TEXAS
The University of Texas at Austin

I320D – Topics in Human Centered Data Science

Text Mining and NLP Essentials

Week 12: Language Models and Embeddings (…) , NLP Applications

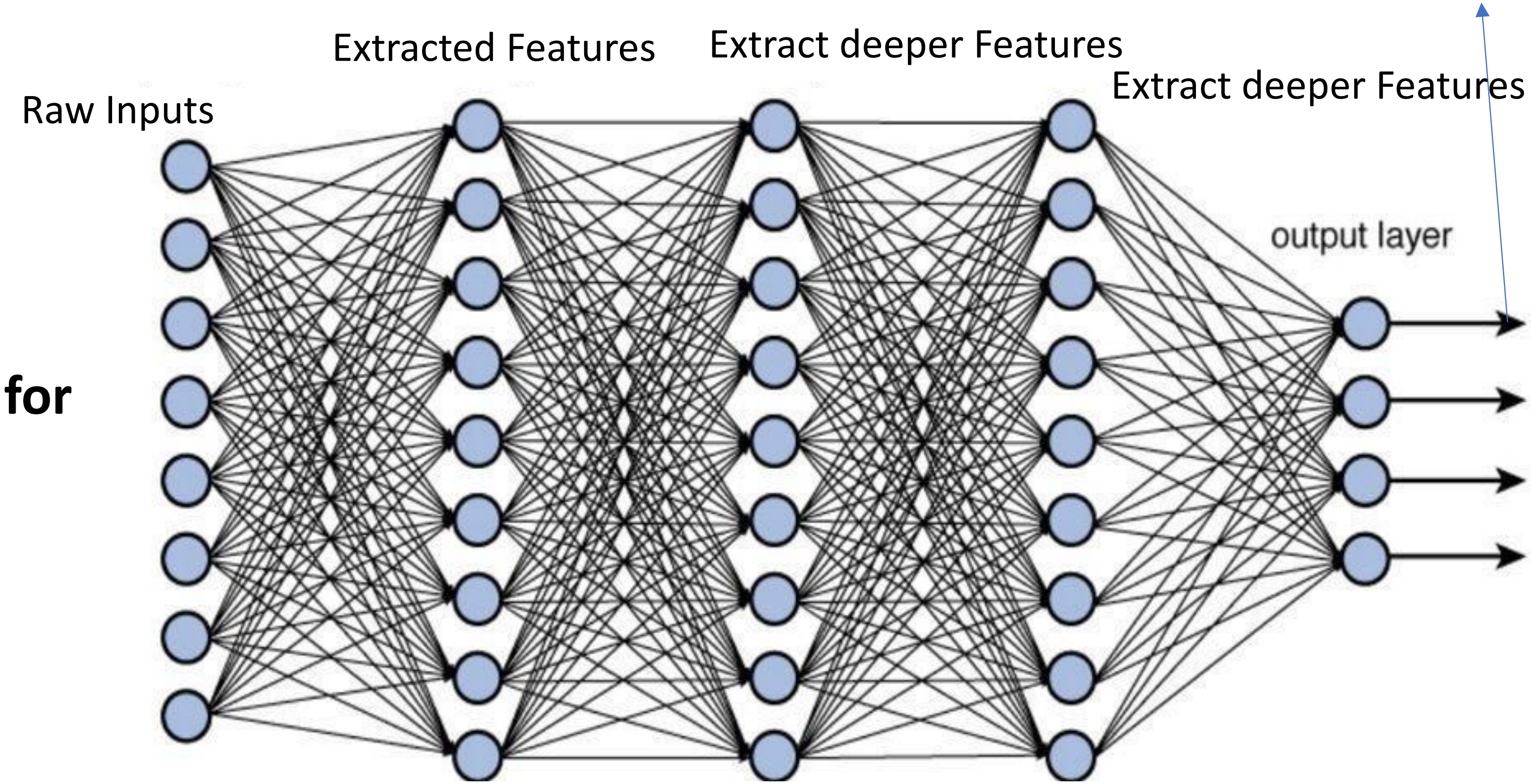
Dr. Abhijit Mishra

Ongoing Assignments / Project

- **Course Project:**
  - Feedback Shared, leverage office hours on Monday / Wednesday
  - 5 mins “Work progress” presentation on **Apr 15**
  - Final Presentation: **Apr 29**
  - Final Report Due : **May 6**
- **Assignment 5: Text Classification (**Due April 7)

Wee 11 Recap: Feed Forward Netwrks

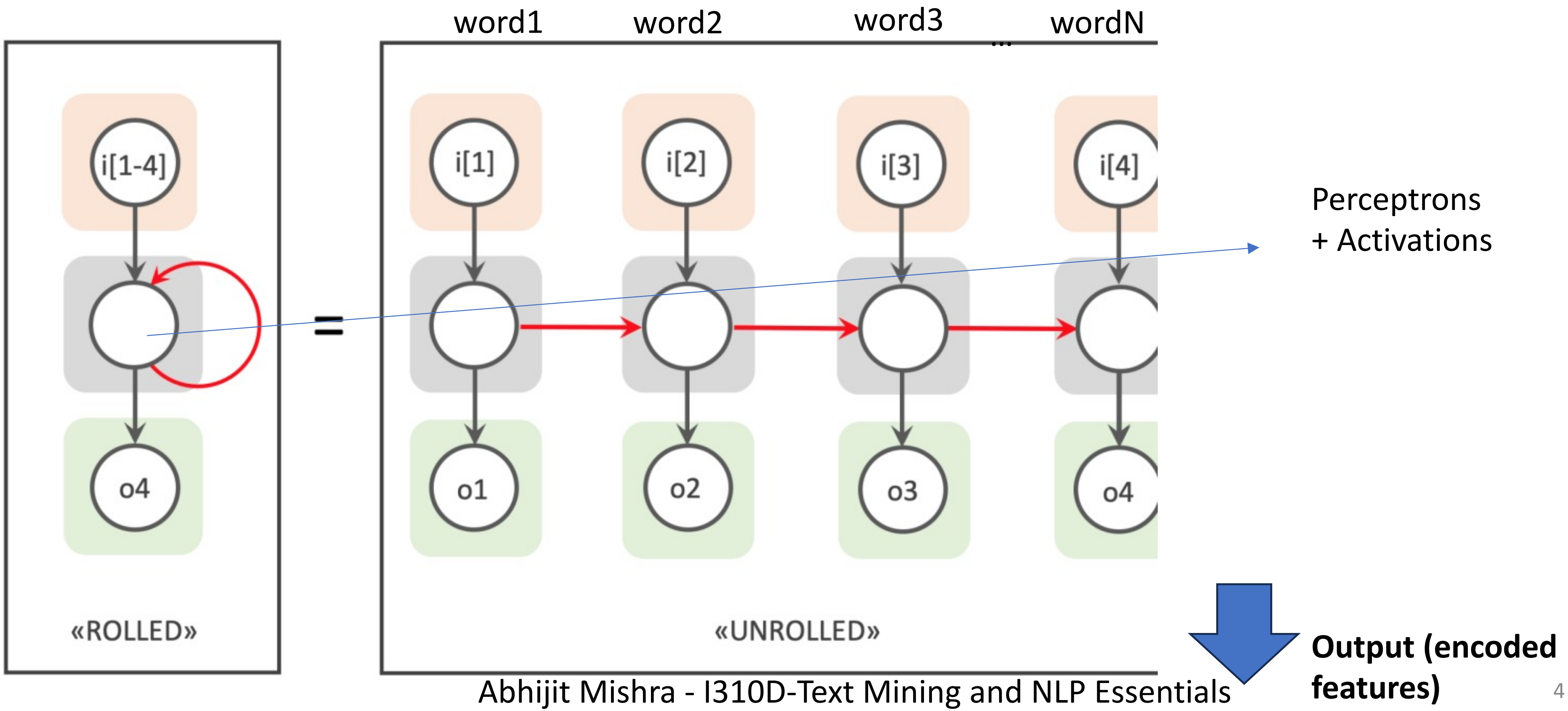
1 hot encodings for words



outputs

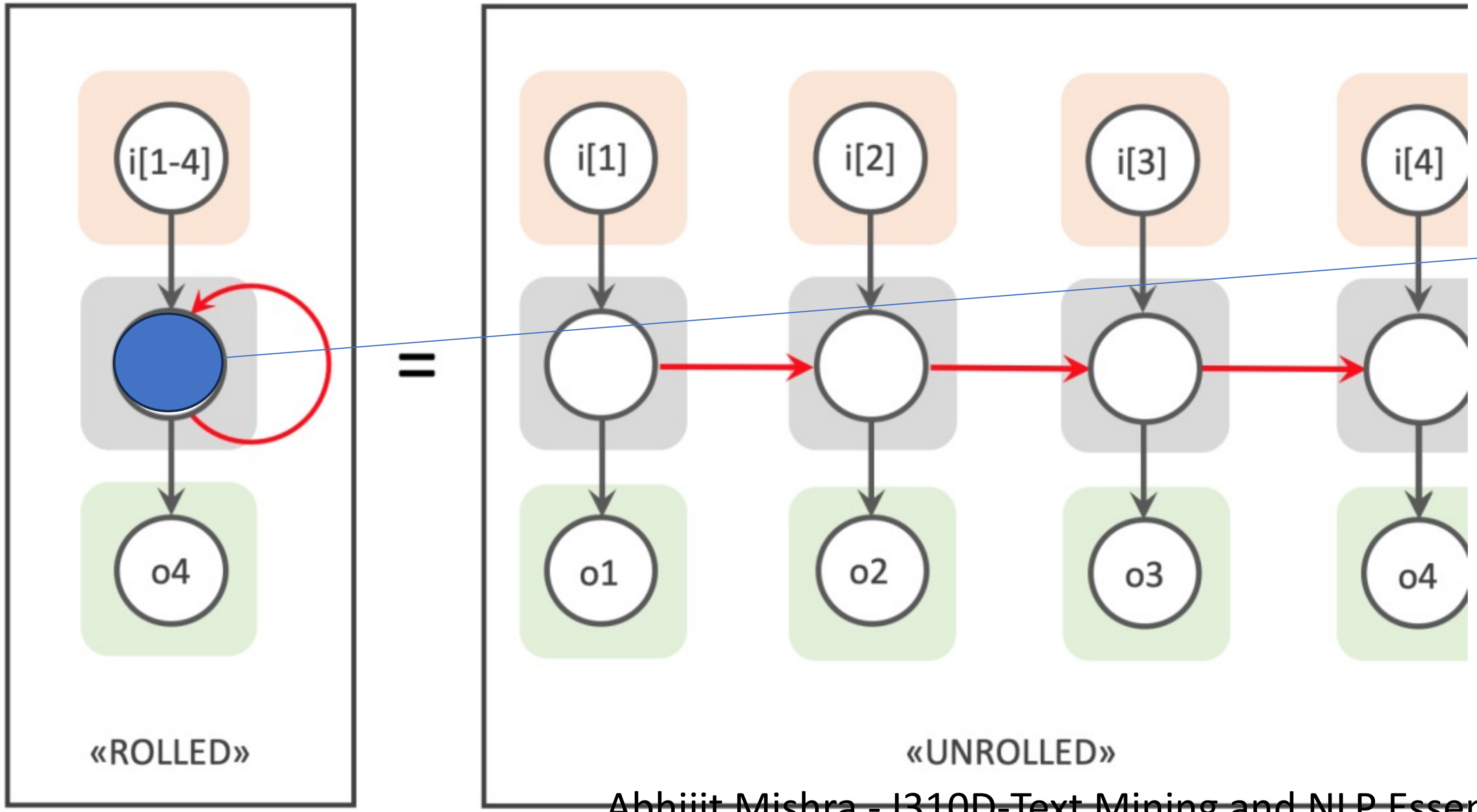
Week 11: Recap

Recurrent Neural Nets (RNNs)

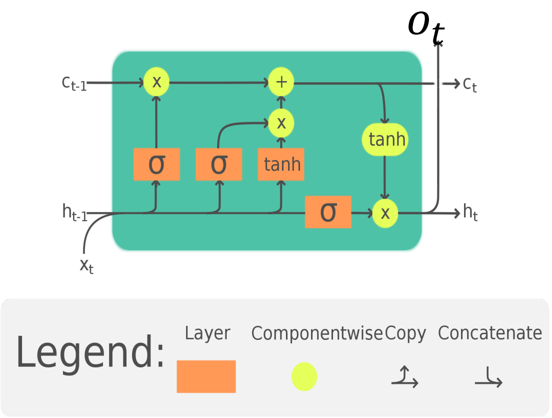


Week 11: Recap

Long Short Term Memories (Schmidhuber et al, 1997)



