AI Module: Transformer Models

Deepak Subramani
Assistant Professor
Dept. of Computational and Data Science
Indian Institute of Science Bengaluru

Outline for Week 03

- Part 01: BERT Models for Discriminative Tasks (NLU)
  - BERT and friends – the essential foundational model for text tasks
  - How is BERT trained?
  - What can BERT be used for with examples
  - Assignment on BERT
- Part 02: Transformer Decoder
  - Intro to Neural Machine Translation
  - Masked attention
  - Understanding Decoder through animations
  - Assignment on decoder and NMT

Recap of Transformer Encoder

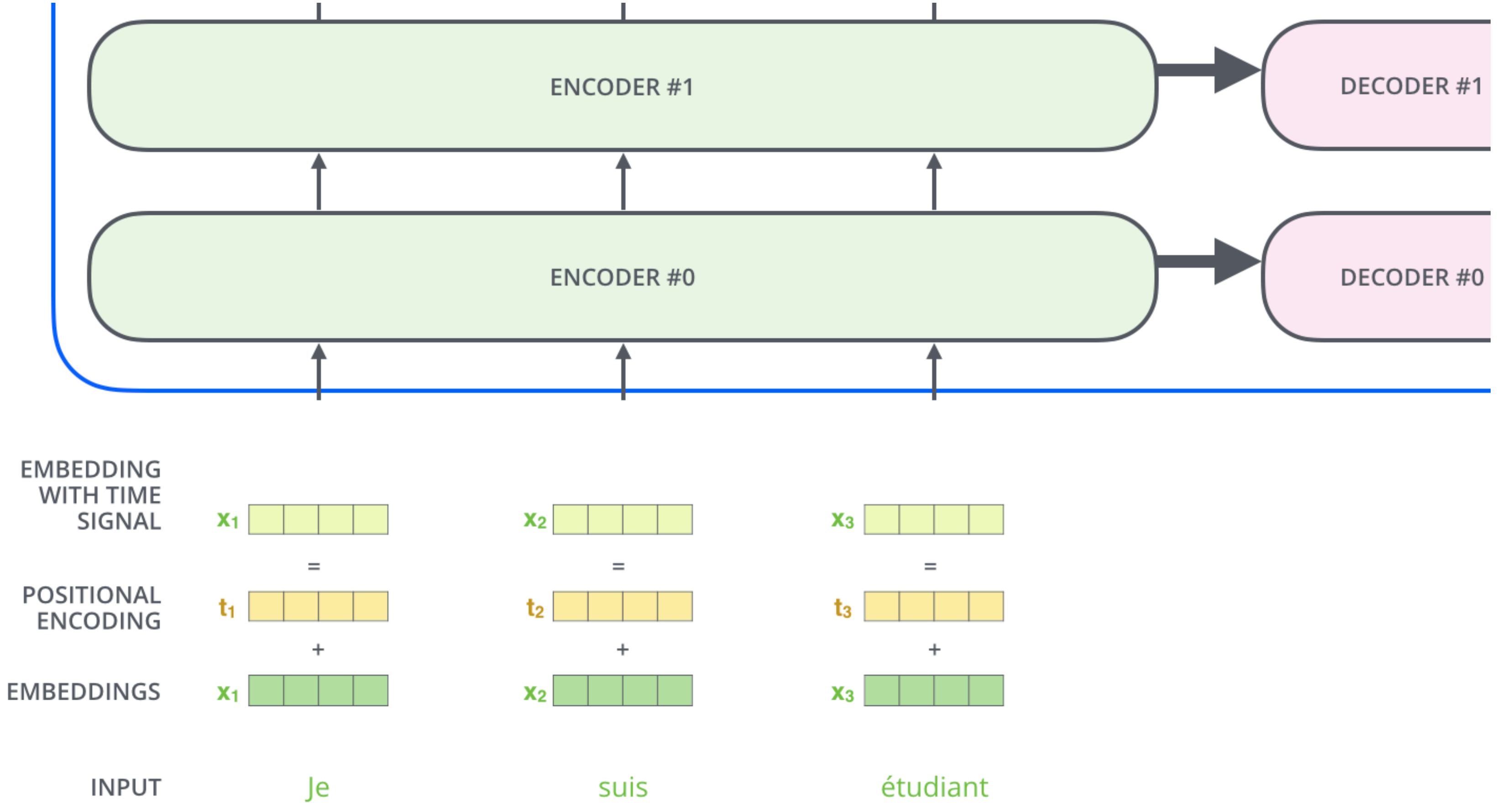


Image source: https://jalammar.github.io/illustrated-transformer/

Department of Computational and Data Sciences

भारतीय विज्ञान संस्थान

INDIAN INSTITUTE OF SCIENCE

Language Models

- Language Model (LM) is a probabilistic model of the natural language
- It understands the statistical relationship between words and can be used to fill missing words and complete sentences
- Language Models are trained by artificially introducing missing words and creating sentence completion tasks
- This procedure is called as “Pretraining” as the LM so learnt can be used with fine tuning for other particular tasks
- Large Language Models are usually transformer based neural models with a large number of parameters that are pretrained on a large number of tokens using masked model pretraining

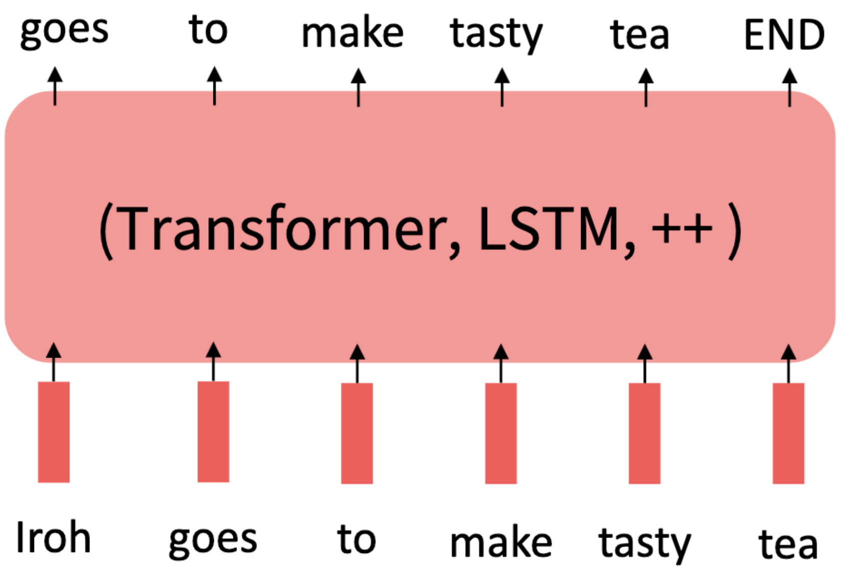
What is learned through pretraining?

- IISc is located in __________, India. [Trivia]
- He put ___ fork down on the table. [syntax]
- Naruto is the hokage of the village and ______ feels responsible for all the inhabitants. [coreference]
- I went to the ocean to see the fish, turtles, seals, and _____. [lexical semantics/topic]
- Miyazaki has created a complete, complex world with this film, and it's certainly a magical journey. The movie was ___. [sentiment]
- San went into the kitchen to make some tea. Standing next to San, Ashitaka pondered his destiny. Ashitaka left the ______. [some reasoning]

Pretraining and Finetuning

Step 1: Pretrain (on language modeling)

Lots of text; learn general things!



Step 2: Finetune (on your task)

Not many labels; adapt to the task!

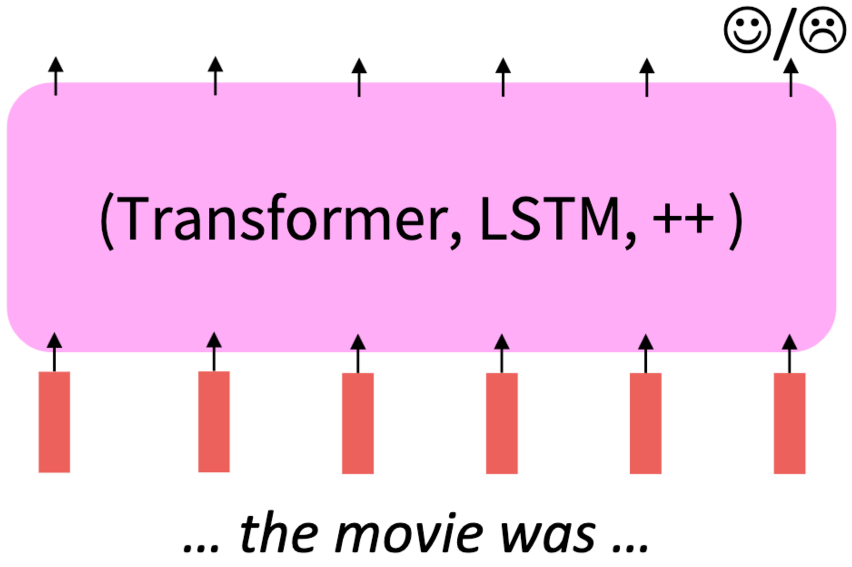


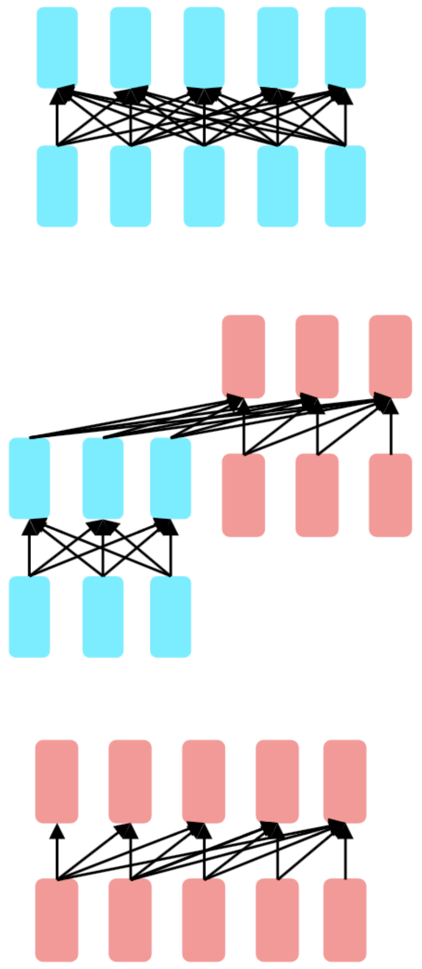
Image source: https://web.stanford.edu/class/cs224n/slides/cs224n-2023-lecture9-pretraining.pdf

