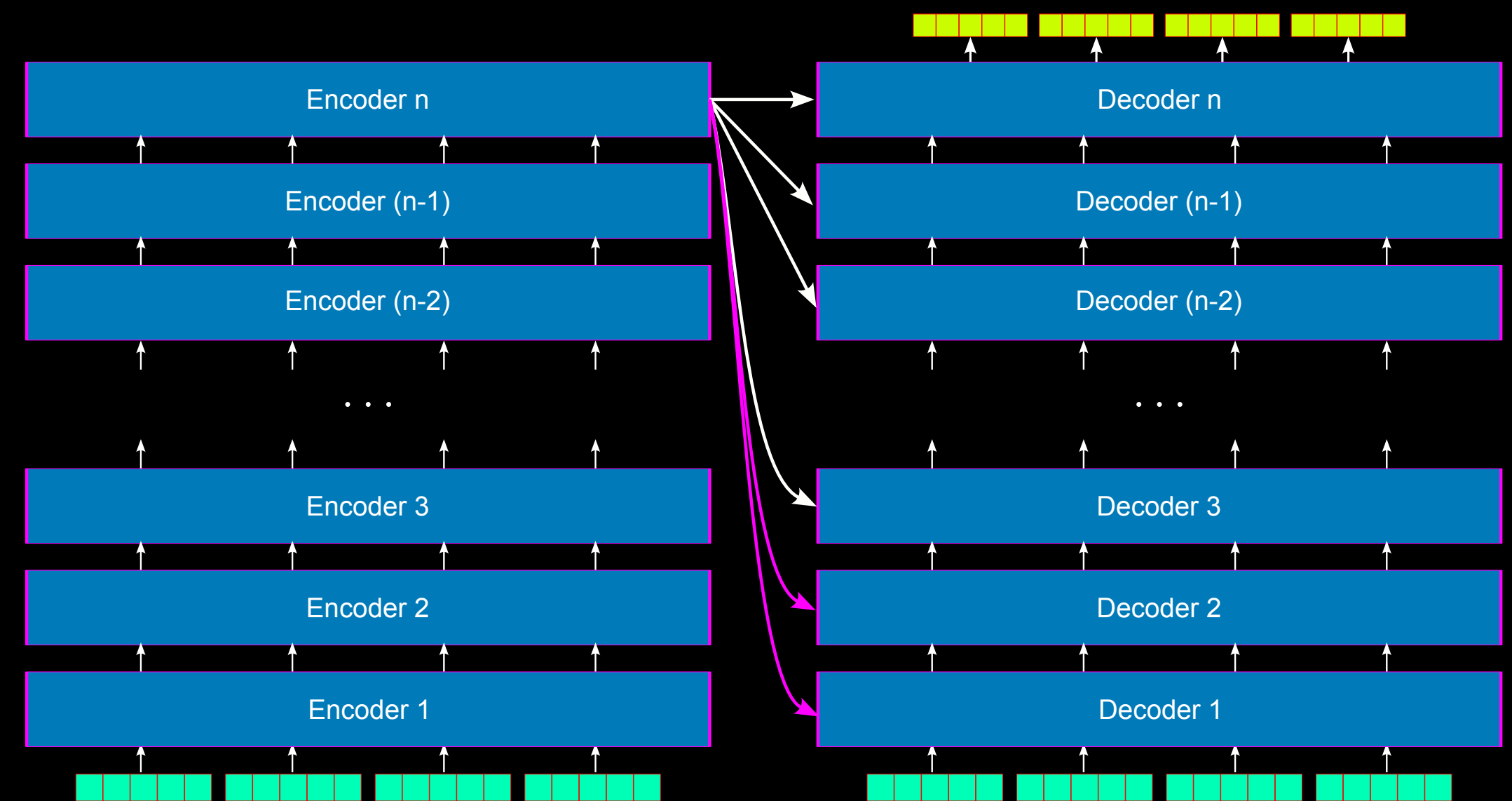
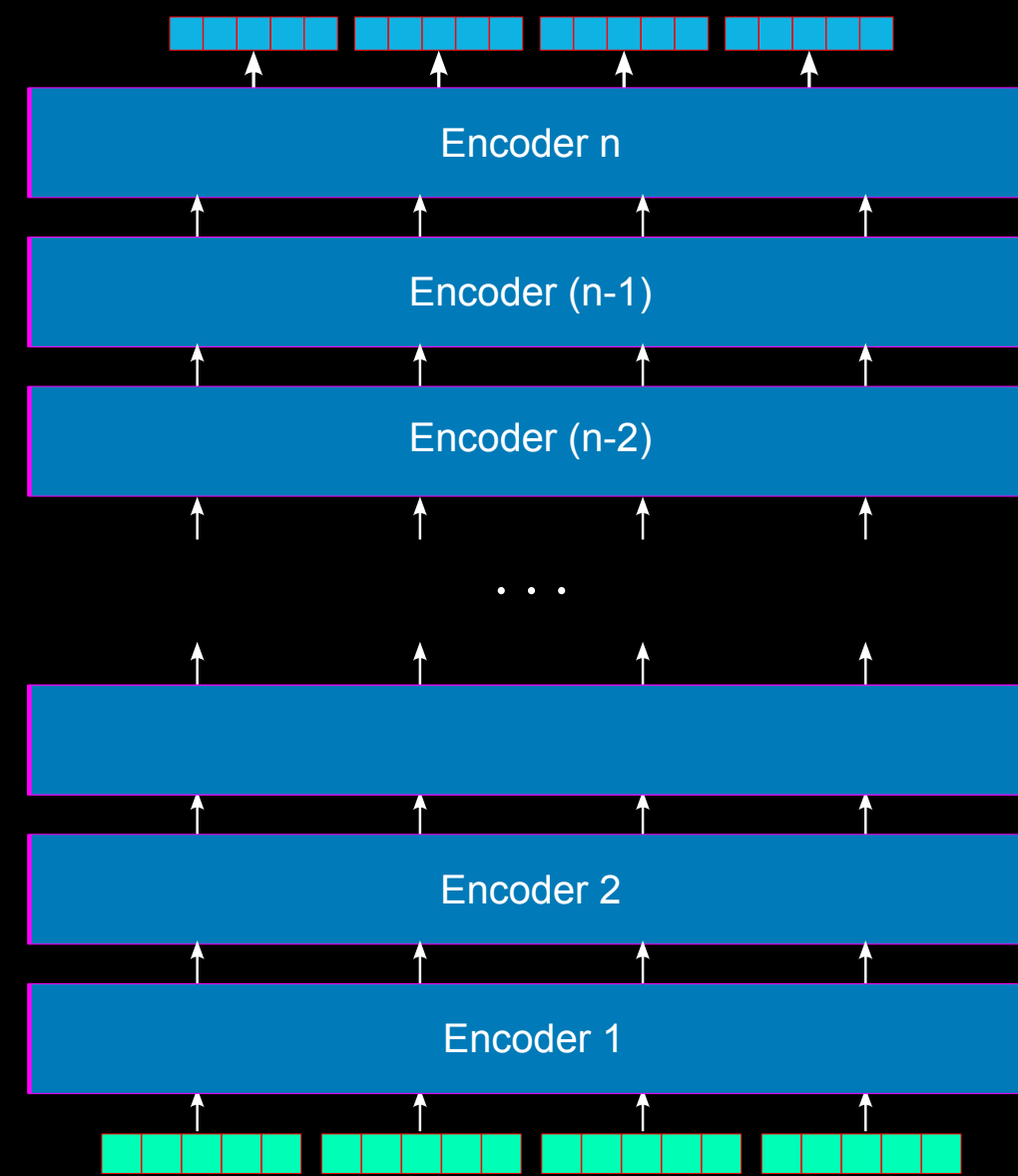