LSTM cells



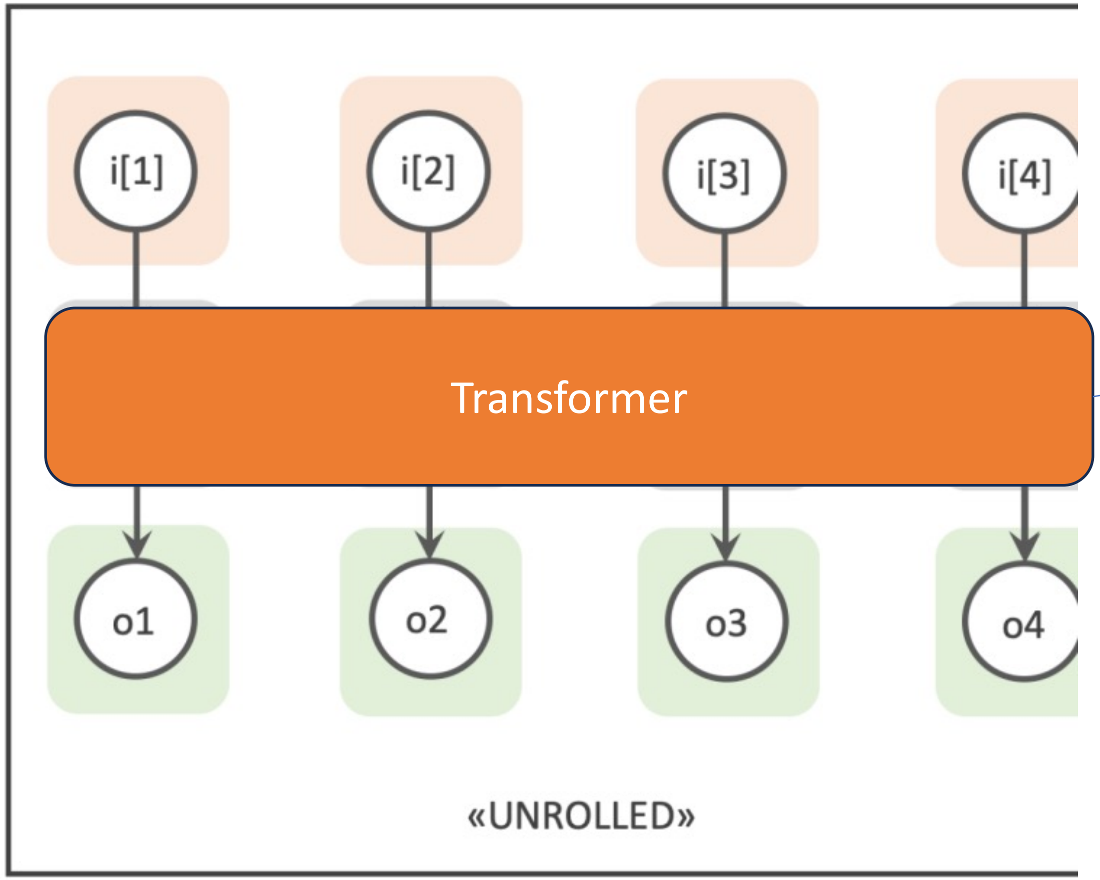
Output (encoded features)

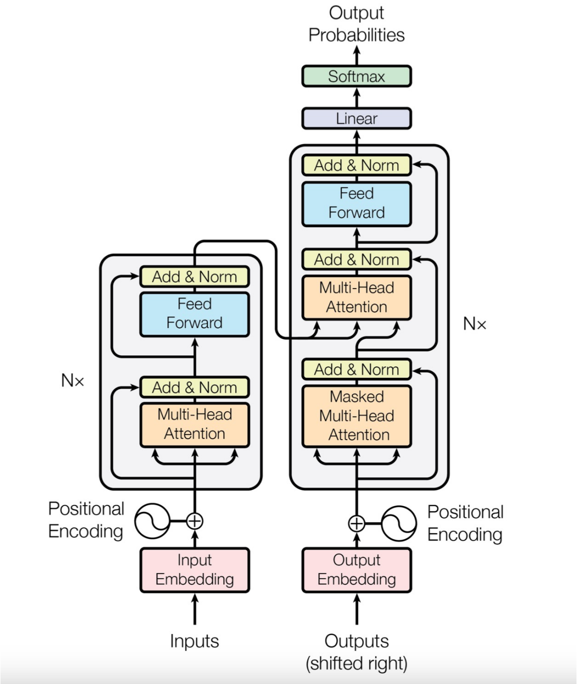
In Python

```
# Define the LSTM model
model = Sequential()
model.add(Embedding(max_features, 128, input_length=maxlen))
model.add(SpatialDropout1D(0.2))
model.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(1, activation='sigmoid'))

# Compile the model
model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])

# Train the model
model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs,
validation_data=(x_test, y_test))
```

Week 11: Recap

Transformers (Vaswani et al, 2018)





Output (encoded features)

Recap: Language Models (LMs)

- Language models are statistical or deep learning models that learn to predict the probability of a sequence of words in a sentence or text
- For a sequence of words $W = (w_1, w_2, w_3, ..., w_n)$
- A language model can be expressed as

$f(X, \theta) \Rightarrow$

$$P(W|\theta) = P(w_1|\theta) \cdot P(w_2|w_1, \theta) \cdot P(w_3|w_1, w_2, \theta) \cdot \ldots \cdot P(w_n|w_1, w_2, ..., w_{n-1}, \theta)$$

- Here theta =>model parameters

LMs are Generative Models

- Language Models are Generative in Nature

Generative Modeling of Text

- Tasks where:
  - Input is a sequence $X = \{x_1, x_2 \ \ldots, x_N\}$ **or** $\mathbf{X} \in \mathbb{R}^{\boldsymbol{N}}$
  - Output is a sequence $Y = \{y_1, y_2 \ \ldots, y_M\}$ **or** $\mathbf{Y} \in \mathbb{R}^{\boldsymbol{M}}$

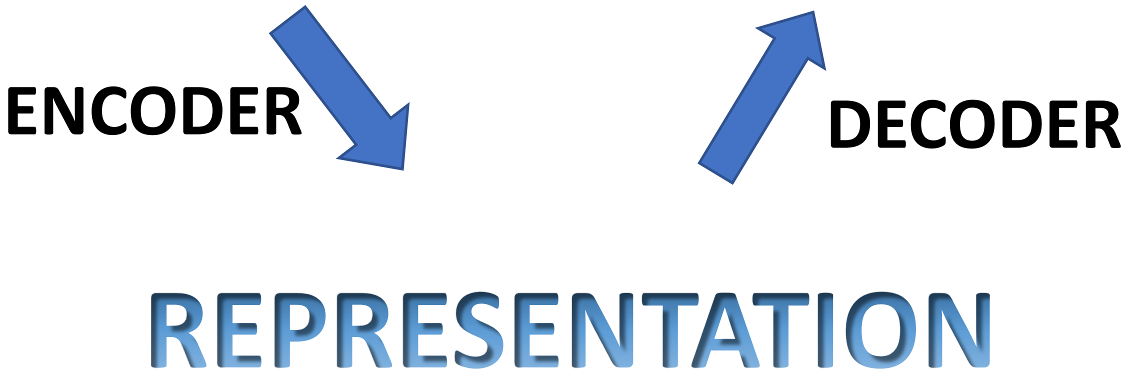
- Example:
  - Text summarization
  - Machine Translation
  - Chat generation

Sequence Generation – Text Summarization Example

SOURCE: *Roger Federer wins a record eighth men's singles title at Wimbledon on Sunday.*

TARGET:

Roger Federer won the Wimbledon



Sequence Generation – English-Spanish Translation Example

SOURCE: *Roger Federer wins a record eighth men's singles title at Wimbledon on Sunday.*

TARGET:

Roger Federer gana un octavo título individual masculino en Wimbledon el domingo.

ENCODER

DECODER

REPRESENTATION

Sequence Generation – Language Modeling Example

SOURCE: *Roger Federer wins a record eighth*

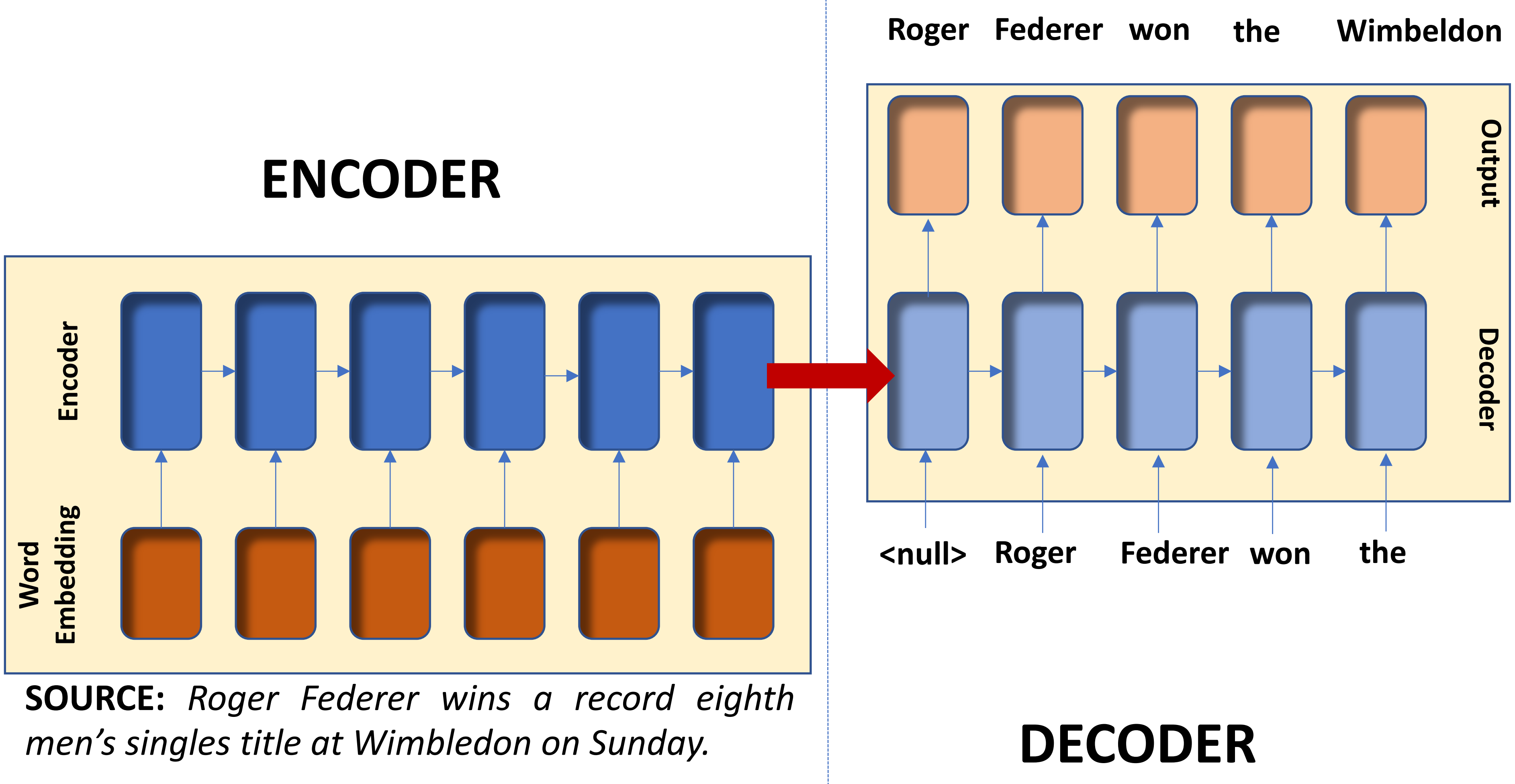
TARGET:
men's singles title at Wimbledon on Sunday.

ENCODER

DECODER

REPRESENTATION

Zooming into Encoder-Decoder Models



SOURCE: *Roger Federer wins a record eighth men's singles title at Wimbledon on Sunday.*

We have Various choices for the BLUE blocks (RNNs, LSTMs, Transformers)

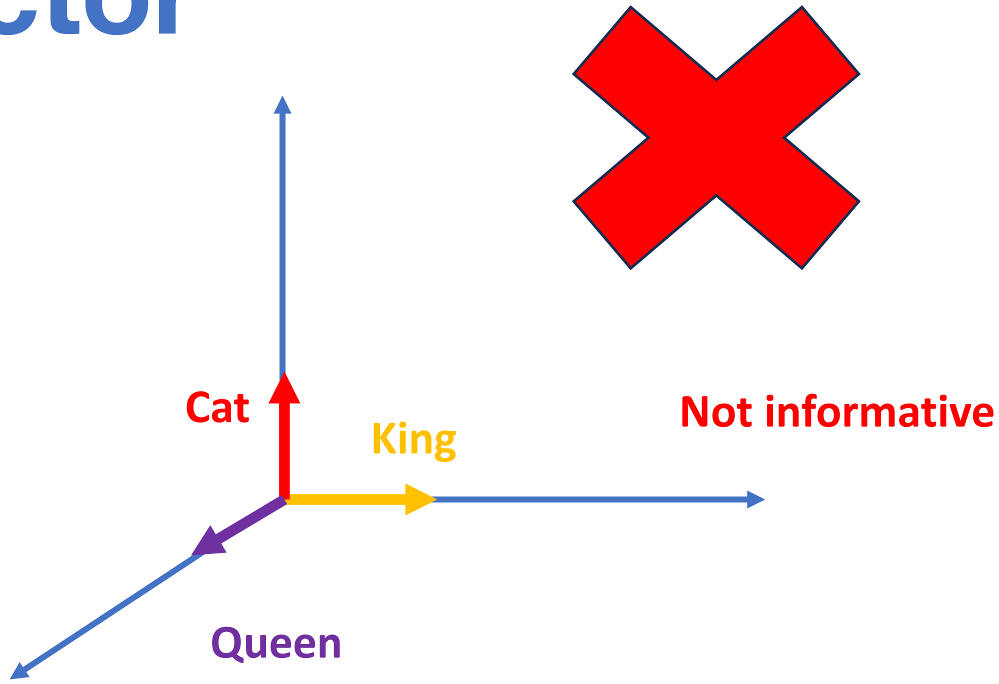
Embeddings: Representation Learning

- **Feature Engineering from input words / input sentences?**
  - What do we intend to do?