Gradient Descent

- Pretraining provides parameters, $\hat{\theta}$ by minimizing the pre-training loss.
- Finetuning approximates $min_\theta \mathcal{L}_{finetune}(\theta)$, starting at $\hat{\theta}$.
- The pretraining may matter because stochastic gradient descent sticks (relatively) close to $\hat{\theta}$ during finetuning.
  - Finetuning local minima near $\hat{\theta}$ tend to generalize well.
  - Gradients of finetuning loss near $\hat{\theta}$ propagate nicely.

Types of pre-trained LMs

- Encoder only: bidirectional context
- Encoder-decoder: combining goodness of both
- Decoder only: doesn’t have bidirectional context.



Encoders

Encoder-Decoders

Decoders

Image source: https://web.stanford.edu/class/cs224n/slides/cs224n-2023-lecture9-pretraining.pdf

Pretraining Encoders

- Encoders have bidirectional context.
- Masked Language Model:
  - A small percentage of the input tokens are masked at random,
  - The model is trained to predict the masked tokens.
- The final hidden states (encoded representations) corresponding to masked tokens are fed into an output SoftMax over the entire vocabulary.

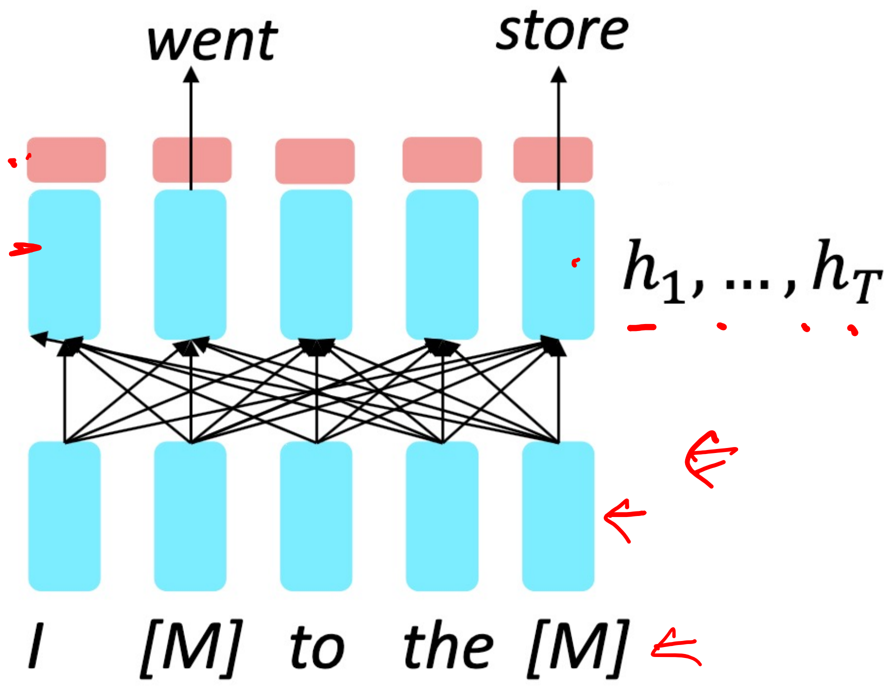


Image source: https://web.stanford.edu/class/cs224n/slides/cs224n-2023-lecture9-pretraining.pdf

[Devlin et al., 2018]()

Bidirectional Encoder Representations from Transformers (BERT) pre-training with Masked Language Modeling (MLM)

- Mask 15% of the WordPiece tokens in each sequence at random.
- The model only predicts masked words.
- Mismatch between pre-training and fine-tuning, since the [MASK] token does not appear during fine-tuning.
- Predict a random 15% of (sub)word tokens.
  - Replace input word with [MASK] 80% of the time.
  - Replace input word with a random token 10% of the time.
  - Leave input word unchanged 10% of the time.

Devlin et al., 2018

BERT - MLM

- Unlabelled sentence: my dog is hairy.
- Random masking: 4-th token (which corresponds to hairy)
  - 80% of the time: Replace the word with the [MASK] token, e.g., my dog is hairy → my dog is [MASK].
  - 10% of the time: Replace the word with a random word, e.g., my dog is hairy → my dog is apple. (Random replacement is just 10% of 15%. Empirically, this doesn't seem to affect model's understanding capability).
  - 10% of the time: Keep the word unchanged, e.g., my dog is hairy → my dog is hairy. The purpose of this is to bias the representation towards the actual observed word.