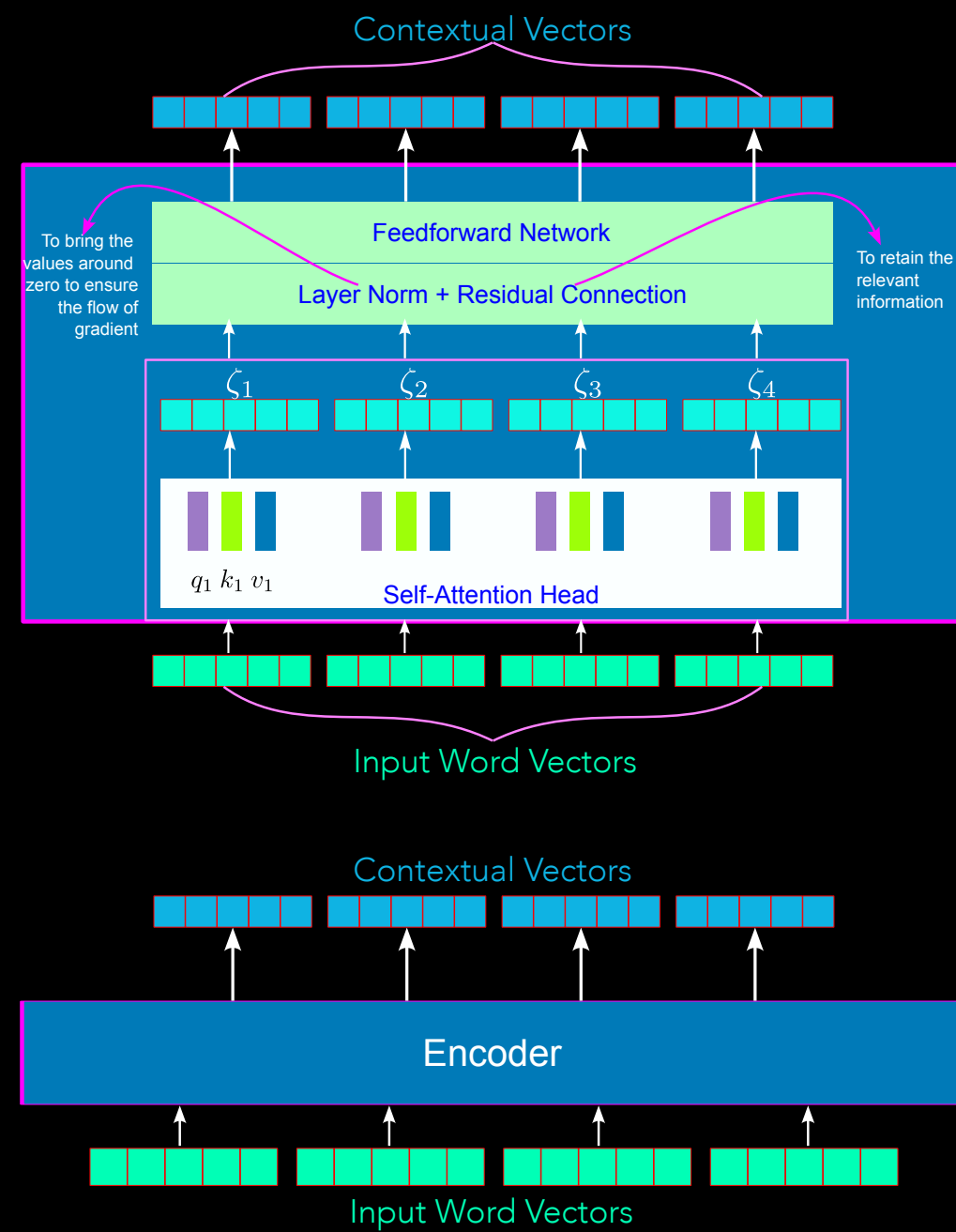
Token Encoding: Encode each token in the sequence into a vector