- Extract meaningful representations:
  - In computer understandable numerical forms
  - Should capture relationships between words
    - Synonymy (e.g., “specimen”, “sample”)
    - Antonymy (e.g., ”man”, “woman”)
    - Conceptual similarity (e.g., “Wednesday”, “Monday”) (“USA”,”Canada”)

Issues with One-hot vector

Word	1-hot vector
Queen	[1, 0, 0]
King	[0, 1, 0]
Cat	[0, 0, 1]

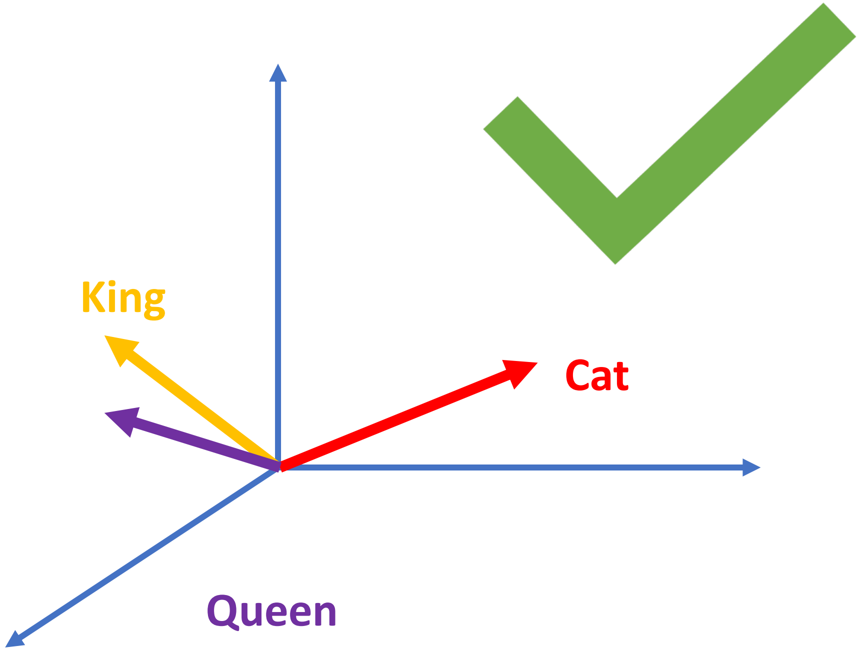


$$D_{cosine}(\text{“}Cat\text{”}, \text{“}King\text{”}) = D_{cosine}(\text{“}Cat\text{”}, \text{“}Queen\text{”}) = D_{cosine}(\text{“}King\text{”}, \text{“}Queen\text{”}) = 1$$

$$D_{Euclid}(\text{“}Cat\text{”}, \text{“}King\text{”}) = D_{Euclid}(\text{“}Cat\text{”}, \text{“}Queen\text{”}) = D_{Euclid}(\text{“}King\text{”}, \text{“}Queen\text{”}) = \sqrt{2}$$

Instead, we need

Word	1-hot vector
Queen	[1.5, -1.3, -0.9]
King	[2.1, -0.7, 0.2]
Cat	[0.3, 1.9, -0.4]



$$D_{cosine}(\text{“}Cat\text{”}, \text{“}King\text{”}) = 1.17, D_{cosine}(\text{“}Cat\text{”}, \text{“}Queen\text{”}) = 1.38,$$
$$D_{cosine}(\text{“}King\text{”}, \text{“}Queen\text{”}) = 0.19$$

$$D_{Euclid}(\text{“}Cat\text{”}, \text{“}King\text{”}) = 3.21, D_{Euclid}(\text{“}Cat\text{”}, \text{“}Queen\text{”}) = 3.45$$
$$D_{Euclid}(\text{“}King\text{”}, \text{“}Queen\text{”}) = 1.38$$

What are Word Embeddings?

- Embeddings are matrices of shape $V \times E$.
- V represents the size of the vocabulary (or total number of valid words in a language).
- E represents the dimension of each vector for the word.
- Typically $E \ll V$
- In other words, we are projecting sparse **one-hot encodings** of words (of dimension V) to dense Embeddings of size E
- Size of E is typically in 100s (300, 600, 1000).

Digression: Matrix Multiplication

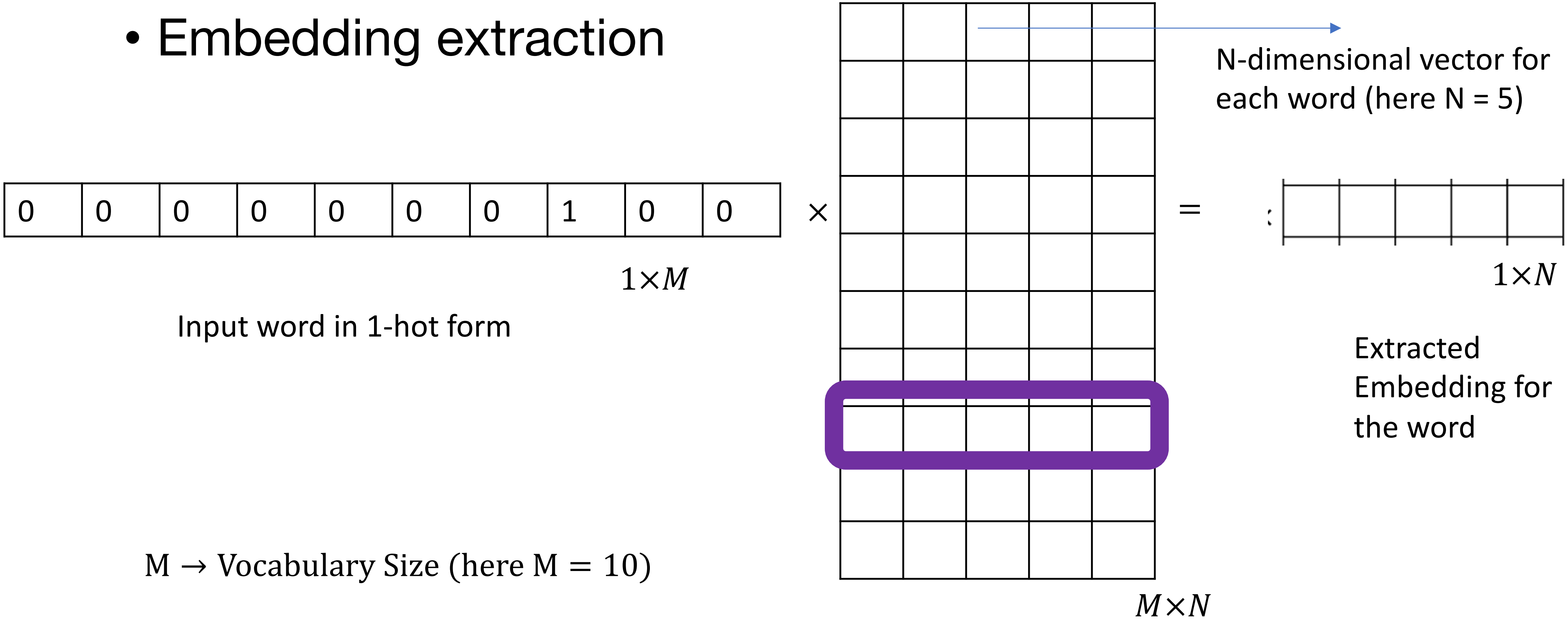
- Multiplying to matrices of shape $M{\times}N$ and $N{\times}P$ *yields an* $M{\times}P$ matrix
- Example:

$$[1\ 2\ 3]\times\begin{bmatrix}2 & 3\\ 4 & 5\\ 6 & 7\end{bmatrix} = [1{\times}2+2{\times}4+3{\times}6\ \ 1{\times}3+2{\times}5+3{\times}7]$$

$$= [2+8+18\ \ 3+10+21] = [28\ 34]$$

Digression: Projection through matric multiplication

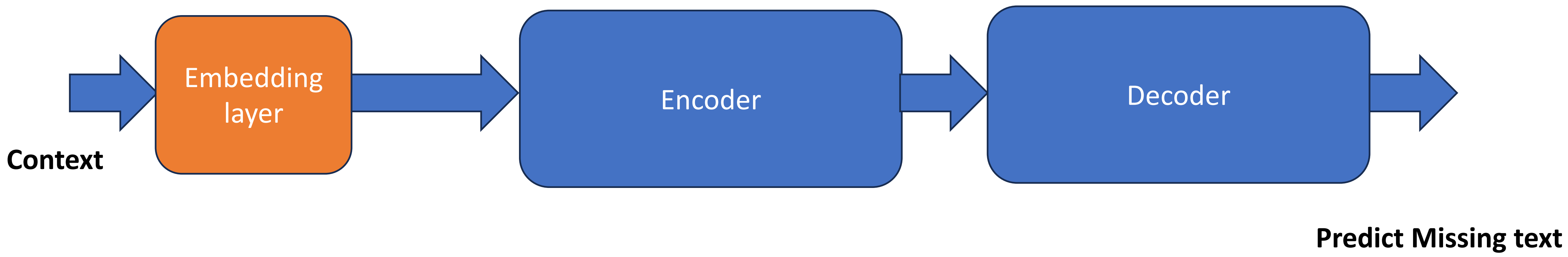
- Embedding extraction



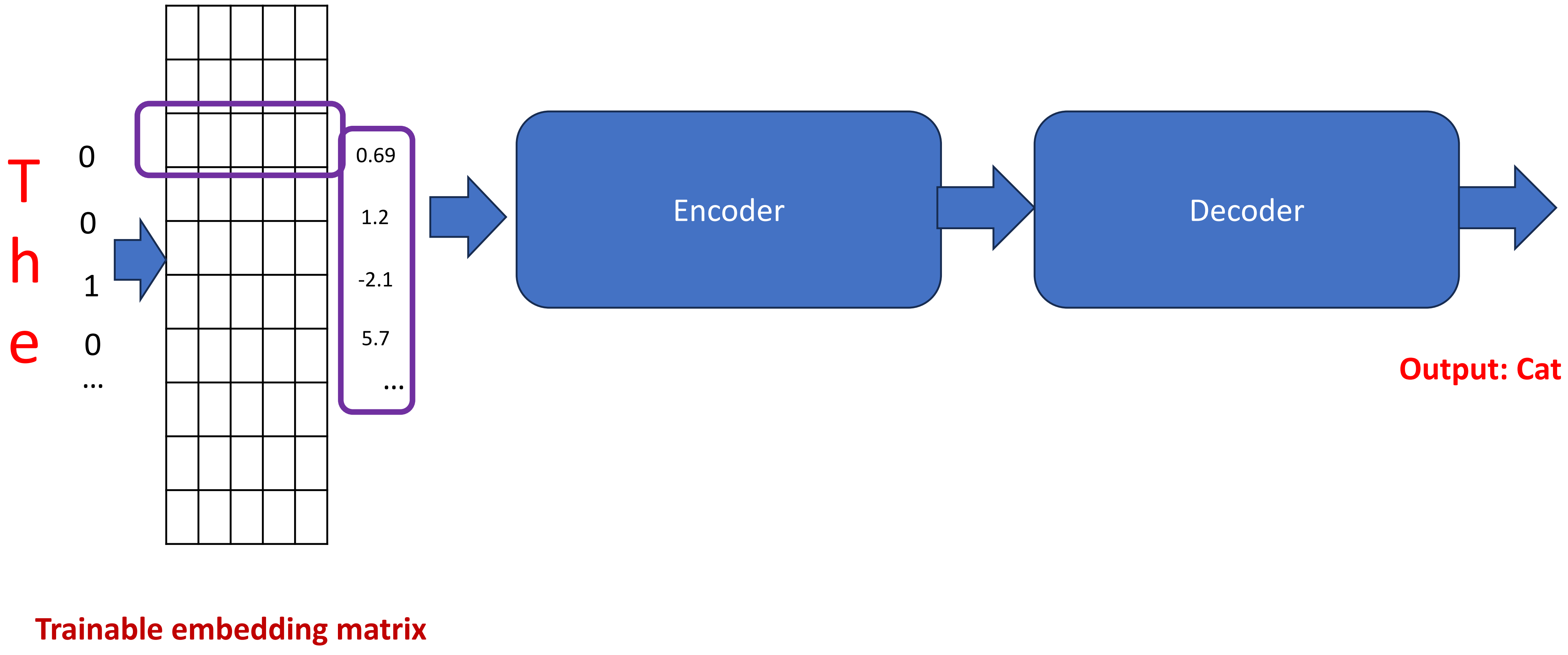
Embedding Matrix (initialized randomly, updated through backpropagation)