Devlin et al., 2018

BERT - MLM

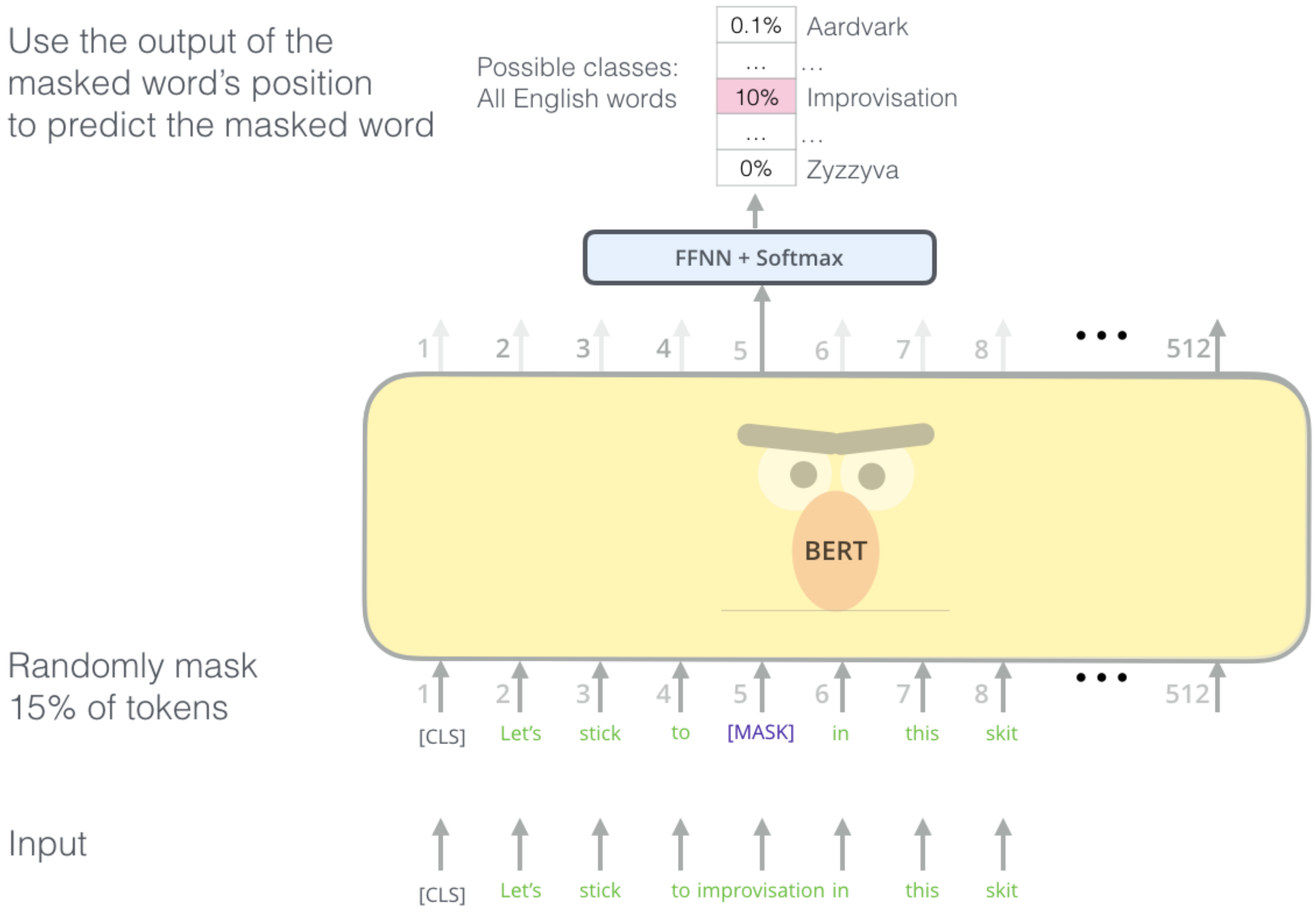


Image source: https://jalammar.github.io/illustrated-bert/

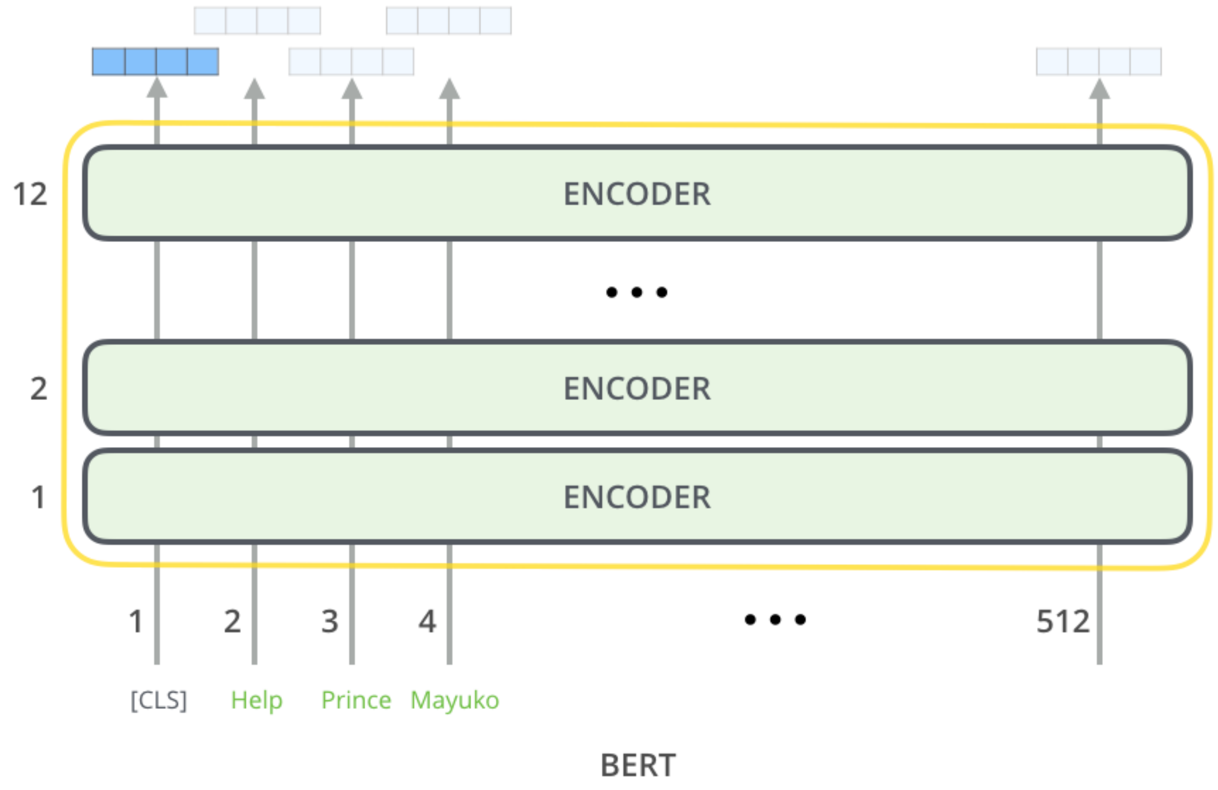
BERT - Specifics

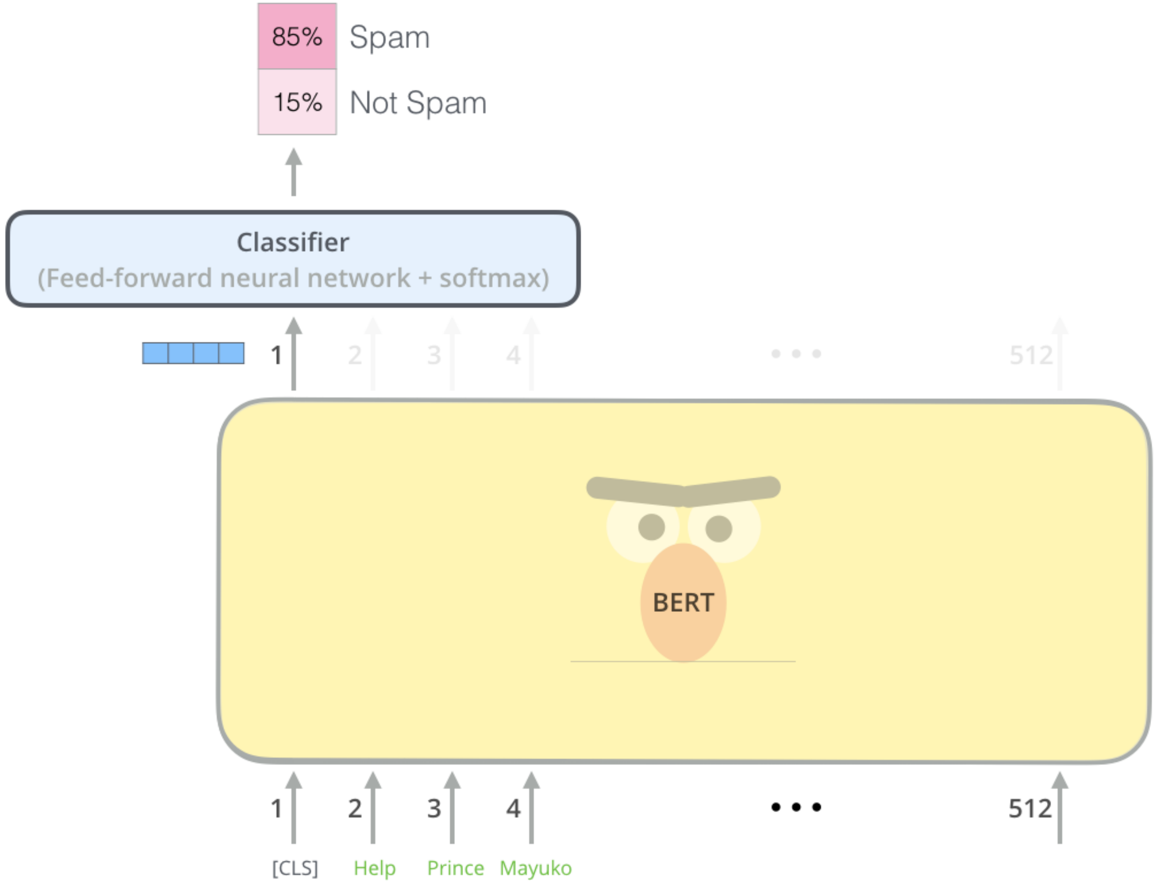
- Two models were released:
  - BERT-base: 512 sequence length , 12 layers, 768-dim hidden states, 12 attention heads, 110 million parameters.
  - BERT-large: 512 sequence length , 24 layers, 1024-dim hidden states, 16 attention heads, 340 million parameters.
- Trained on:
  - BooksCorpus (800 million words).
  - English Wikipedia (2,500 million words).
- Pretraining is expensive and impractical on a single GPU.
  - BERT was pretrained with 64 TPU chips for a total of 4 days.

BERT Finetuning

- For each task, task-specific inputs and outputs are plugged into BERT and all the parameters are finetuned end-to-end.
- At the input, sentence A and sentence B from pre-training are analogous to hypothesis-premise pairs in entailment, question-passage pairs in question answering etc.
- At the output, the token representations are fed into an output layer for token level tasks, such as sequence tagging or question answering, and the [CLS] representation is fed into an output layer for classification, such as entailment or sentiment analysis.

BERT for Text Classification





BERT Tasks

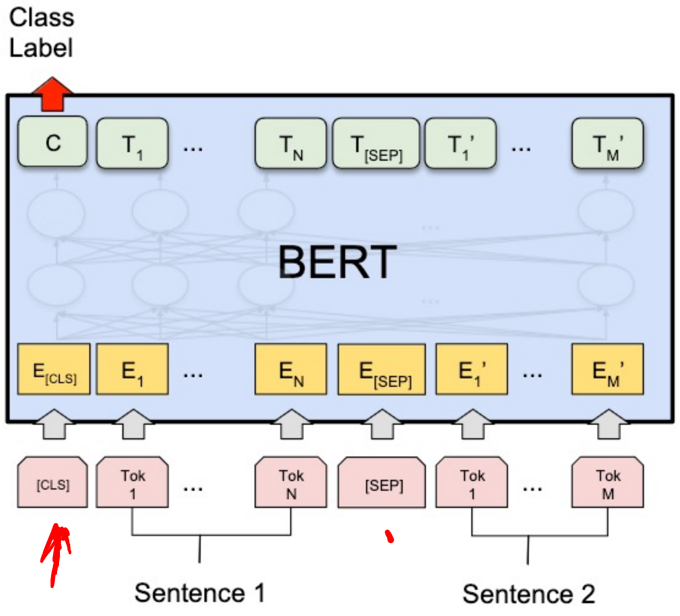
- MNLI (Multi-Genre Natural Language Inference):
  - large-scale, crowdsourced entailment classification task.
  - the goal is to predict whether the second sentence is an entailment, contradiction, or neutral with respect to the first one.
- QQP (Quora Question Pairs):
  - binary classification task where the goal is to determine if two questions asked on Quora are semantically equivalent.
- QNLI (Question Natural Language Inference):
  - version of the Stanford Question Answering Dataset which has been converted to a binary classification task.
  - positive examples are (question, sentence) pairs with the correct answers, and negative examples are pairs with no correct answer.