Attention Mechanism: Perform a linear combination of encoded vectors, weighted by “attention weights”

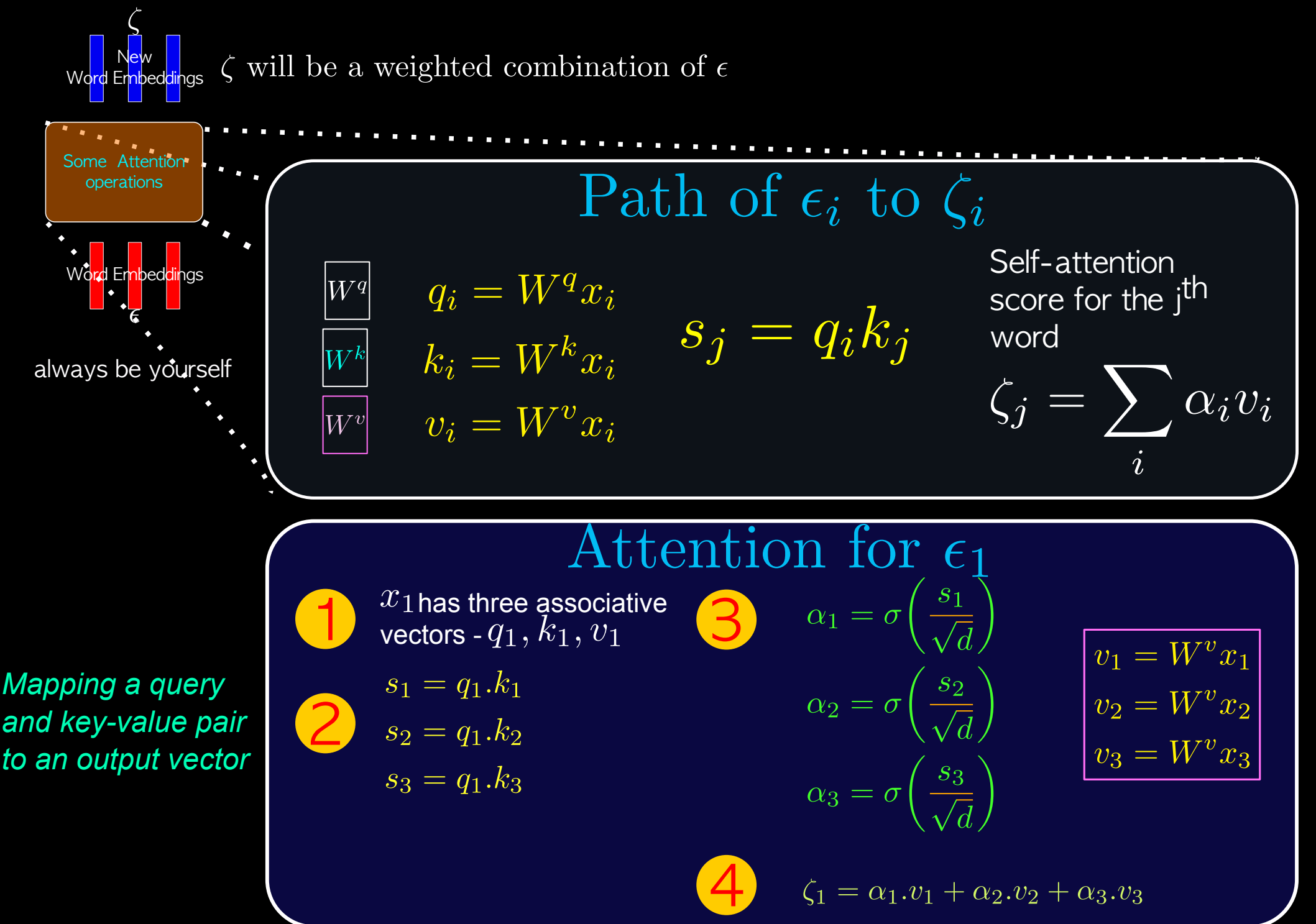
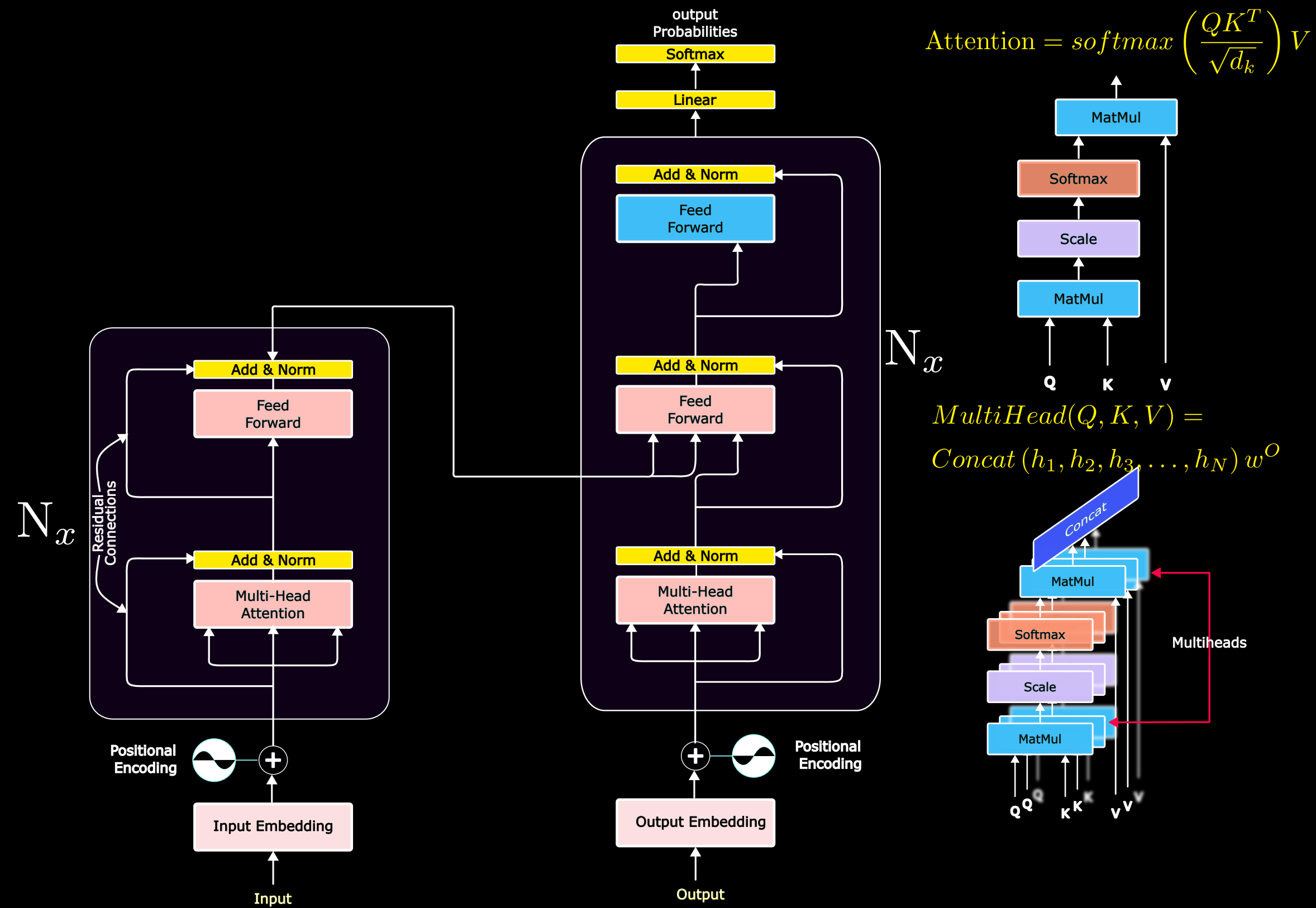