Encoder Decoder Models with Embeddings

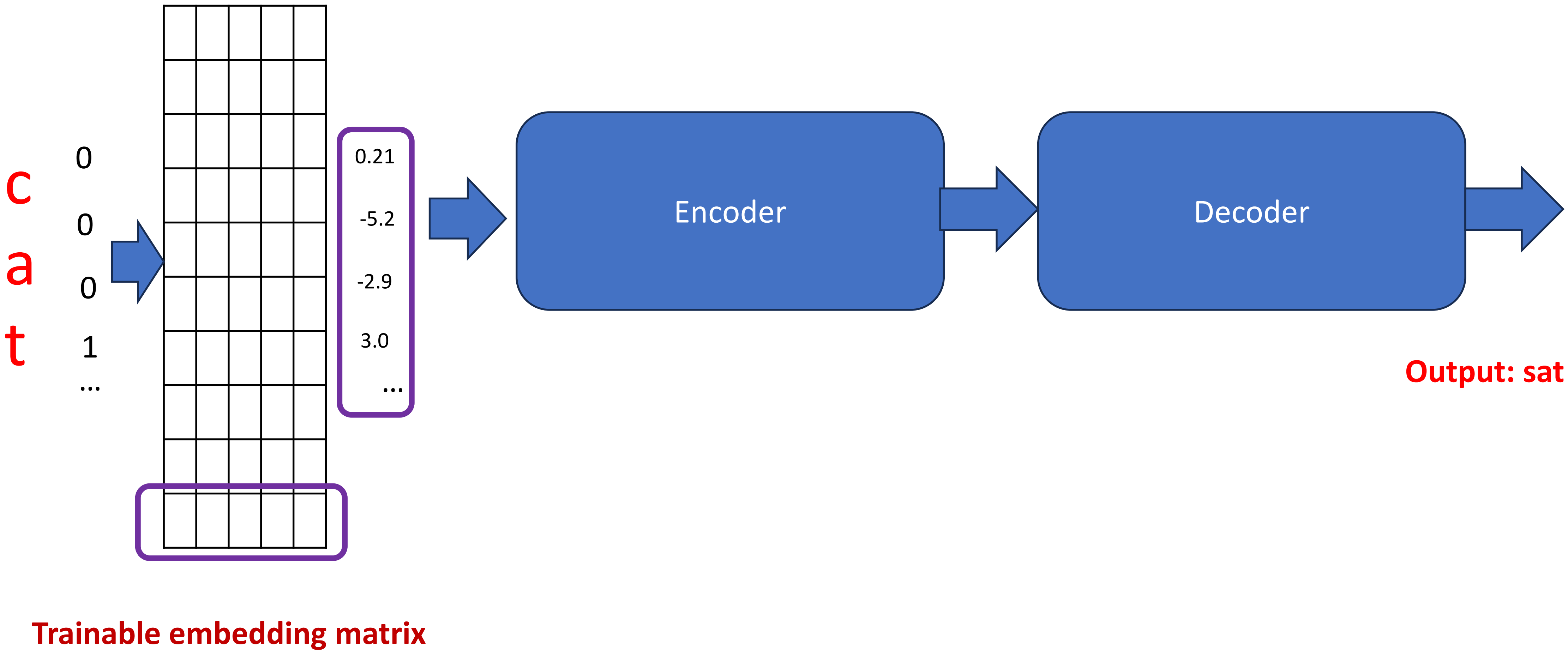
- Example Sentence: **“The cat sat on the mat”**



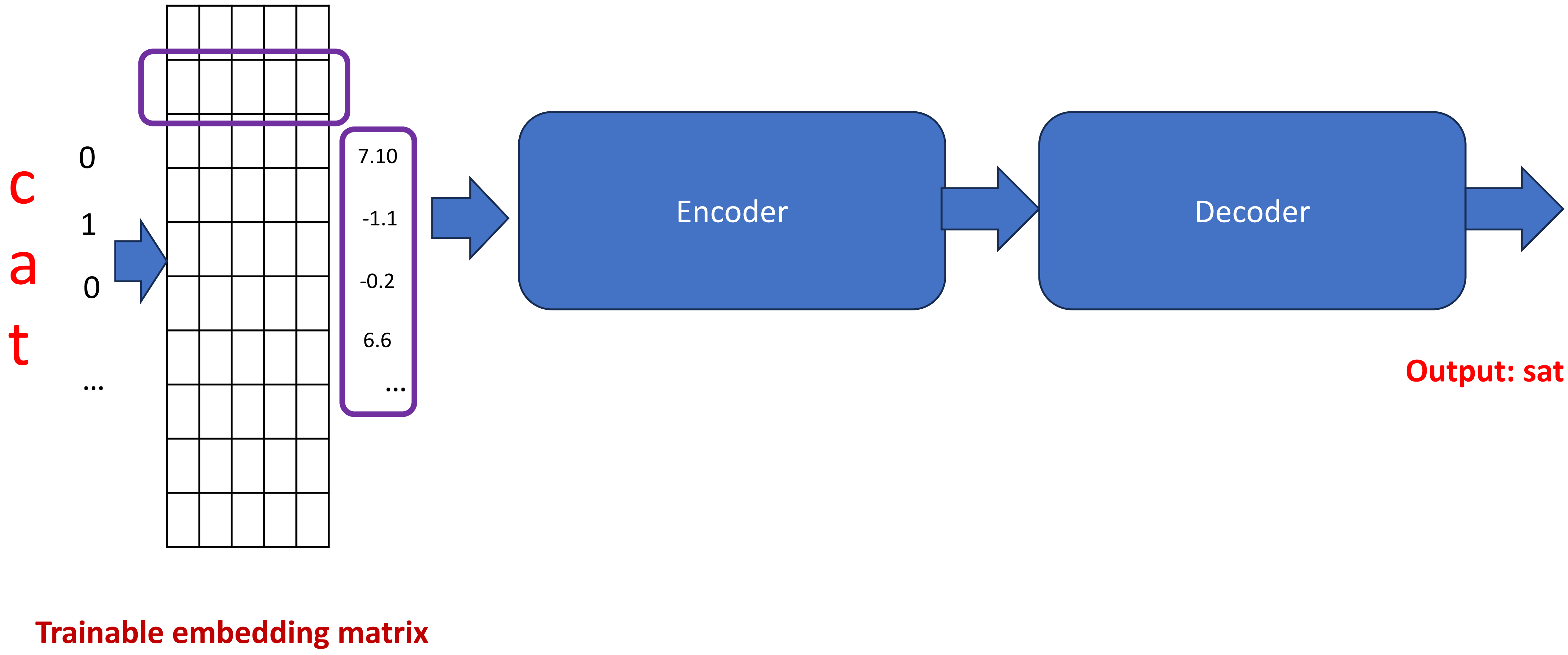
- Processing: **“The”**



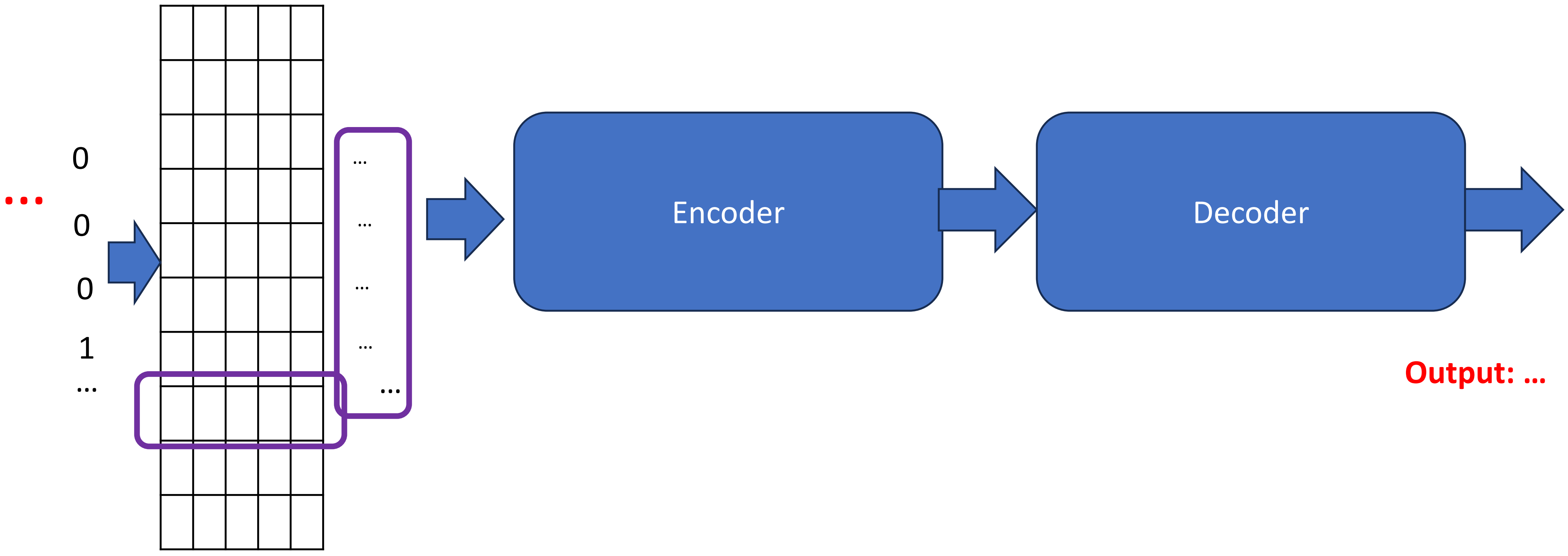
- Processing: “**cat**”



- Processing: “**sat**”



- Processing: "..."



Trainable embedding matrix

Embedding Layer

- A matrix of size $(Vocabulary\ size \times Embedding\ dimension)$
- Initialized with random values but updated using backpropagation
- From 1-hot to embeddings
    - 1 lookup operation

Word2Vec: Learning Embeddings with Feed Forward Encoder Decoder based LMs

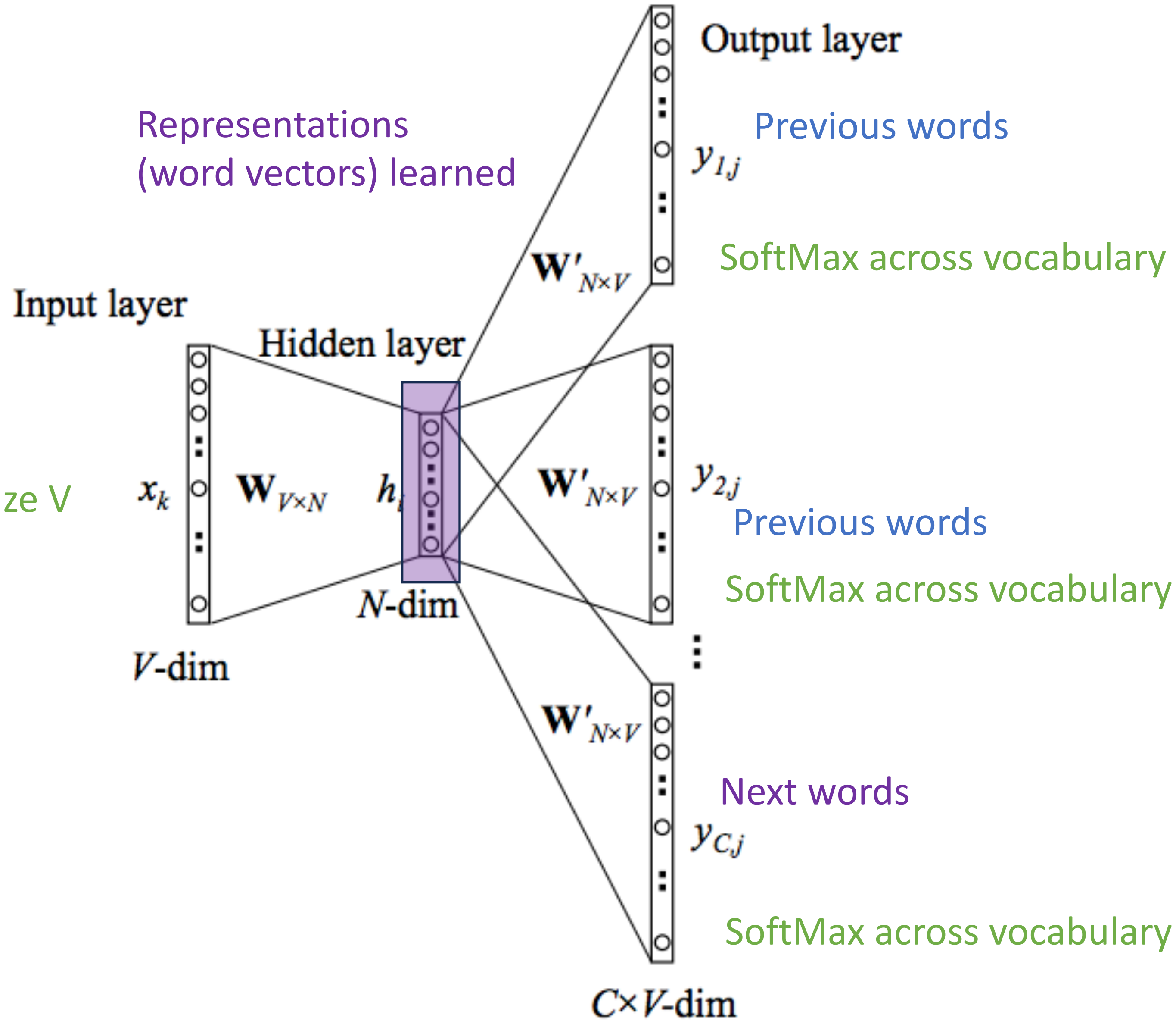
- **Skip Gram Objective :** Given a word can we predict the previous and the next words (or predict surrounding context given an input
- A feed forward network can be designed to perform this task
- **Dataset:**
    - Examples containing <input word, context> can be automatically created using large amount of corpus (e.g., Wikipedia, News Database etc)