BERT Tasks

- SST-2 (The Stanford Sentiment Treebank):
  - binary single-sentence classification task consisting of sentences extracted from movie reviews with human annotations of their sentiment.
- CoLA (The Corpus of Linguistic Acceptability)
  - binary single-sentence classification task.
  - the goal is to predict whether an English sentence is linguistically "acceptable" or not.
- STS-B (The Semantic Textual Similarity Benchmark)
  - collection of sentence pairs drawn from news headlines and other sources.
  - annotated with a score from 1 to 5 denoting how semantic similarity.

BERT Tasks

- MRPC (Microsoft Research Paraphrase Corpus):
  - sentence pairs automatically extracted from online news sources, with human annotations for whether the sentences in the pair are semantically equivalent.
- RTE (Recognizing Textual Entailment)
  - binary entailment task like MNLI, but with much less training data.



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

NLP Pipeline

Pipelines generally consist of

1. Tokenizer: Convert raw text to tokens
2. Model: Take tokens to output of your task
3. Post-processing: Enhance the output

Look for a finetuned BERT for your task. See HuggingFace library for all the finetuned models for each task.

https://huggingface.co/transformers/v3.2.0/pretrained_models.html

BERT and Friends

- RoBERTa – train BERT for longer and remove NSP
- SpanBERT: Masking contiguous spans of words makes the pretraining harder and more useful

- Most straightforward way is to finetune every parameter

- Researchers have tried to finetune only some part of the model also

BERT Additional Material

- Original BERT Paper was pre-trained on NSP task in addition to MLM
- Later it was shown that MLM pre-training is sufficient and NSP is not improving the pre-trained model's performance
- In the next three slides, we provide the details of NSP for your reference.

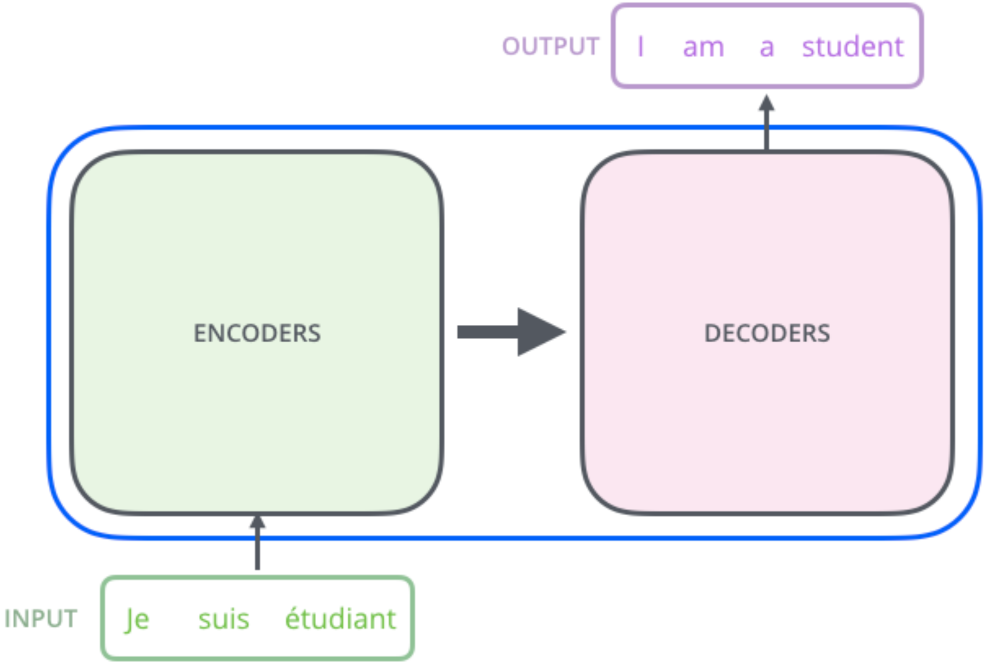
Department of Computational and Data Sciences

INDIAN INSTITUTE OF SCIENCE
भारतीय विज्ञान संस्थान

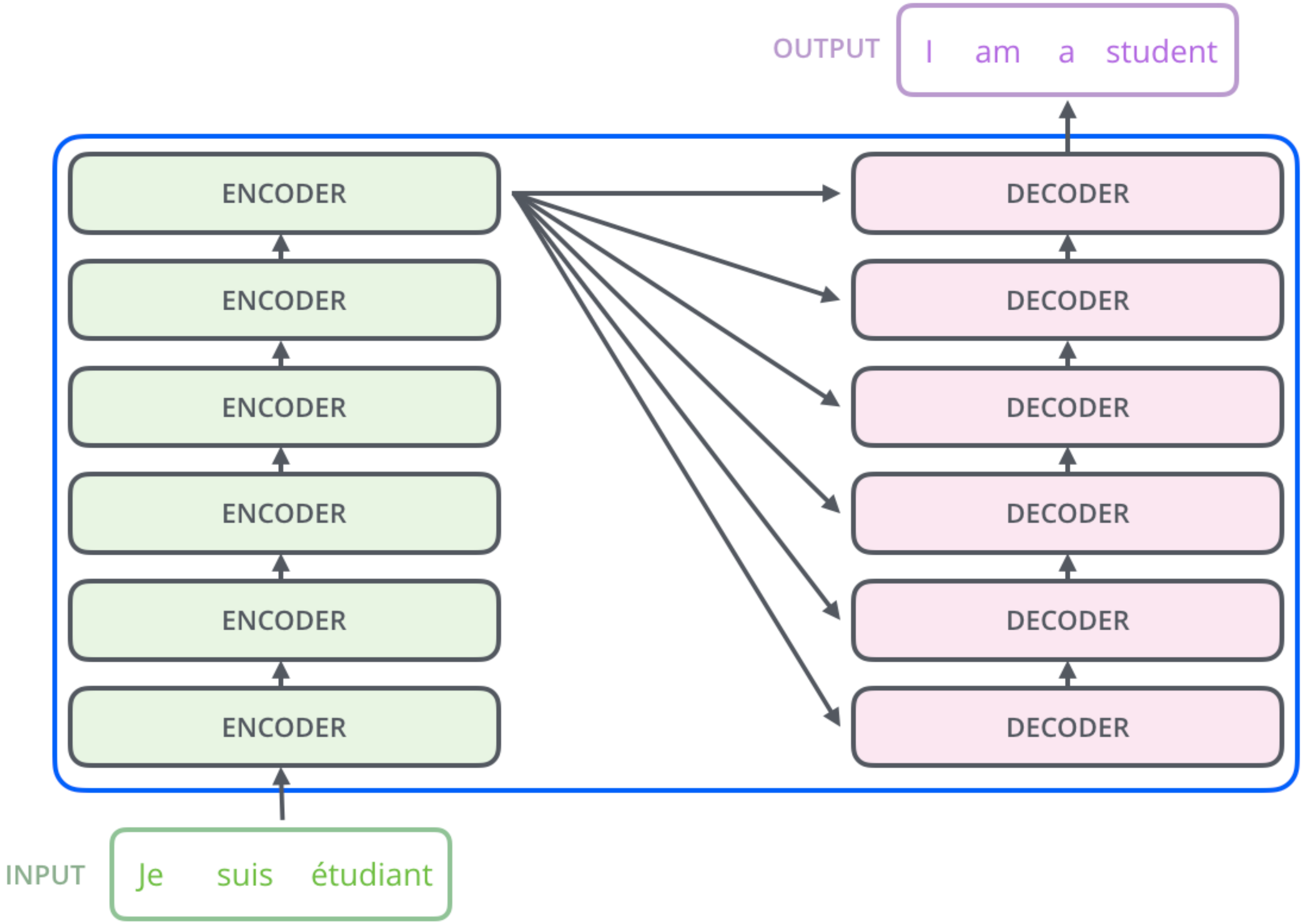
Outline for Week 03

- Part 01: BERT Models for Discriminative Tasks
  - BERT and friends – the essential foundational model for text tasks
  - How is BERT trained?
  - What can BERT be used for with examples
  - Assignment on BERT
- Part 02: Transformer Decoder
  - Intro to Neural Machine Translation
  - Masked attention
  - Understanding Decoder through animations
  - Assignment on decoder and NMT