$e_{i,j}$ —attention score
 $a_{i,j}$ —attention distribution
 c_i —attention output

TRANSFORMER



TRANSFORMER



POSITIONAL EMBEDDING

- ✦ Transformers process input tokens in parallel
 - ✦ Transformers process tokens simultaneously
 - ✦ Positional encoding helps differentiate tokens based on their positions
- ✦ Positional embeddings retain positional information
- ✦ PE encodes the order of tokens in a sequence

ABSOLUTE PE

Position of the token in the sentence

- ✦ Sinusoidal functions generate unique positional embeddings

- ✦ $PE(pos, 2i) = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$ for even dimensions of the embedding

- ✦ $PE(pos, 2i + 1) = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$ for odd dimensions

Index of the dimension

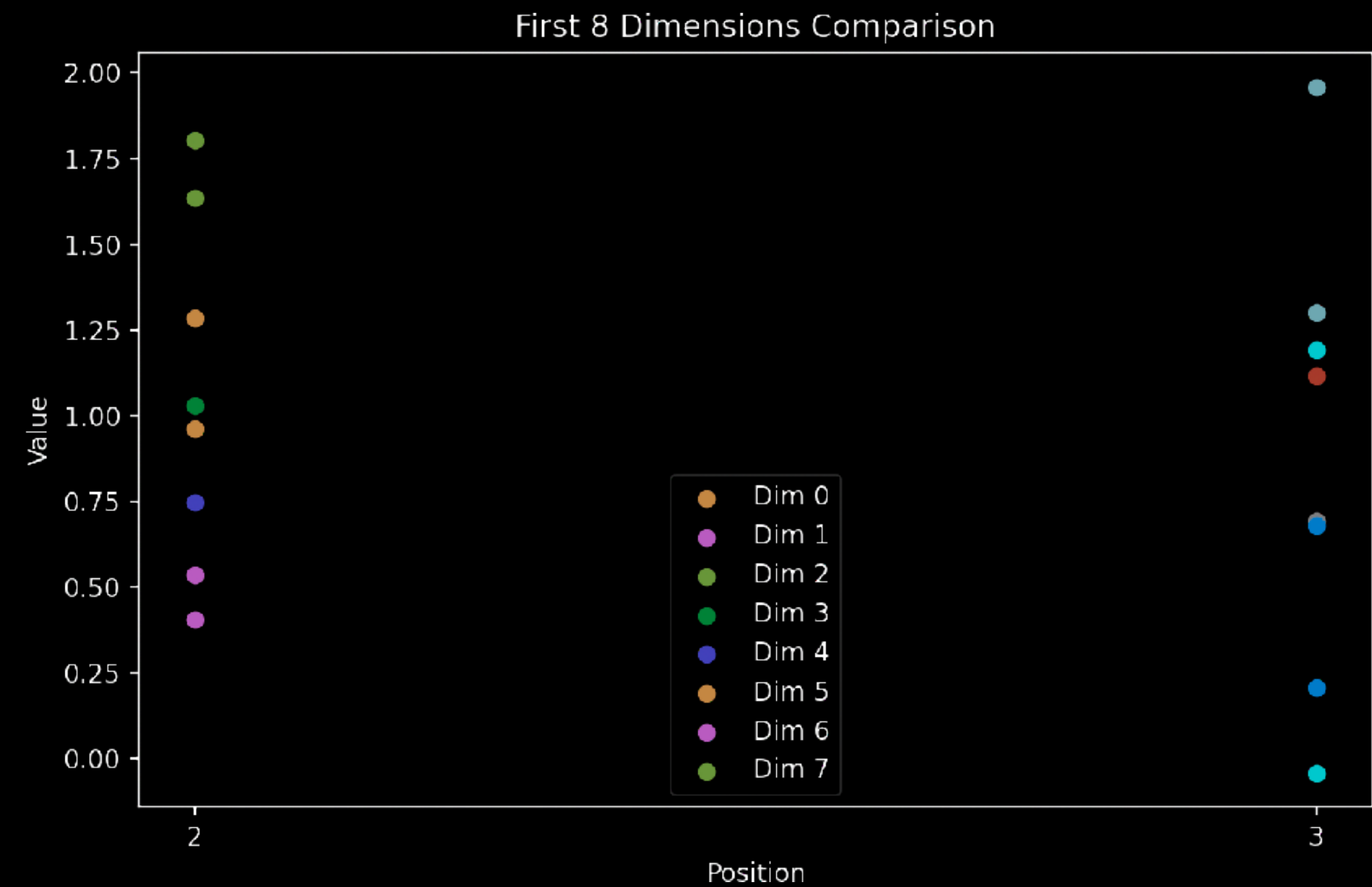
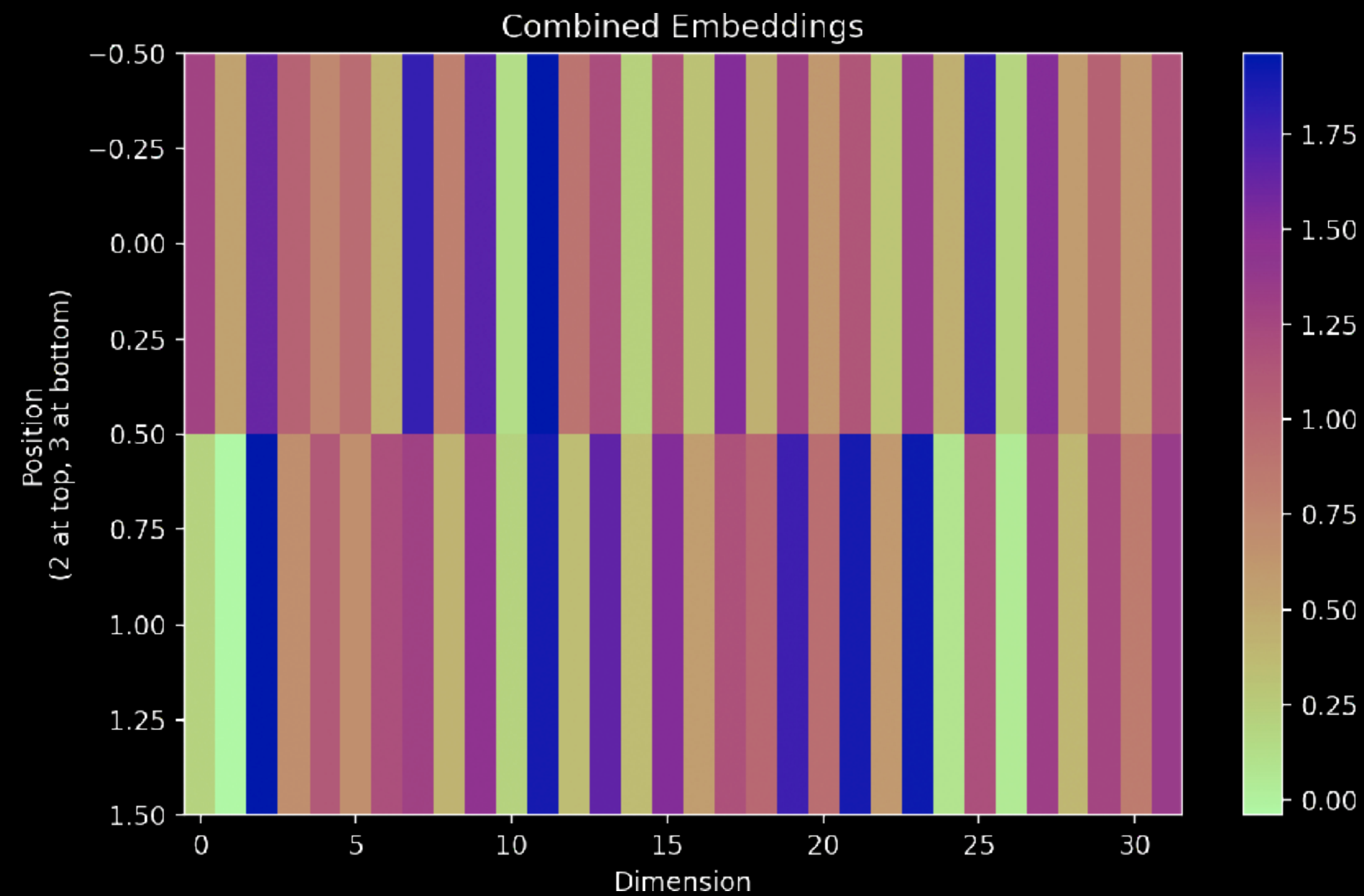
PYTHON CODE FOR PE

```
def positional_encoding(pos, d_model):  
    pe = np.zeros(d_model)  
    for i in range(d_model):  
        if i % 2 == 0: # Even indices: sine  
            pe[i] = np.sin(pos / (10000 ** ((2 * (i//2)) / d_model)))  
        else: # Odd indices: cosine  
            pe[i] = np.cos(pos / (10000 ** ((2 * ((i-1)//2)) / d_model)))  
    return pe
```