Skip Gram Example

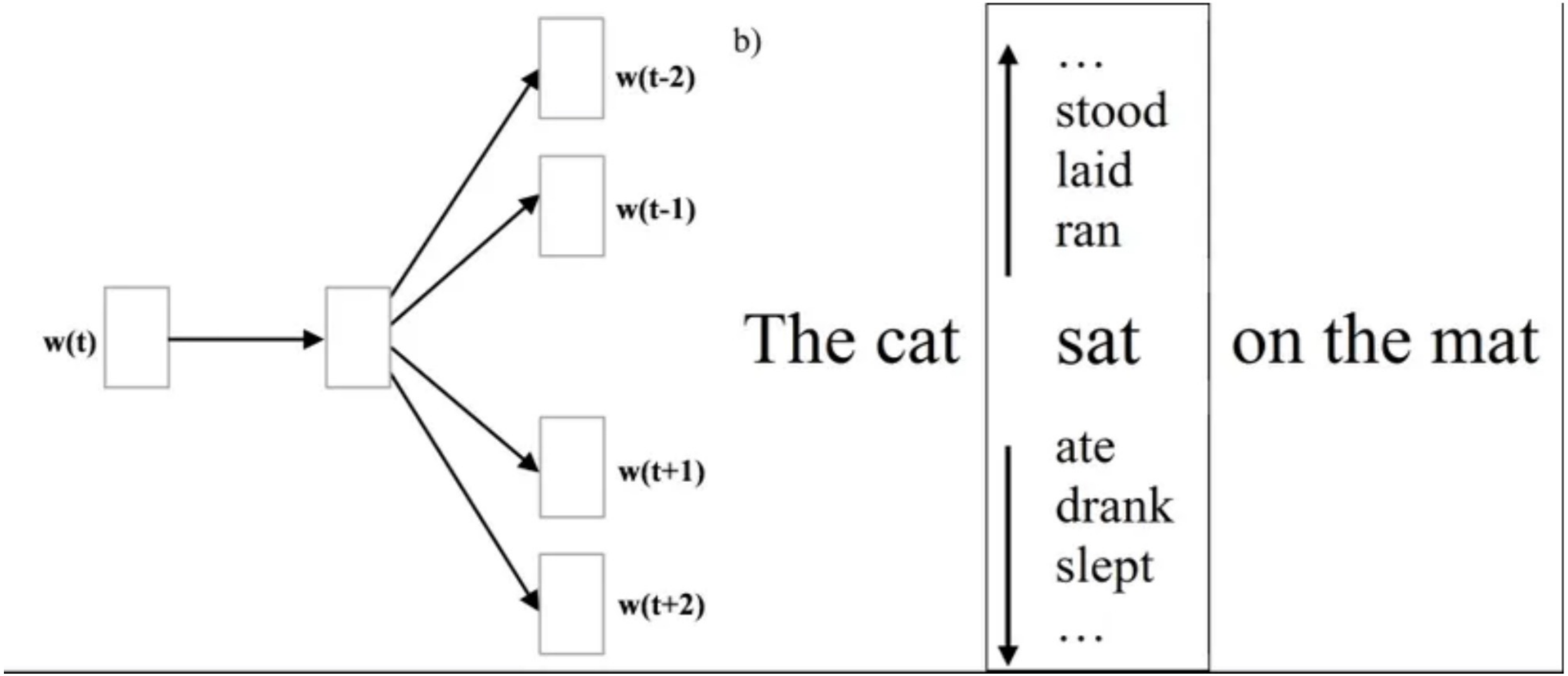
- Original sentence: “The cat sat on the mat”
- Preparing Training data:
  - **Input: “sat”**
  - **Output: “the cat on”**

SkipGram Model using Feed Forward Nets

Input word (1-hot vector) of size V



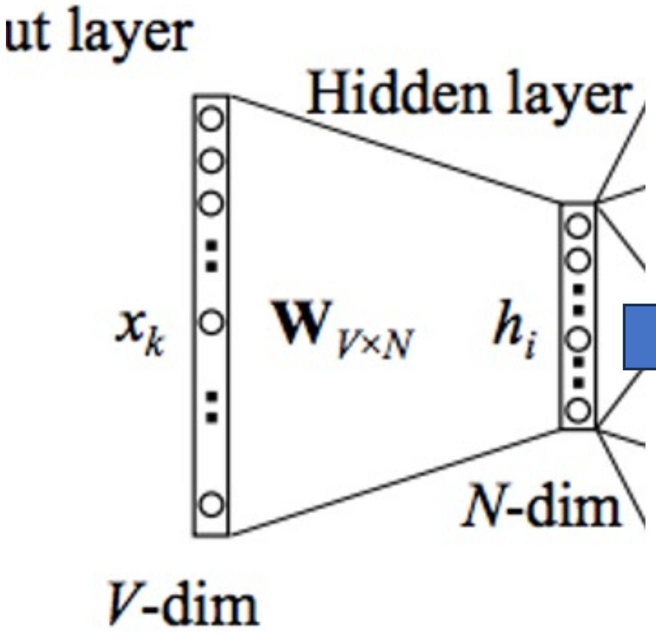
SkipGram Model using Feed Forward Nets



https://towardsdatascience.com/skip-gram-nlp-context-words-prediction-algorithm-5bbf34f84e0c

After Training: During Inference

Word



Feature Vector of size N

These features are the word embeddings we use

Another Option: The CBOW Model

- What about we predict center word, **given context word, opposite to the skip-gram model**?
- Yes, this is called Continuous Bag Of Words model in the original Word2Vec paper.

Word2Vec: Pros and Cons

- **Pros:**
  - Respects language and order to some extent
  - Efficient Training Process: Simple FFDs
  - Semantic Relationships Preservation
- **Cons:**
  - Loss of local context
  - Does not capture POS variations and word senses
    - Word “bank” will mostly be treated as NOUN
    - Word “bank” will always yield the same vector

GloVE

GloVe: Global Vectors for Word Representation

Jeffrey Pennington, Richard Socher, Christopher D. Manning
Computer Science Department, Stanford University, Stanford, CA 94305
jpennin@stanford.edu, richard@socher.org, manning@stanford.edu

- The GloVe algorithm extracts word vectors by optimizing a defined objective function that captures the statistical co-occurrence information between words

Glove-step-2: Form objective function

$$minimize_{w_1, w_2 \,....w_V, b_1, b_2, ...b_V} \; (J)$$

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

Co-occurrence between word i and j

Parameters to be learned

Co-occurrence between word i and j

$f\left(X_{ij}\right)$is a weighting function that assigns more weight to less frequent co-occurrences to prevent extremely frequent words from dominating the training.

Glove-step-2: Optimize objective

$$minimize_{w_1, w_2 \dots w_V, b_1, b_2, \dots b_V} \; (J)$$

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(\underline{w_i^T \tilde{w}_j} + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

These are the word embeddings and needs to be learned

How?

$$minimize_{w_1, w_2 \, \ldots . w_V, b_1, b_2, \ldots b_V} \; (J)$$

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

Initialize randomly and update through gradient descent

How?

$$minimize_{w_1, w_2 \ldots w_V, b_1, b_2, \ldots b_V} \quad (J)$$

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

We can randomly initialize a d dimensional vector per word (e.g., d = 50, d = 200)

How to evaluate word embeddings?

- Also by word analogy tasks (Mikolov et al, 2013)
- Solving analogies of the form **"a is to b as c is to ?" or "a:b :: c:"** where you are given three words and you need to find the fourth word that completes the analogy
- Examples:

1. "Man is to woman as king is to ____"
2. "Spain is to Madrid as France is to ____"
3. "Eat is to food as drink is to ____"

Sentence Vectors

Transformer Based LM: BERT Example

- BERT: **B**idirectional **E**ncoder **R**epresentation **T**ransformers
- Trained with two objectives :
  - Masked token prediction
  - Next sentence prediction

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

Training Task: Masked Token Prediction

[CLS]

Let's

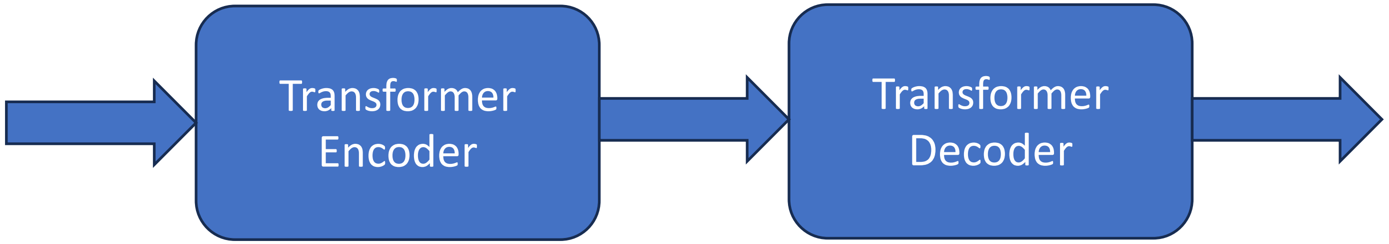
watch

a

<MASK>

tonight

[SEP]



[CLS]

Let's

watch

a

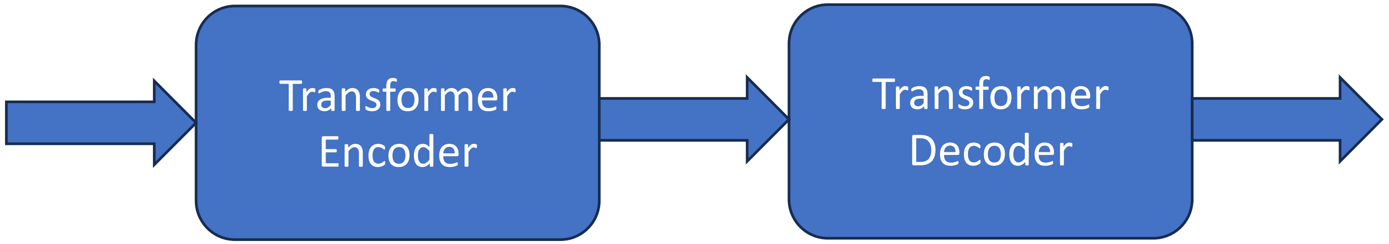
movie

tonight

[SEP]

Training Task: Next Sentence Prediction

[CLS]
Let's
watch
a
movie
tonight
[SEP]



[CLS]
We
will
also
grab
popcorn
[SEP]

Transformer Based LM: BERT Example

[CLS]

Let's

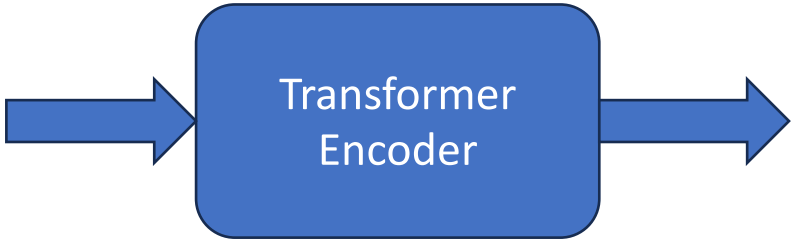
watch

a

movie

tonight

[SEP]



Sentence Embeddings Extraction /
Sentence Feature Extraction

We only use a portion of a network that helps extract features

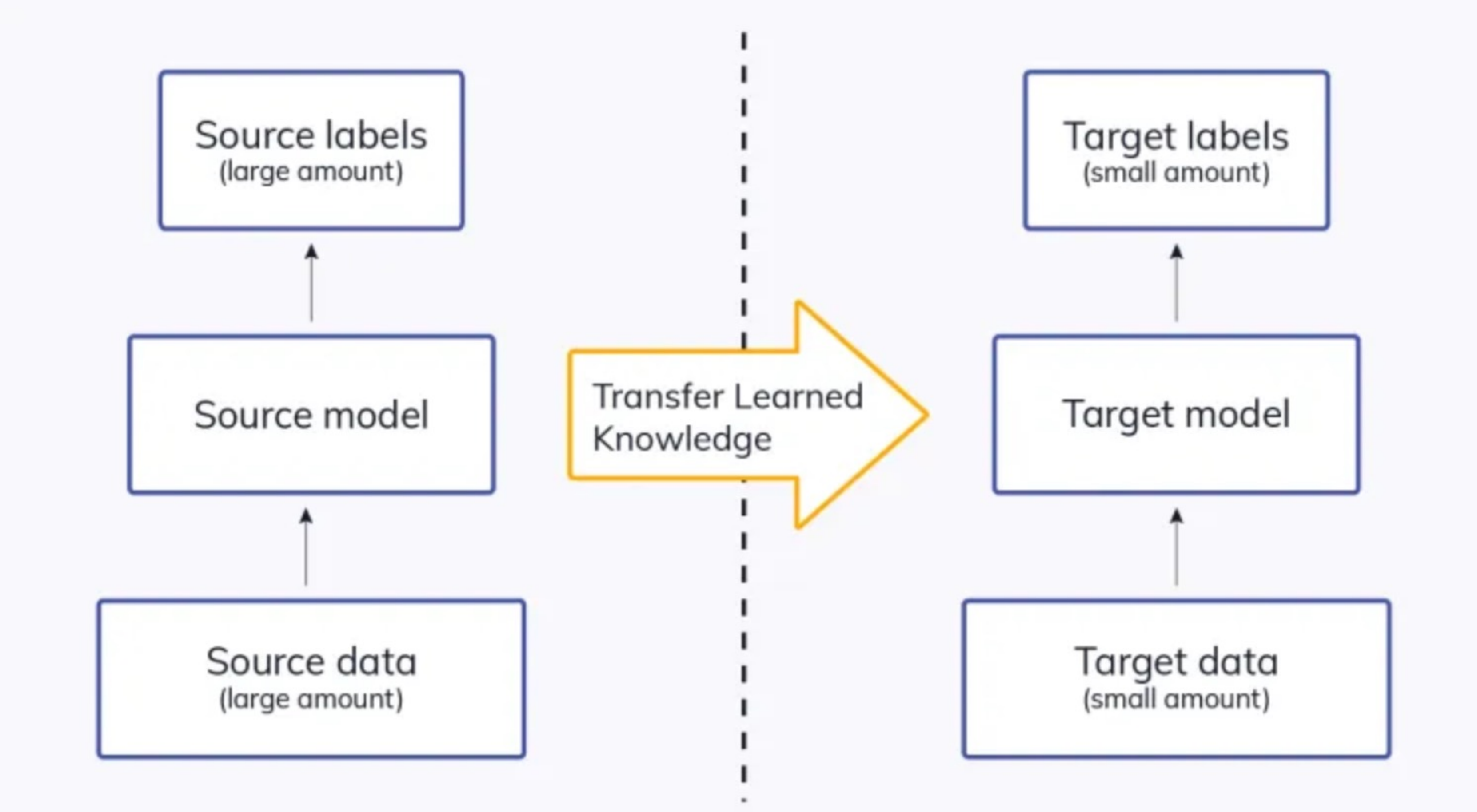
(also known as encoder)