Neural Machine Translation



Full Picture

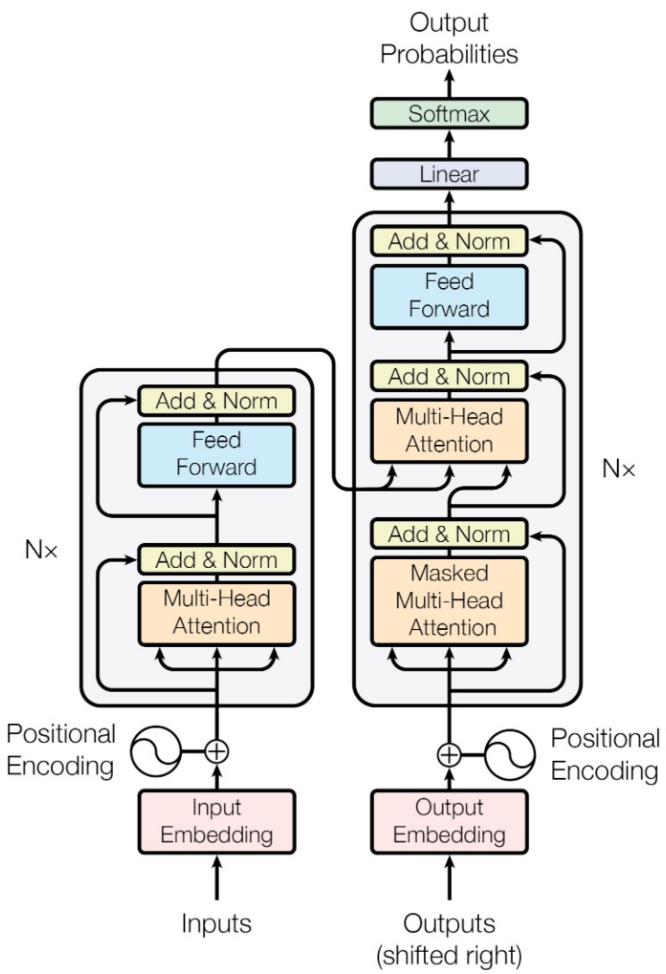


Neural Machine Translation Model

- The decoder input is shifted by one step, i.e., the decoder is given as input the word that it *should* have output at the previous step.
- For first prediction, the input is <SOS> token and decoder ends the sentence with <EOS> token.
- During inference, as the target sentence is not available the word that it output at the previous step is fed to the decoder.

NMT with Transformer: High Level

- This is a NMT model with 6 encoders and 6 decoders.
- Each encoder has a self-attention layer followed by a feedforward layer. Residual connection and layer normalization is applied in both the layers.
- Each decoder has an additional attention layer that takes in encoder output. Also, masked attention is used in Decoder to prevent attending to subsequent positions.

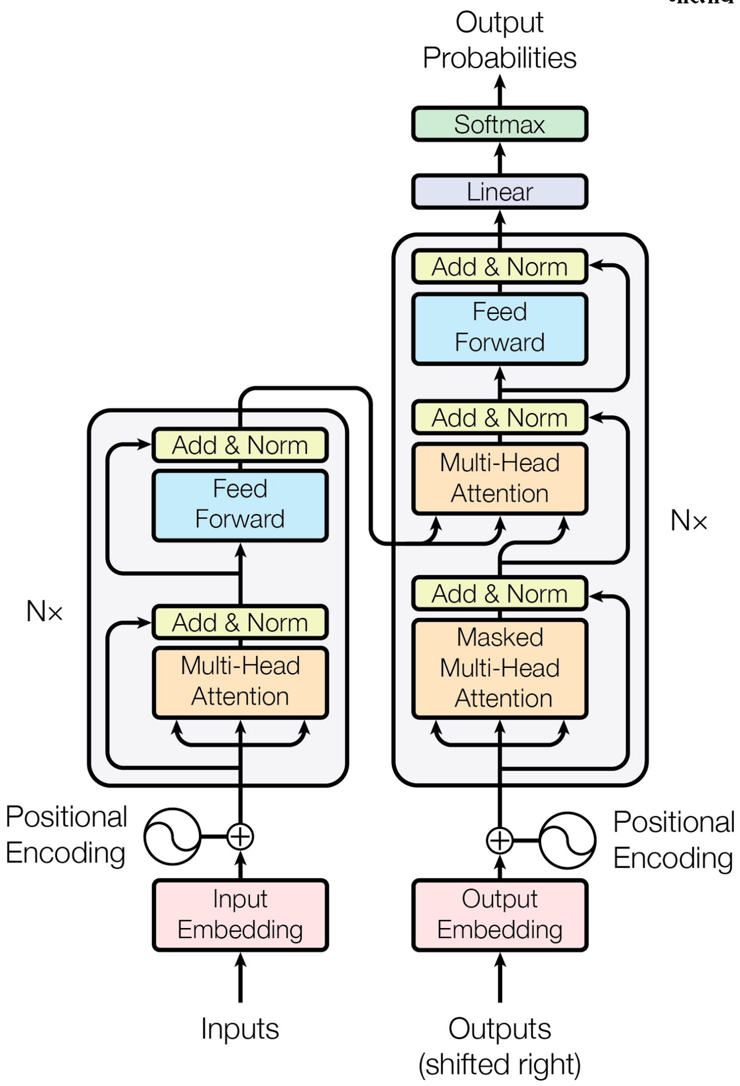


[Vaswani et al., 2017, Attention Is All You Need]()

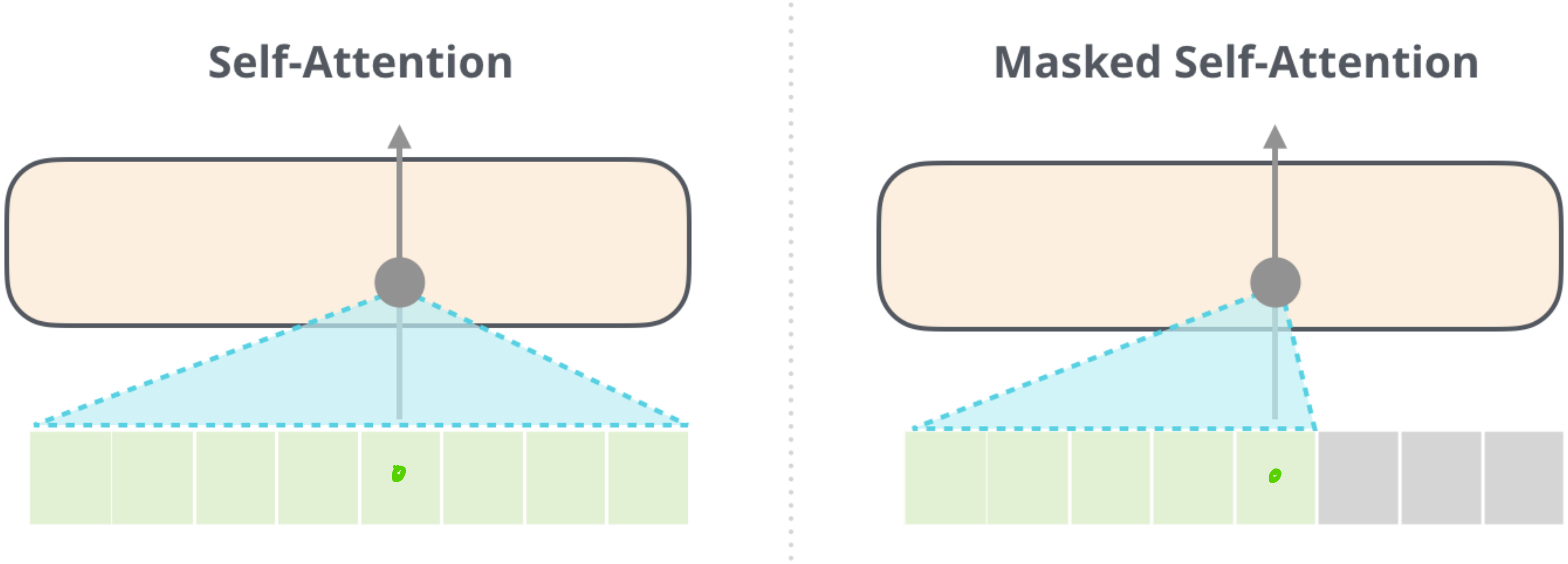
Sequence-to-Sequence using Transformers

Department of Computational and Data Sciences

- Right part of the figure – Decoder
  - The target sequence is shifted to the right by one and a <sos> is added
  - Output embedding is done similar to input embedding for the target language
  - Positional embedding is also added
- After embedding, a masked multi-head attention is used
  - A causal mask is used so that the output sequence doesn't use attention with tokens in later positions (at step n, it should not use tokens at n+1 onwards)
- After the masked MHA, the next multihead attention uses query=masked MHA output, key=value=encoder output
- The same encoder output is fed into all the decoder units in the right stack



Masked Self-Attention



Masked Self-Attention

Queries

robot	must	obey	orders

X

Keys

robot	must	obey	orders
robot	must	obey	orders
robot	must	obey	orders
robot	must	obey	orders

=

Scores
(before softmax)

0.11	0.00	0.81	0.79
0.19	0.50	0.30	0.48
0.53	0.98	0.95	0.14
0.81	0.86	0.38	0.90

Apply Attention Mask

Masked Scores
(before softmax)