PE PROPERTIES



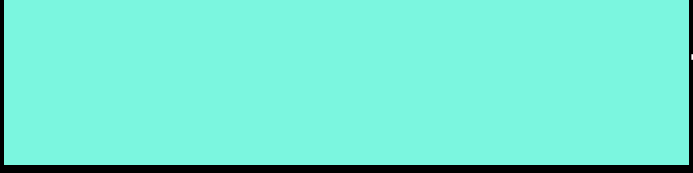
- ✦ Each position has a unique embedding
- ✦ Fixed across different sentences
 - ✦ Position 3 in different sentences shares the same encoding
- ✦ The PE dimension is same as the word embedding size

VISUALIZATION OF THE PE

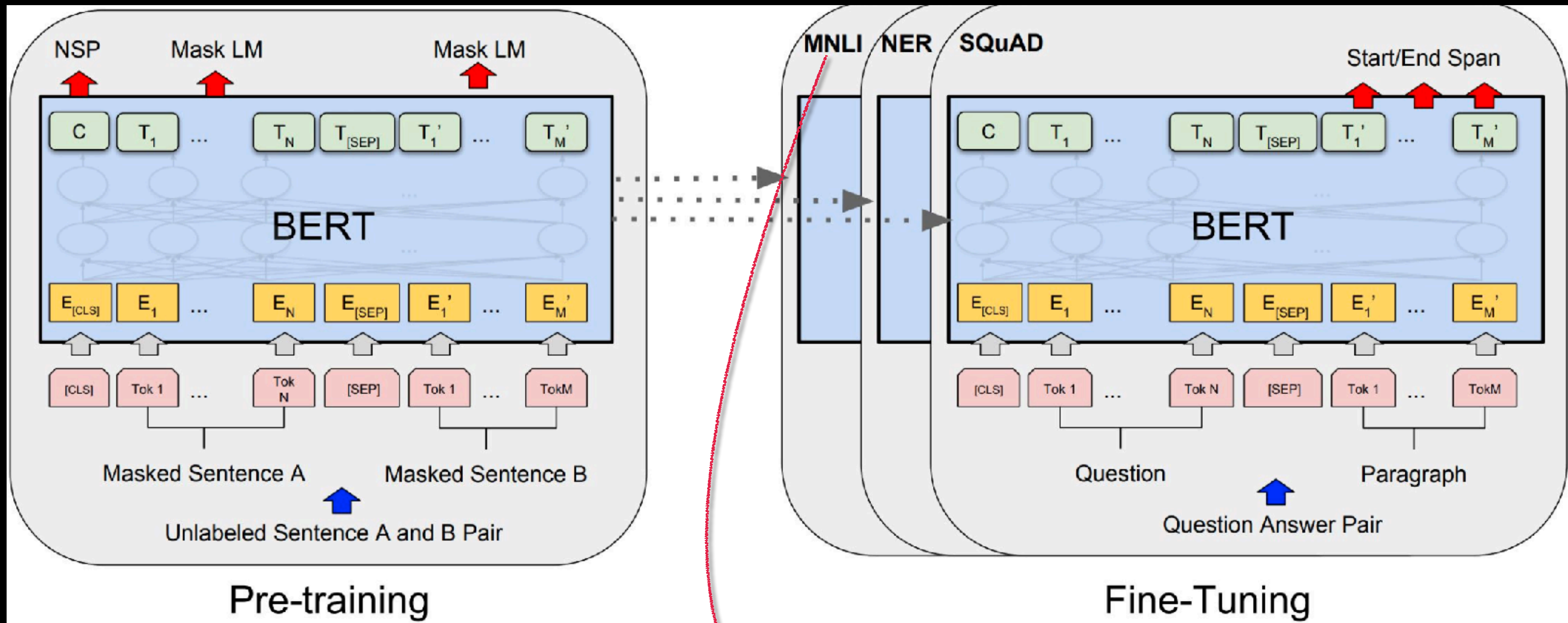


Simulated for a word embedding of dimension
32 for word positions 2 and 3

MASKED LANGUAGE MODEL

- ✦ Mask out $k\%$ of the input words, and then predict the masked words
- ✦ The book is on the 
- ✦ Understanding  entailment and contradiction is  to understanding natural language
- ✦ 15% of the words are masked

BERT-FINE TUNING



The goal of Multi-Genre Natural Language Inference(MNLI) is to determine the relationship between the two sentences: entailment, contradiction, or neutral

BIDIRECTIONAL AUTOREGRESSIVE TRANSFORMER(BART)

- ✦ A transformer model
- ✦ Bidirectional encoding
- ✦ Decoder with autoregressive next word generation

BART ARCHITECTURE

- ✦ Encoder - Combines BERT's bidirectional encoder (for context understanding)
- ✦ Decoder: GPT-style with GPT's autoregressive decoder (for text generation)
- ✦ Pre-training Objective - Input corrupted text
 - ✦ Token masking
 - ✦ Token deletion
 - ✦ Span of text masking with a single mask
 - ✦ Sentence permutation and rotation
- ✦ Capture bidirectional context for aspect/opinion
- ✦ Generate structured outputs (e.g., aspect-sentiment pairs) via decoder