Sentence Vectors: Pros and Cons

- **Pros:**
  - Contextual understanding
  - Bi-directional learning
- **Cons:**
  - Computational complexity
  - Lack of interpretability
  - Large memory footprint

Transfer Learning and NLP Applications

Transfer Learning : Idea

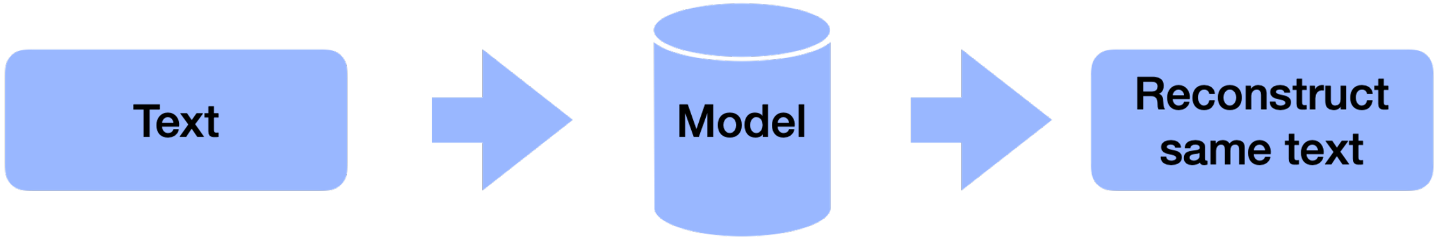


Transfer Learning

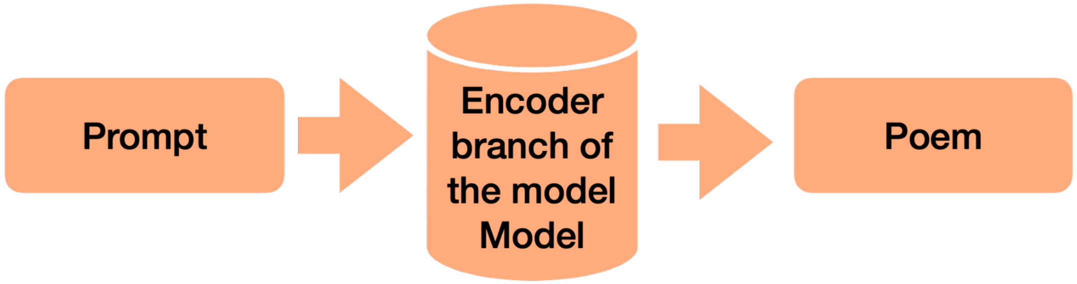
- Supervised learning for one task and transferred to another task that does not have a lot of labeled data
- Typically, models are pre-trained on large amount of data for a well known task
- Transferred to other task using small amount of training data for the target task
- Highly popular in Deep Learning

Transfer Learning in text - Example

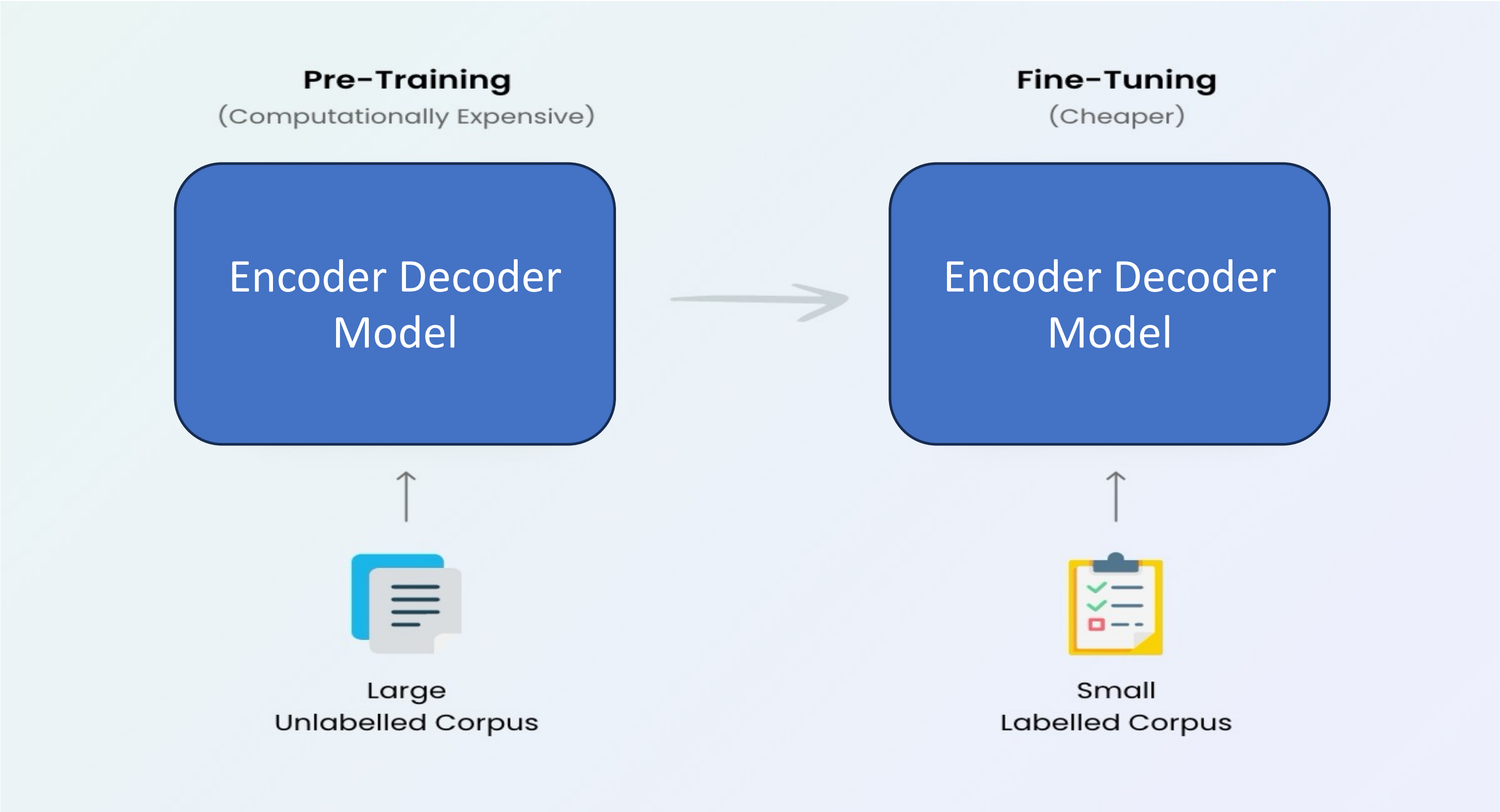
- Train a language model for reconstruct Missing. / Future text.



- Transfer and fine-tune “a branch” of the trained model to do specialized text generation (e.g., Poem), with few examples

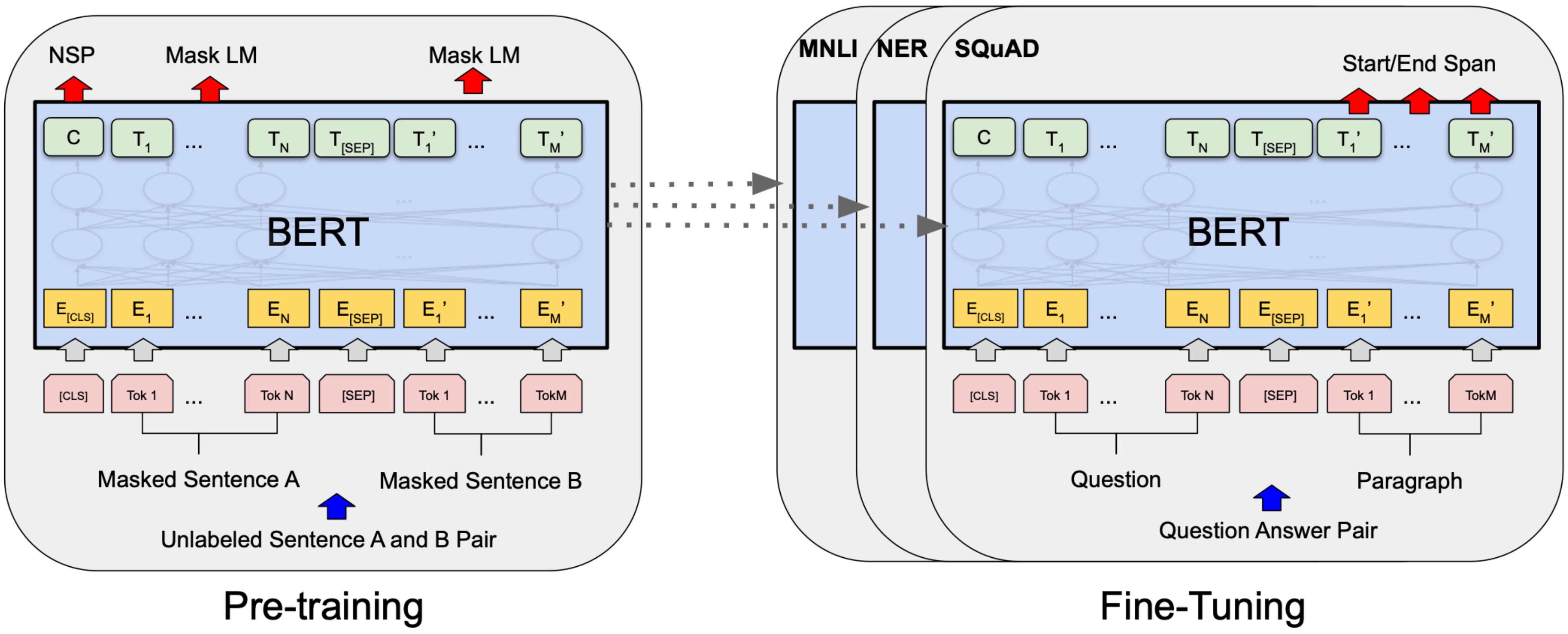


Transfer Learning in Text



Some Pre-trained Language Models

BERT


NSP
Mask LM
Mask LM
C
T1
TN
T[SEP]
T1'
TM'
BERT
E[CLS]
E1
EN
E[SEP]
E1'
EM'
[CLS]
Tok 1
Tok N
[SEP]
Tok 1
TokM
Masked Sentence A
Masked Sentence B
Unlabeled Sentence A and B Pair
MNLI
NER
SQuAD
Start/End Span
Question
Paragraph
Question Answer Pair
Pre-training
Fine-Tuning

BERT – pretraining

- BERT aims to learn a contextualized representation for each token in a given sentence by optimizing the following objective function
- Let T be the input sequence of tokens. BERT learns the parameters θ by maximizing the log likelihood of the next sentence prediction and the masked language model objectives

$$\max_\theta \sum_{(T_a,T_b)\in D} \log p(IsNext|T_a, T_b, \theta) + \sum_{t\in T} \log p(t|T_{\neg t}, \theta)$$

Dataset used

- BERT was trained on a large corpus that includes BooksCorpus (800 million words) and English Wikipedia (2,500 million words).

Tokenization

- BERT uses WordPiece tokenization, which breaks words into subwords based on a fixed vocabulary. This approach enables the model to handle out-of-vocabulary words effectively.

Downstream Fine-tuning Tasks:

- BERT can be fine-tuned for various downstream NLP tasks, including but not limited to text classification, question-answering, named entity recognition, text entailment, and sentiment analysis.