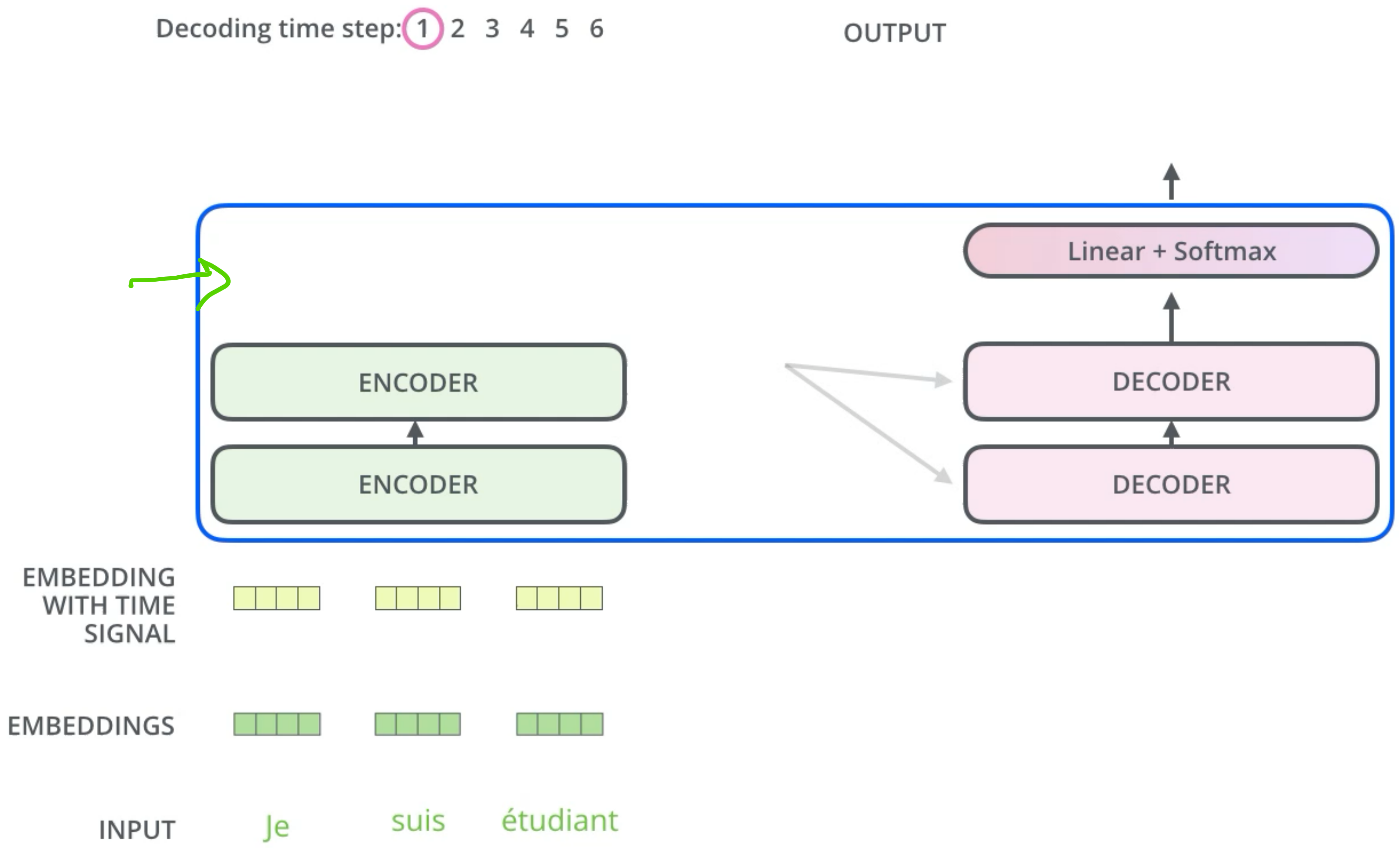
0.11	-inf	-inf	-inf
0.19	0.50	-inf	-inf
0.53	0.98	0.95	-inf
0.81	0.86	0.38	0.90

Scores

1	0	0	0
0.48	0.52	0	0
0.31	0.35	0.34	0
0.25	0.26	0.23	0.26

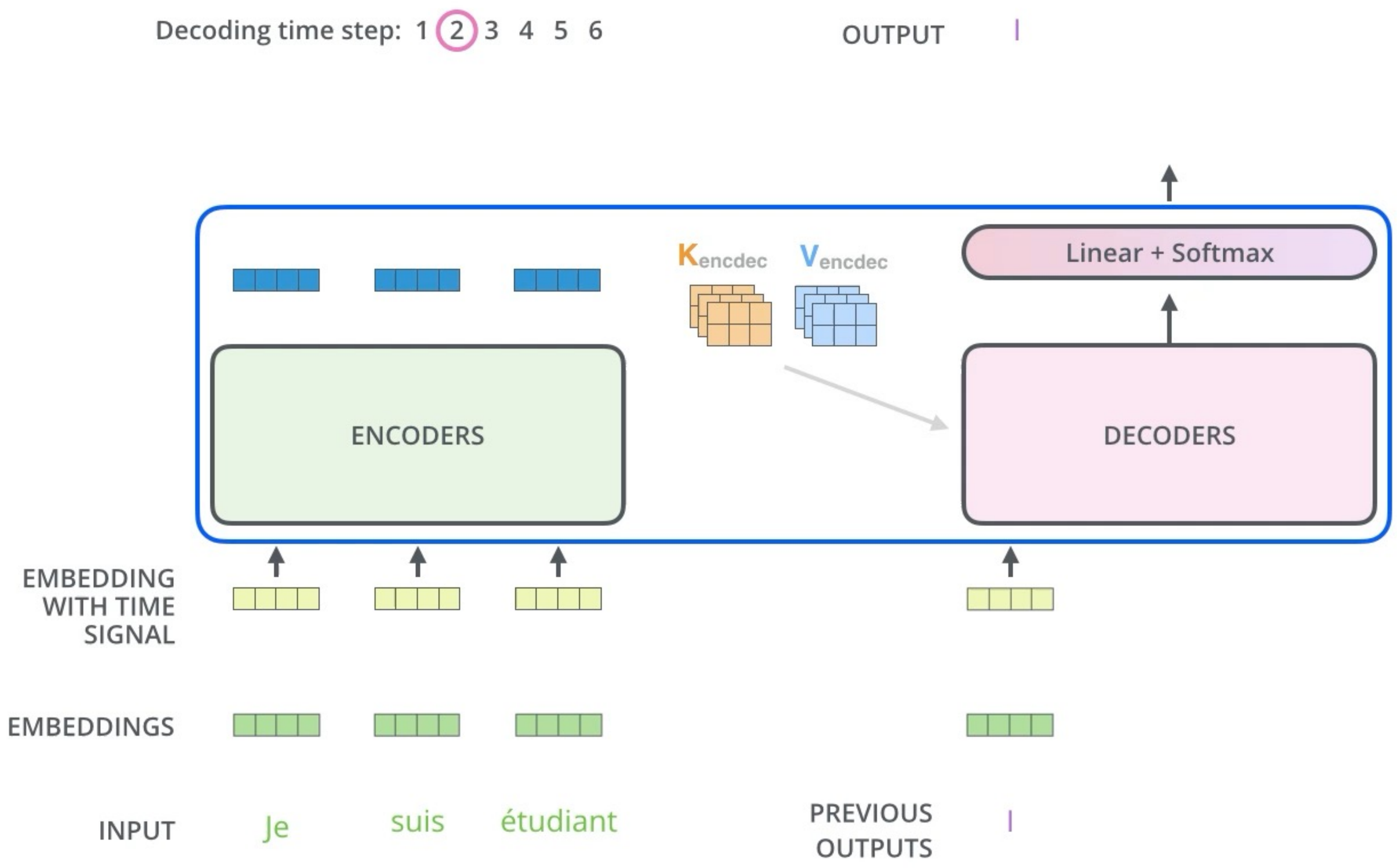
Image source: https://jalammar.github.io/illustrated-gpt2/

Seq2seq using Transformers: Decoder



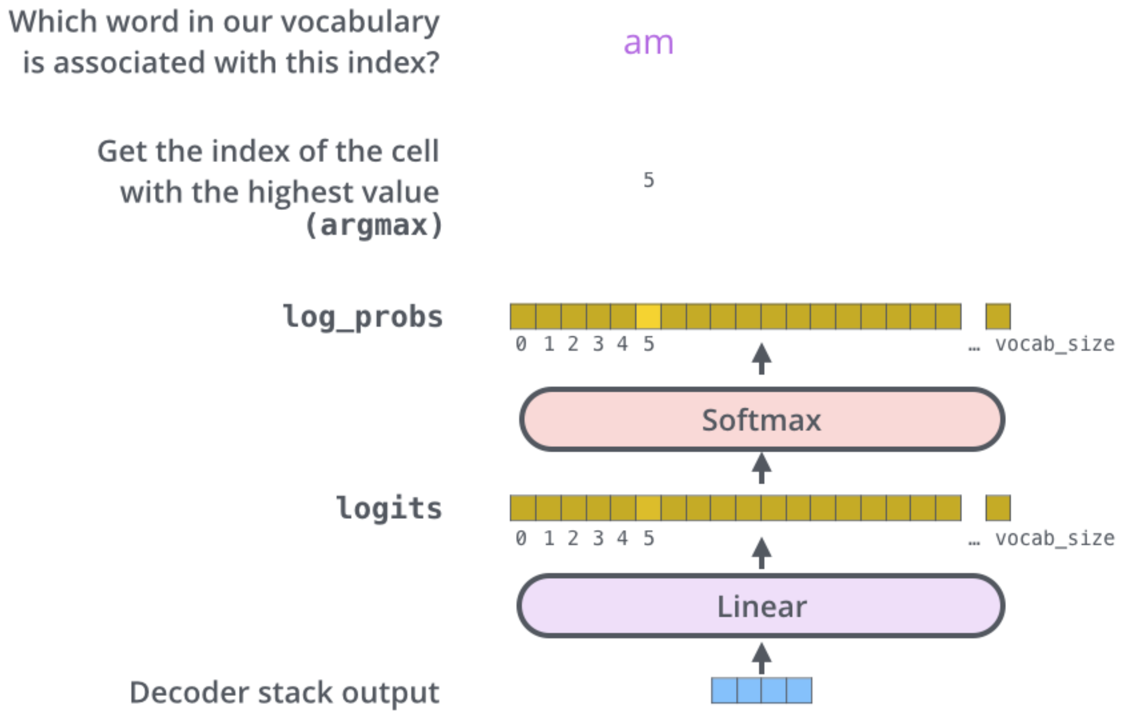
Video source: https://jalammar.github.io/illustrated-transformer/