ABSA SUBTASKS

Don't believe that these screen protectors have glue in them

- ✦ Aspect Term Extraction (ATE): Identify product/service aspects (e.g., "protection").
- ✦ Opinion Term Extraction (OTE): Detect opinion phrases (e.g., "poor").
- ✦ Aspect Sentiment Classification (ALSC): Assign polarity (positive/negative/neutral) to aspects

1. Aspect Term Extraction (ATE): This task identifies the focal points of sentiment in the text, such as "camera" and "battery life" in a smartphone review.
2. Opinion Term Extraction (OTE) aims to identify adjectives or adverbs expressing feelings or attitudes towards aspects, such as "stunning" for the camera or "disappointing" for battery life.
3. Aspect-Level Sentiment Classification (ALSC) categorizes the sentiment towards each aspect as positive, negative, or neutral
4. Aspect-Oriented Opinion Extraction (AOE) associates sentiments with their corresponding aspects, linking "stunning" to "camera."
5. Aspect Extraction and Sentiment Classification (AESC): Aspects are tagged with their sentiment in one step 1
6. Aspect-Opinion Pair Extraction (AOPE)P: airs each aspect with its qualifying opinion, forming pairs like ("camera", "stunning")
7. Aspect Sentiment Triplet Extraction (ASTE): Combines aspects, opinions, and sentiments into a triplet

MODEL CONFIGURATION

- ✦ Encoder Layers: 12+ layers for deep context capture.
- ✦ Decoder Attention: Cross-attention heads focus on encoder's aspect-related hidden states
- ✦ Pre-training on in-domain text (e.g., product reviews) before ABSA fine-tuning
- ✦ Hyperparameters
 - ✦ Batch Size: 32–64 (adjust based on GPU memory)
 - ✦ Learning Rate: $2e-5$ with linear warmup for 10% of steps.
- ✦ Training Steps: 50k–100k for domain adaptation

DATA PREPARATION

- ✦ Domain-Specific Text: Use reviews, social media posts, or customer feedback (e.g., Yelp, Amazon reviews).
- ✦ Aspect Annotations: Label aspects, opinions, and polarities [6][7]
 - ✦ For the text - "The camera is excellent but battery drains quickly.",
"aspects": [
 {"term": "camera", "polarity": "positive", "span": (4, 10)},
 {"term": "battery", "polarity": "negative", "span": (20, 26)}
]

PREPROCESSING

- ✦ Tokenization: Use BART's tokenizer with absolute positional embeddings (pad on the right)
- ✦ Text Corruption: Apply BART-style noise:
 - ✦ Mask 30% of aspect-related tokens (e.g., "The [MASK] is excellent").
 - ✦ Shuffle sentences containing aspects

FINE-TUNING FOR ABSA

- ✦ Generative ABSA
 - ✦ Convert ABSA to text generation:
 - ✦ Input: `"Identify aspects and sentiments: {text}"`
 - ✦ Output: `"aspect1: polarity1; aspect2: polarity2"` .
- ✦ Use sequence-to-sequence training with cross-entropy loss.
- ✦ Span Extraction
 - ✦ Use the decoder to predict start/end positions of aspects and opinions (QA-style) .
- ✦ Evaluation Metrics
 - ✦ F1-Score: For aspect/opinion extraction.
 - ✦ Accuracy: For sentiment classification.
 - ✦ Jaccard Similarity: For overlap between predicted and true spans

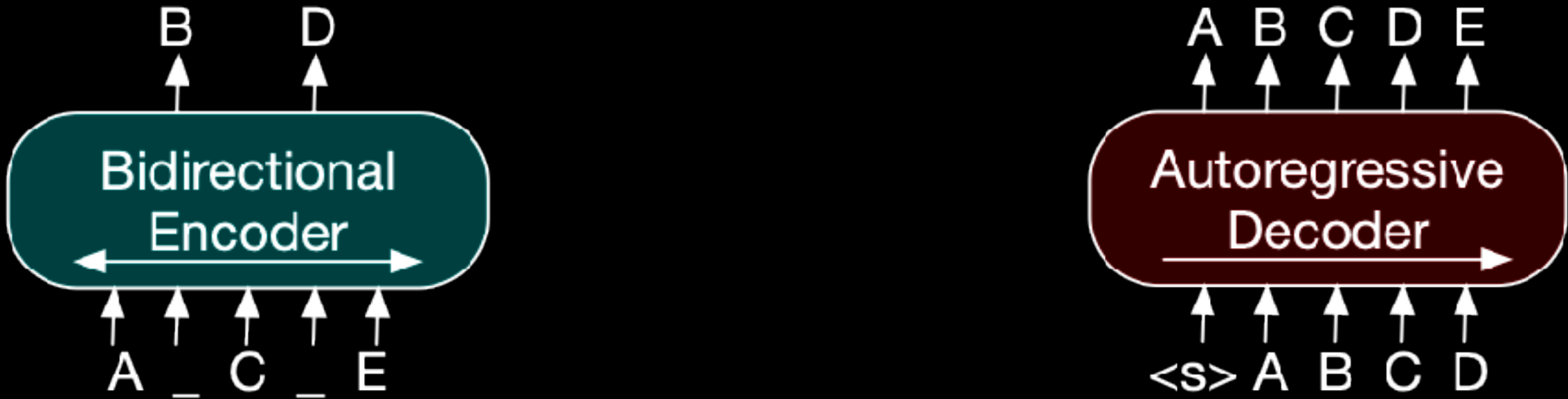
For sets A and B : $\{0,1,2,5,6\}$ and $\{0,2,3,4,5\}$