RoBERTa

- An extension of BERT

Optimizes only Masked LM objective

$$\max_\theta \sum_{t \in T} \log p(t | T_{\neg t}, \theta)$$

Tokenization

- Similar to BERT, RoBERTa utilizes WordPiece tokenization for subword token handling.
-

Dataset

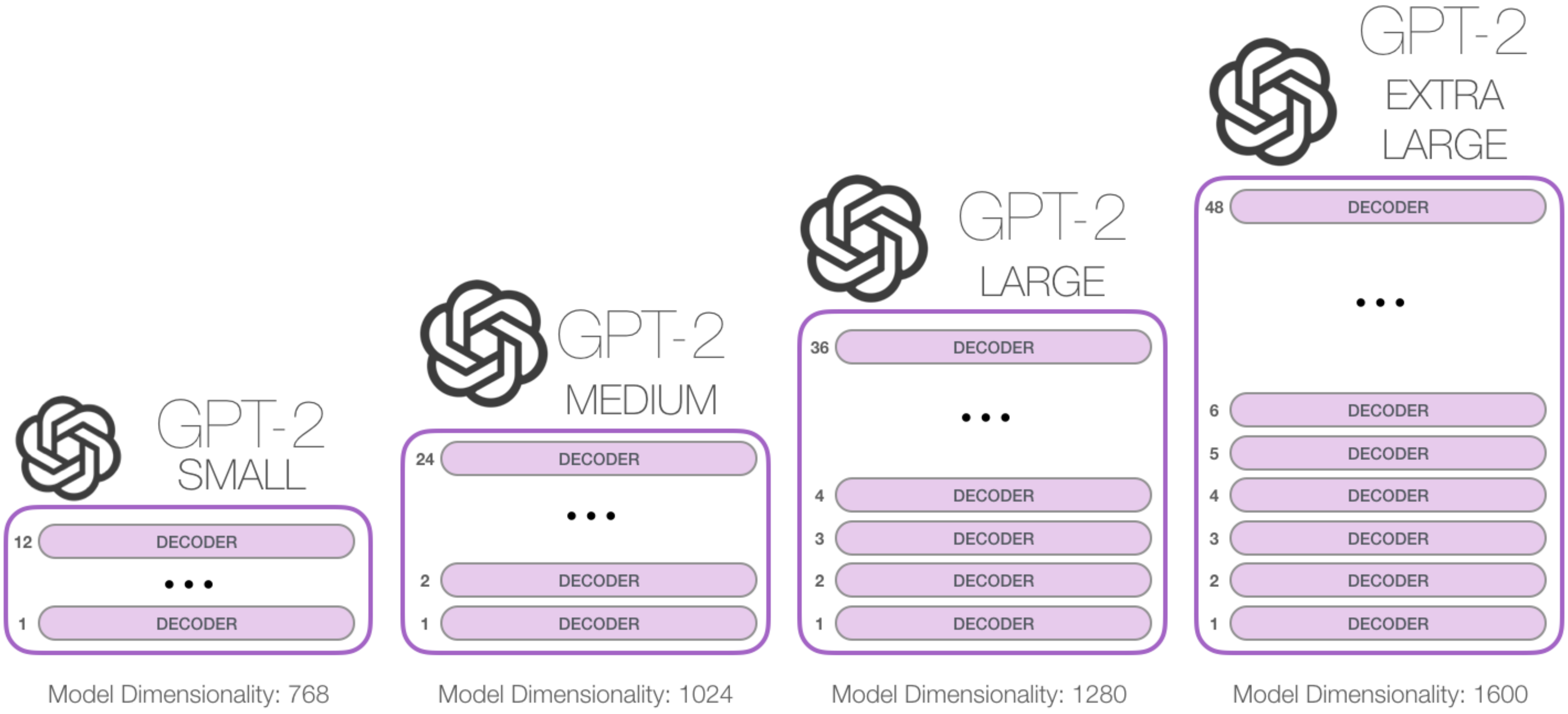
- RoBERTa was trained on a combination of in-domain data (books and articles) and out-of-domain data (web data).

GPT

- The objective is to maximize the log likelihood of the next token in the sequence:

$$\max_\theta \sum_{t \in T} \log p(t | T_{<t}, \theta)$$

GPT Evolution



Tokenization and Dataset

- GPT-2 uses byte pair encoding (BPE) for handling subword tokenization, allowing the model to handle rare and unseen words efficiently.
- GPT-2 was trained on a diverse range of internet text data, encompassing a wide array of sources to ensure a broad understanding of human language.
- Other higher order GPTs follow similar tokenization . Datasets are not disclosed.

BART

- BART is trained as a denoising autoencoder, where the model is tasked with reconstructing the original text from a corrupted version. Its objective is to minimize the reconstruction error, which can be formulated as:

$$\min_\theta \sum_{T \in D} \mathbb{E}_{\tilde{T} \sim \mathrm{Corrupt}(T)} [-\log p(T | \tilde{T}, \theta)]$$

~

Tokenization and Dataset

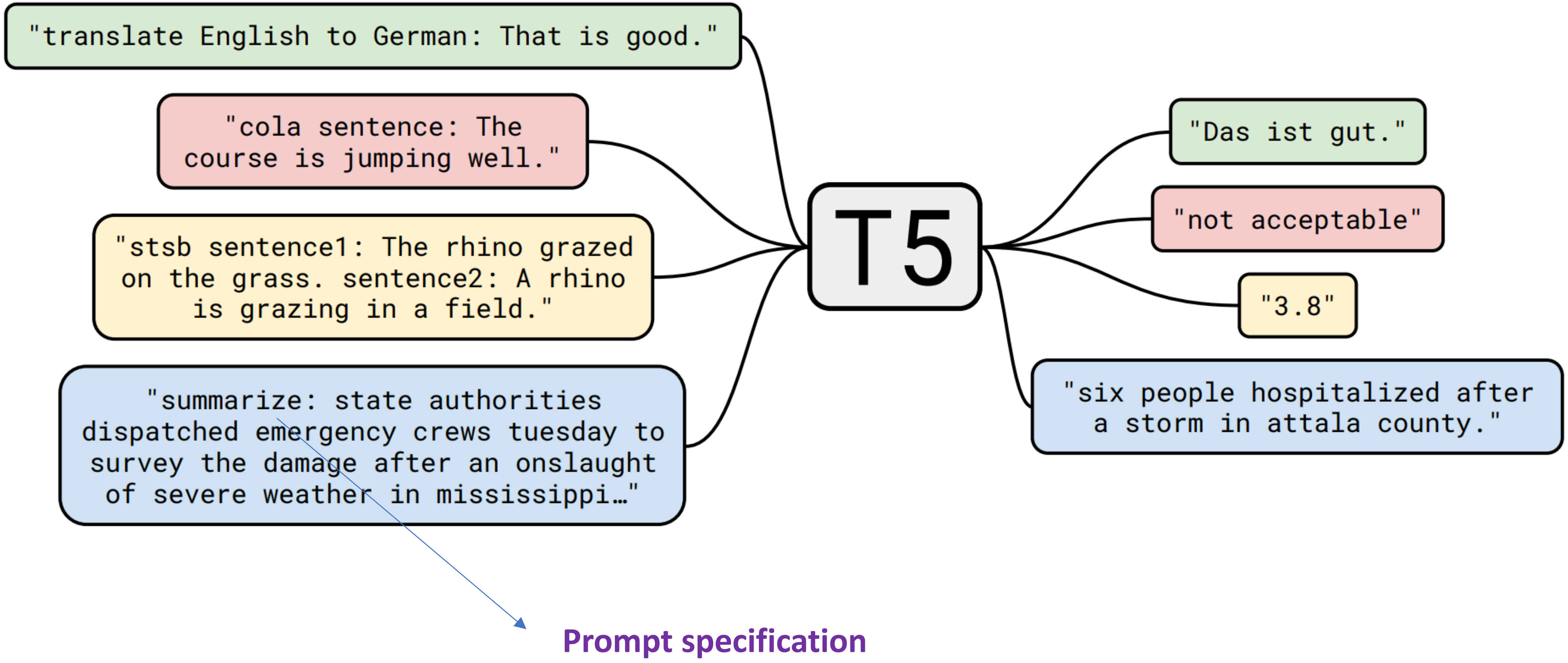
- BART employs a combination of Byte Pair Encoding (BPE) and learned positional embeddings to handle tokenization.
- BART was trained on a mixture of text from a wide range of sources, including books, articles, and websites.

T5

- T5 formulates all tasks as text-to-text problems, unifying different NLP tasks into a single framework. The objective function is to maximize the log likelihood of the target text given the input text:

$$\max_\theta \sum_{(X,Y)\in D} \log p(Y|X,\theta)$$

T5: Treating all problems as a language modeling task



Tokenizers and datasets

- T5 employs a variant of the Byte Pair Encoding (BPE) algorithm for subword tokenization.
- T5 was trained on a diverse corpus, including books, articles, and websites, ensuring a broad understanding of human language.

Fine tuning pre-trained models

- Fine-tuning involves taking a pre-trained model and adapting it to a specific task
- Generic steps:
  - **Select Pre-Trained Model:**
    - E.g. bert-base-uncased for lower-cased data, bert-base-cased for true cased data
    - bert-base-multilingual for multilingual tasks (e.g., Translation)
  - **Data Preparation**
  - **Model Architecture Modification (if necessary)**
  - **Initialize Parameters with the pretrained model**

Fine tuning pre-trained models

- Fine-tuning involves taking a pre-trained model and adapting it to a specific task
- Generic steps:
  - **Fine tuning Process:** Train the model on the task-specific dataset while monitoring its performance. Print train and validation loss

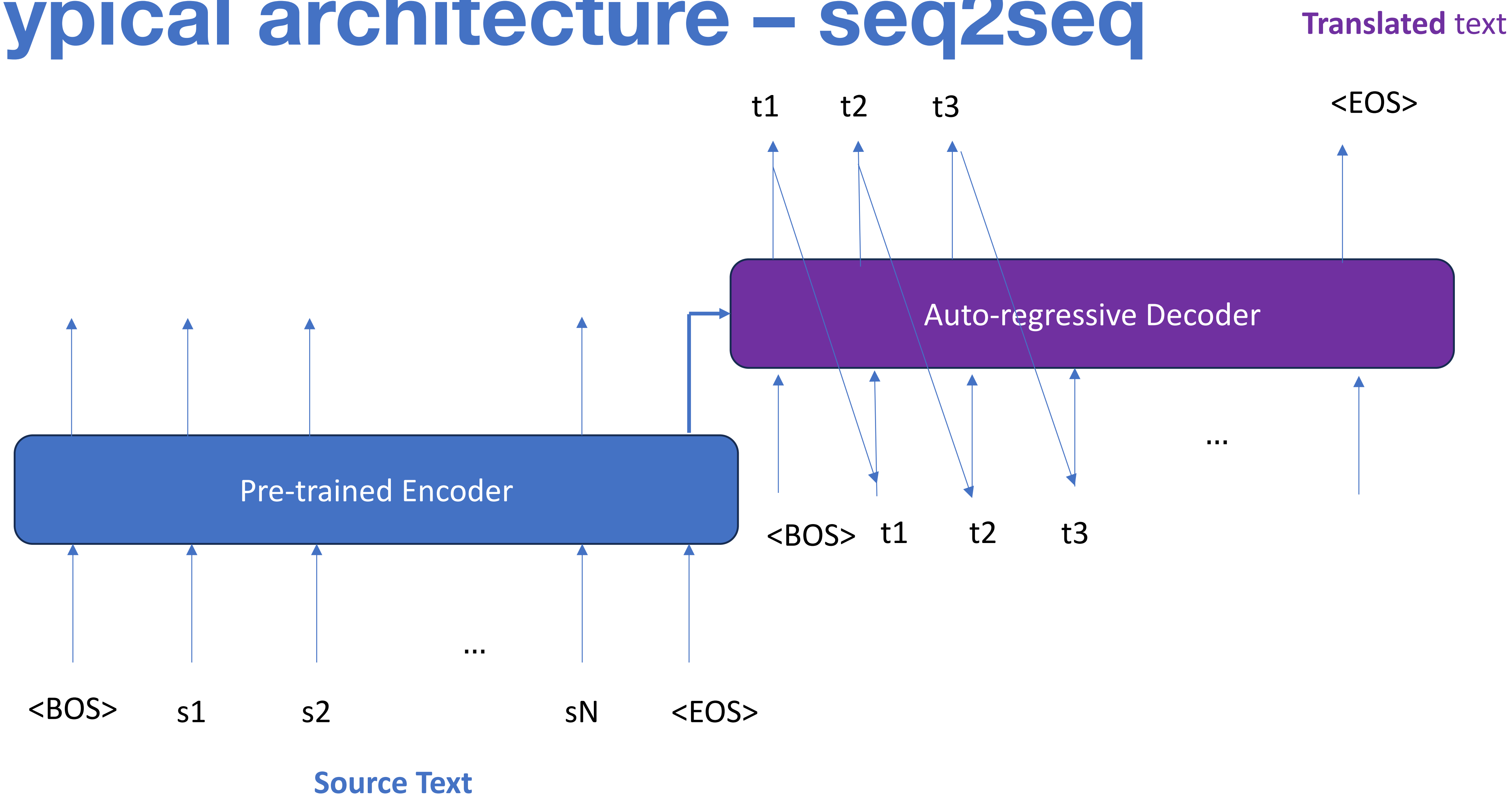
  - **Hyperparameter Tuning (if necessary):** Fine-tune the hyperparameters if the model's performance is not satisfactory. This might involve adjusting the learning rate, batch size, or other optimization parameters.
  - **Testing and Deployment**

Downstream Tasks – Machine Translation

- **Machine Translation**
  - The T5 model by Google has been widely used for various NLP tasks, including machine translation. It can be fine-tuned for specific translation tasks, making it an effective choice for this purpose

  - **Fairseq** provides pre-trained models like **Transformer**, **Transformer Big**, and **Transformer WMT19**, which are commonly used for machine translation tasks.