Seq2seq using Transformers: Decoder

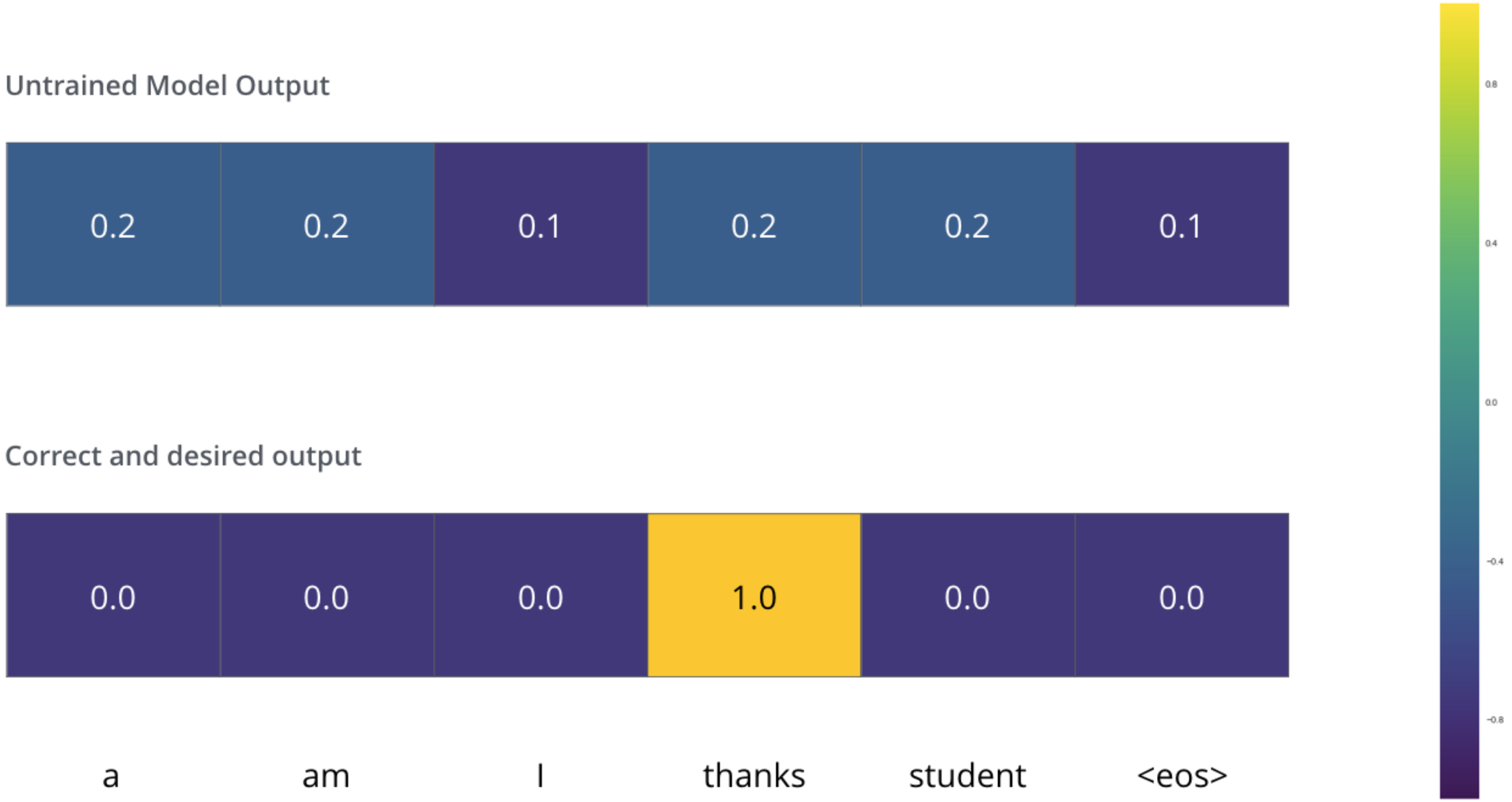


Video source: https://jalammar.github.io/illustrated-transformer/

Final Layer and Output



Loss Function



Cross Entropy Loss

Target Model Outputs

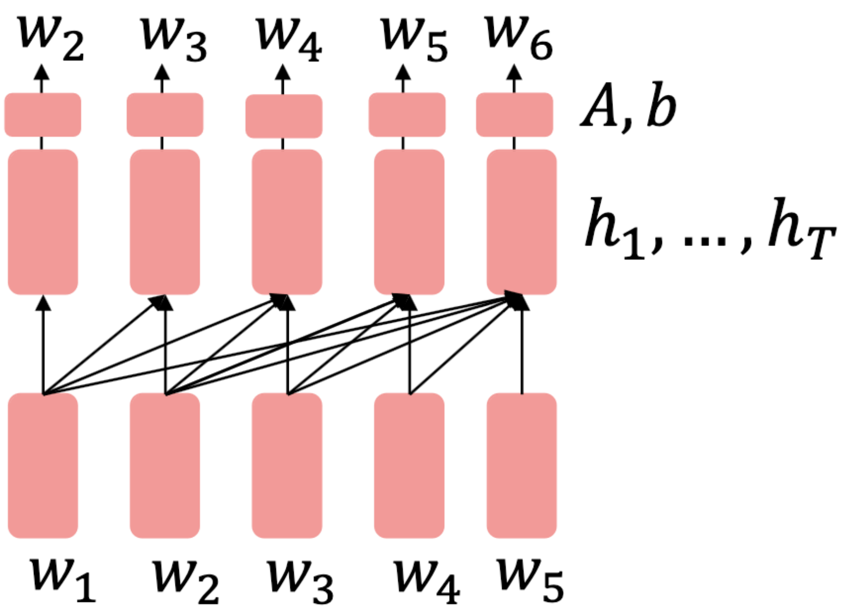
Output Vocabulary:	a	am	I	thanks	student	<eos>
position #1	0.0	0.0	1.0	0.0	0.0	0.0
position #2	0.0	1.0	0.0	0.0	0.0	0.0
position #3	1.0	0.0	0.0	0.0	0.0	0.0
position #4	0.0	0.0	0.0	0.0	1.0	0.0
position #5	0.0	0.0	0.0	0.0	0.0	1.0
	a	am	I	thanks	student	<eos>

Trained Model Outputs

Output Vocabulary:	a	am	I	thanks	student	<eos>
position #1	0.01	0.02	0.93	0.01	0.03	0.01
position #2	0.01	0.8	0.1	0.05	0.01	0.03
position #3	0.99	0.001	0.001	0.001	0.002	0.001
position #4	0.001	0.002	0.001	0.02	0.94	0.01
position #5	0.01	0.01	0.001	0.001	0.001	0.98
	a	am	I	thanks	student	<eos>

Decoders Pretraining

- Decoders generate the next word given its previous words.



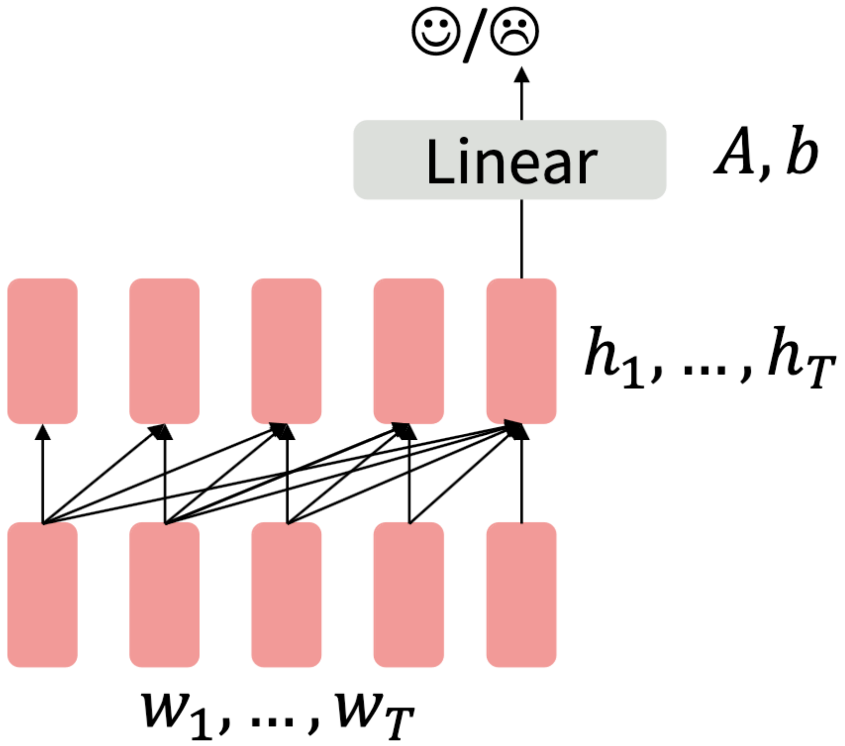


Image source: https://web.stanford.edu/class/cs224n/slides/cs224n-2023-lecture9-pretraining.pdf