$$J = \frac{A \cap B}{A \cup B} = \frac{3}{7}$$

REFERENCES

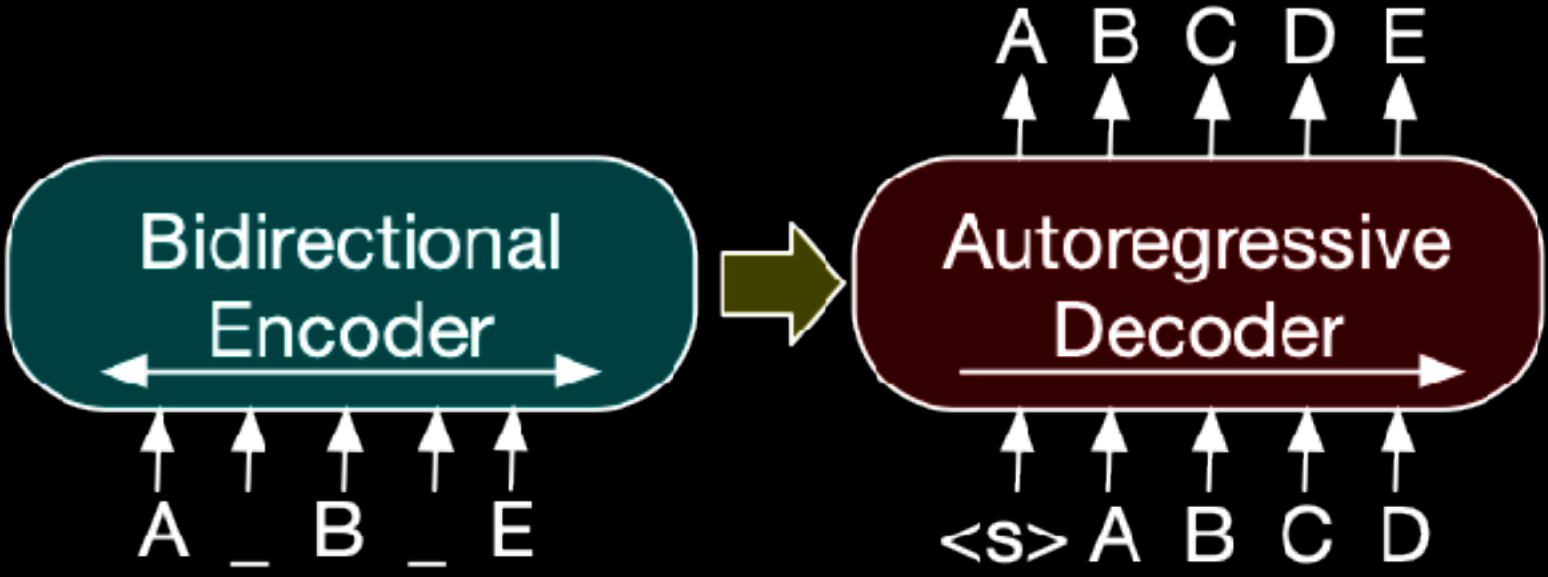
1. A Unified Generative Framework for Aspect-based Sentiment Analysis <https://aclanthology.org/2021.acl-long.188/>
2. BART https://huggingface.co/docs/transformers/en/model_doc/bart
3. Understanding pre-trained BERT for aspect-based sentiment analysis, <https://aclanthology.org/2020.coling-main.21.pdf>
4. Aspect-Based Sentiment Analysis - an overview, <https://www.sciencedirect.com/topics/computer-science/aspect-based-sentiment-analysis>
5. Enhanced local and global context focus mechanism using bart model for aspect based sentiment analysis of social media data, <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10701283>
6. A Systematic Review of Aspect-based Sentiment Analysis - arXiv <https://arxiv.org/abs/2311.10777>
7. Extracting Emotion Phrases from Tweets using BART <https://arxiv.org/abs/2403.14050>

BIDIRECTIONAL AUTO-REGRESSIVE TRANSFORMER



(a) BERT: Random tokens are replaced with masks, and the document is encoded bidirectionally. Missing tokens are predicted independently, so BERT cannot easily be used for generation.

(b) GPT: Tokens are predicted auto-regressively, meaning GPT can be used for generation. However words can only condition on leftward context, so it cannot learn bidirectional interactions.



(c) BART: Inputs to the encoder need not be aligned with decoder outputs, allowing arbitrary noise transformations. Here, a document has been corrupted by replacing spans of text with mask symbols. The corrupted document (left) is encoded with a bidirectional model, and then the likelihood of the original document (right) is calculated with an autoregressive decoder. For fine-tuning, an uncorrupted document is input to both the encoder and decoder, and we use representations from the final hidden state of the decoder.

Figure 1: A schematic comparison of BART with BERT (Devlin et al., 2019) and GPT (Radford et al., 2018).

ADDITIONAL TOPICS TO COVER

- ✦ Situations With Adversarial Generations (SWAG)
- ✦ Multi-Genre Natural Language Inference(MNLI)
- ✦ Sentiment Analysis
- ✦ LORA
- ✦ Compression of Deep Learning Models
- ✦ Language Models as Knowledge Bases?
- ✦ Training language models to follow instructions with human feedback
- ✦ DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter
- ✦ Language Models are Few-Shot Learners
- ✦ Reasoning
- ✦ Multimodal Learning
- ✦ RAG

REFERENCES

✦ <https://github.com/HIT-SCIR/ELMoForManyLangs/tree/master/elmoformanylangs>