Typical architecture – seq2seq



Downstream Tasks – Machine Translation

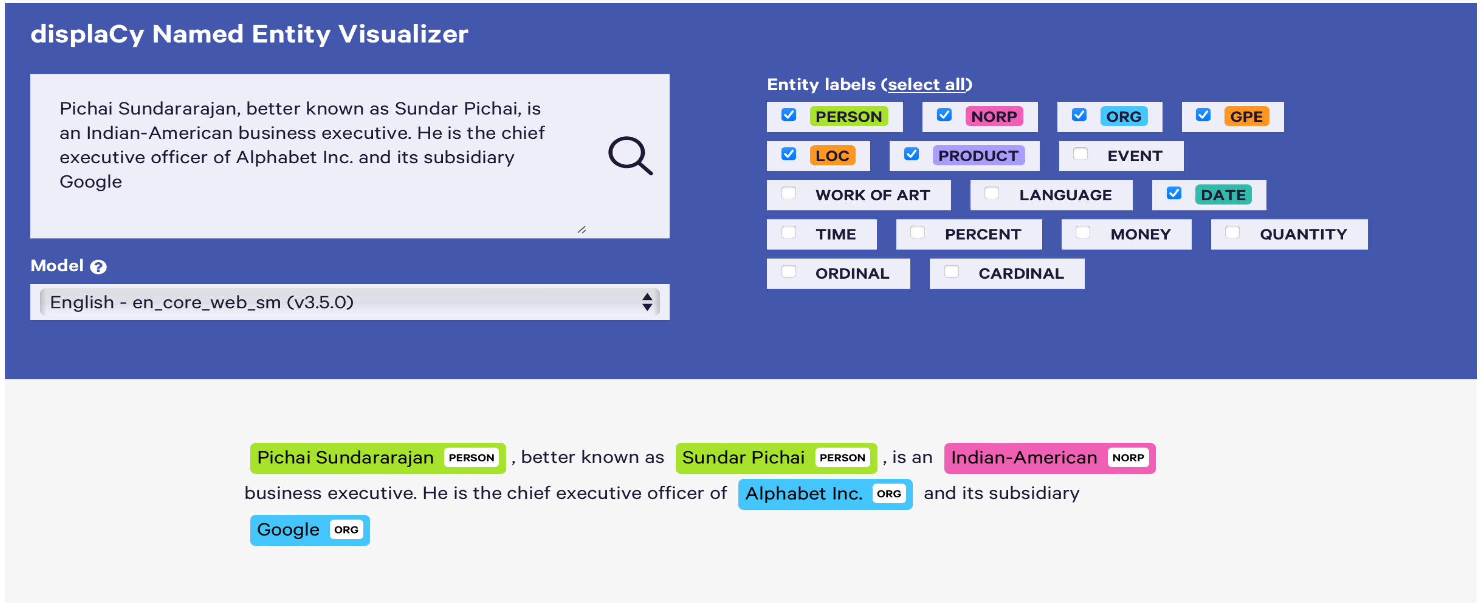
- **Datasets**:
  - WMT (Workshop on Machine Translation) datasets, including WMT14, WMT16, and WMT19.
  - IWSLT (International Workshop on Spoken Language Translation) datasets.
  - Multi30k dataset.
  - TED Talks dataset.
- **Evaluation Metrics**:
  - BLEU (Bilingual Evaluation Understudy): Measures the quality of machine-translated text by comparing it to one or more reference translations.
  - METEOR (Metric for Evaluation of Translation with Explicit Ordering): Considers unigram matching, stem matching, and synonymy.

Downstream Tasks – Summarization

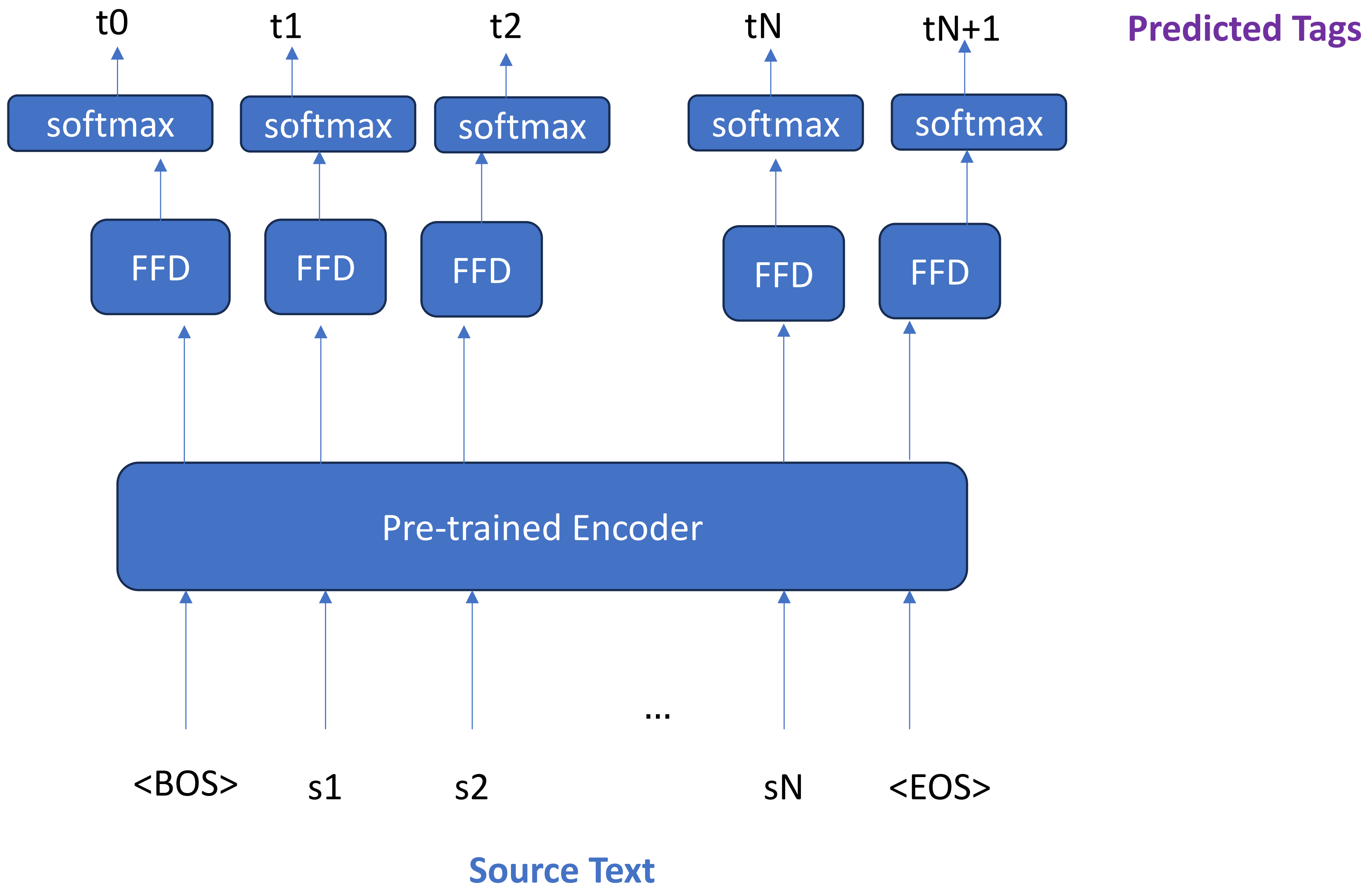
- **Datasets**:
  - CNN/Daily Mail dataset.
  - XSum dataset.
  - Gigaword dataset.
  - Newsroom dataset.
- **Evaluation Metrics**:
  - **ROUGE (Recall-Oriented Understudy for Gisting Evaluation):** Measures the overlap between the generated summary and the reference summaries at different levels (unigram, bigram, etc.).
  - **BLEU (Bilingual Evaluation Understudy):** Often used to evaluate the quality of generated summaries by comparing them to one or more reference summaries.

Sequence Tagging –

- **Tagging each input token with a class label**
  - **Example:** Part of Speech tagging
  - Named Entity Recognition



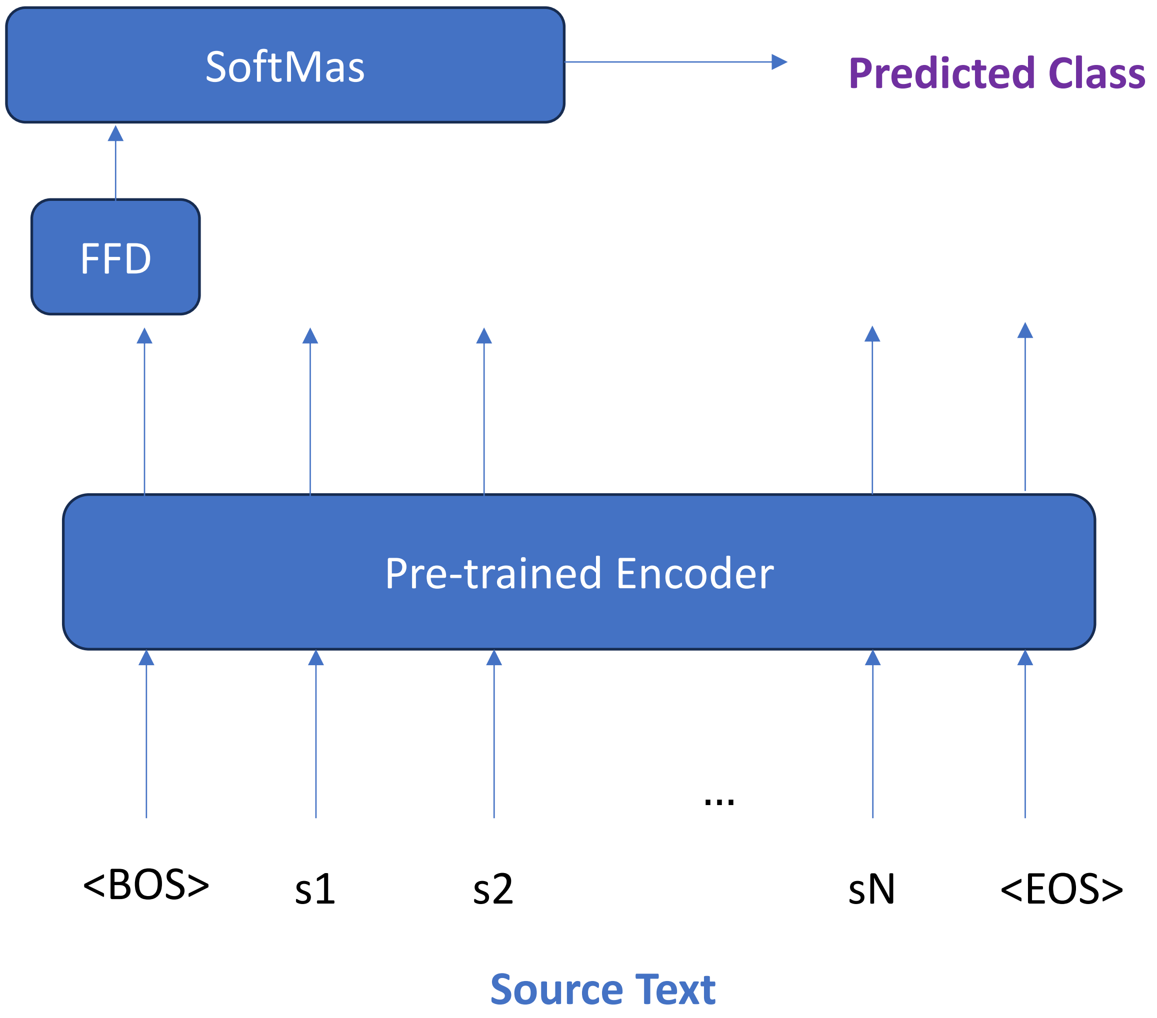
Typical architecture – sequence labeling



Sequence Tagging – NER

- **Datasets**:
  - CoNLL 2003 dataset.
  - OntoNotes dataset.
  - GermEval dataset.
  - ACE (Automatic Content Extraction) dataset.
- **Evaluation Metrics**:
  - Precision, Recall, and F1-score: Commonly used to evaluate the performance of named entity recognition systems by comparing the predicted entities to the ground truth entities.
  - CoNLL score: Used to evaluate the overall performance of a named entity recognition system, combining precision and recall into a single metric.

Typical architecture – text classification



Considering only the context vector from first input is enough

Text Classification

- **Datasets**:
  - IMDB Movie Reviews dataset.
  - AG News dataset.
  - Yelp Reviews dataset.
  - DBpedia dataset.
- **Evaluation Metrics**:
  - Accuracy: Measures the proportion of correctly classified instances.
  - Precision, Recall, and F1-score: Used to evaluate the performance of the classification model, particularly in tasks where class imbalance is present.

Next class

Fine-tuning Language Models for Building Classification and Generation Systems