Types of pre-trained LMs

- Encoder only: BERT and friends
- Encoder-decoder: Flan T5 Model
- Decoder only: GPT

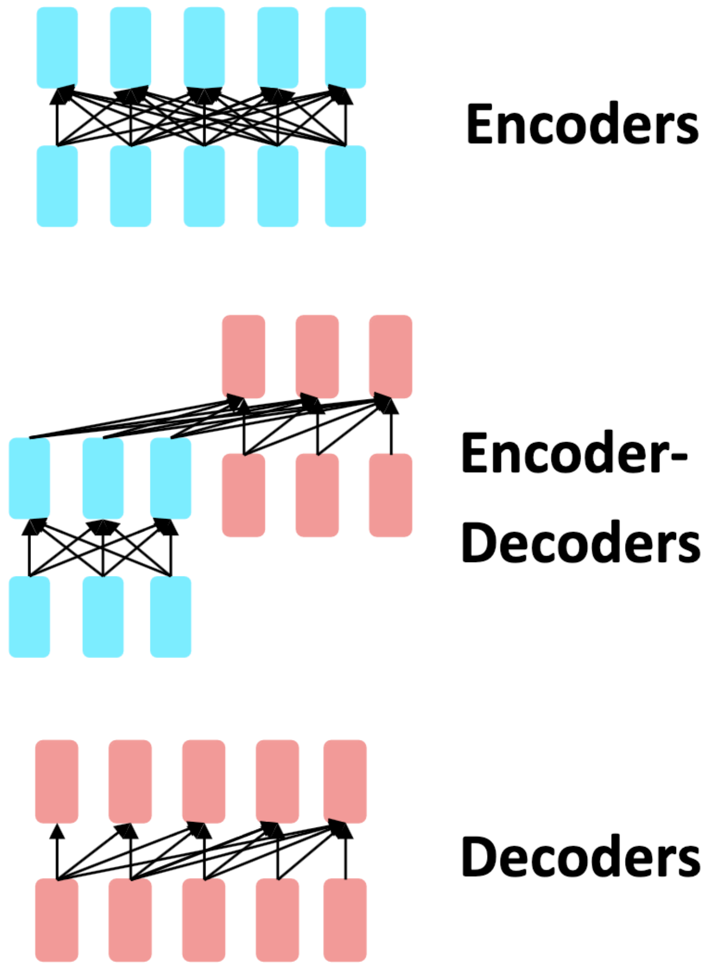


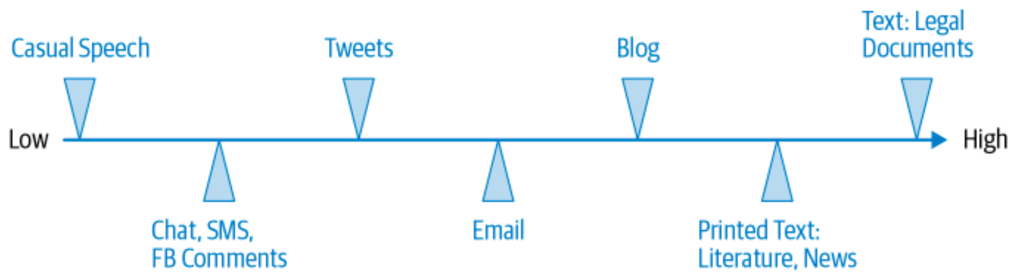
Image source: https://web.stanford.edu/class/cs224n/slides/cs224n-2023-lecture9-pretraining.pdf

Miscellaneous Points

- Social Media Data
- Evaluation Metrics
- Other NLP Tasks
- Chat Bots
- Speech Modeling

Subword Embeddings

- When dealing with Social Media Text Data (SMTD), we will have OOV tokens when using standard vocabulary
- Hence, subword embeddings are more popular for SMTD
- OSU Twitter NLP Tool: https://github.com/aritter/twitter_nlp/tree/master
- NLTK Tweet Special Tokenizer
- Twikenizer – can handle abusive hidden words https://pypi.org/project/twikenizer/



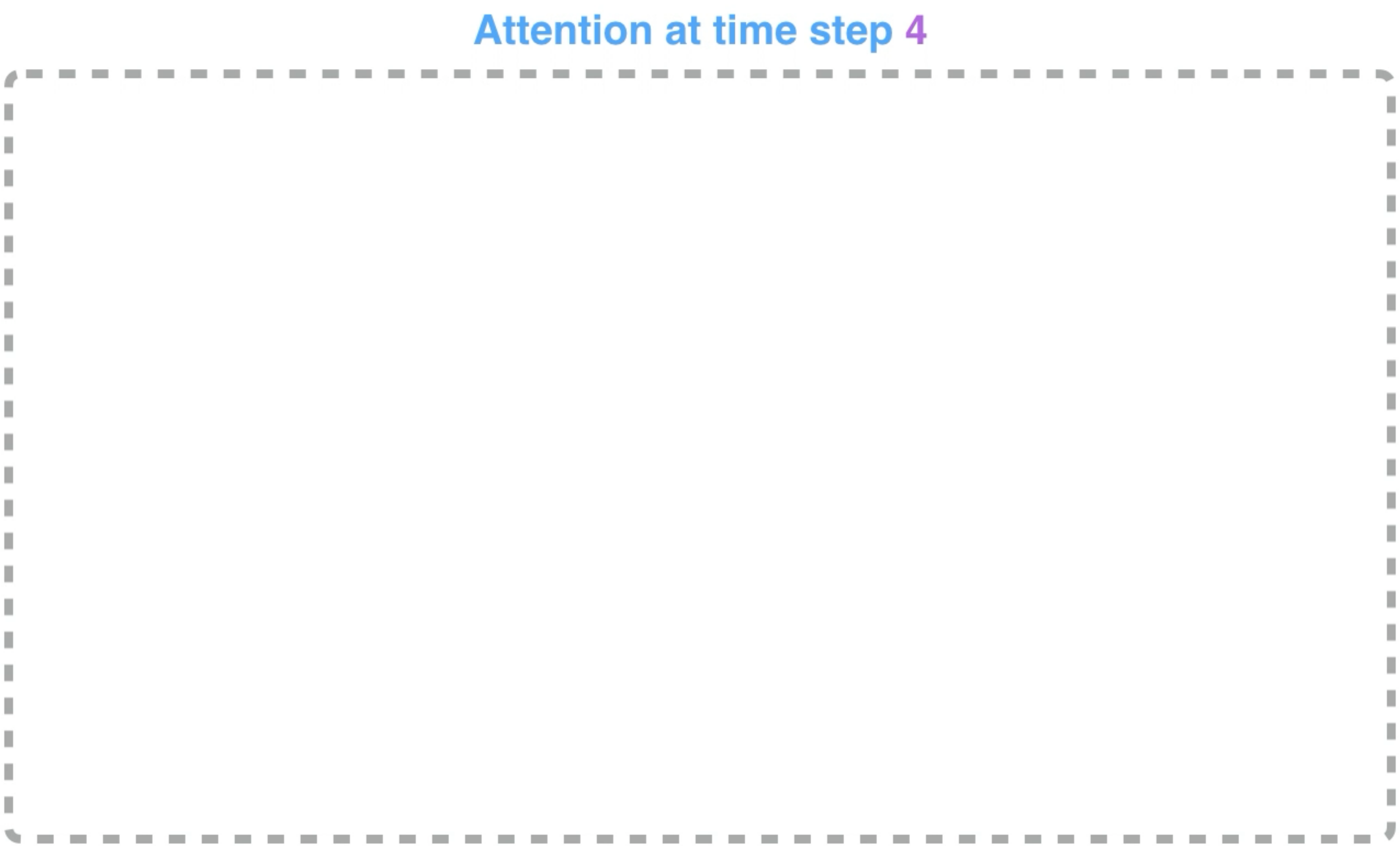
Evaluation Metrics

- Text Classification
  - Accuracy, Precision, Recall, F1 Score
  - Area Under the Curve (AUC) –
    - Combines true positives vs false positives as threshold for prediction is varied.
    - Used to measure the quality of a model independent of prediction threshold, and
    - to find the optimal prediction threshold for a classification task.
- Language Translation and other seq2seq tasks
  - BLEU - bilingual evaluation understudy.
    - Captures the amount of n-gram overlap between the output sentence and the reference ground truth sentence.
    - Has many variants
    - Been adapted to text to text tasks such as paraphrase generation and summarization.
  - METEOR – Advanced BLEU
    - Allows synonyms and stemmed words to be matched with the reference word
- Summarization Tasks
  - ROUGE – Measures Recall
    - Evaluate how many words a model can recall in a summary.
- Language Models
  - Perplexity – Cross entropy in next word prediction task

Other NLP Tasks

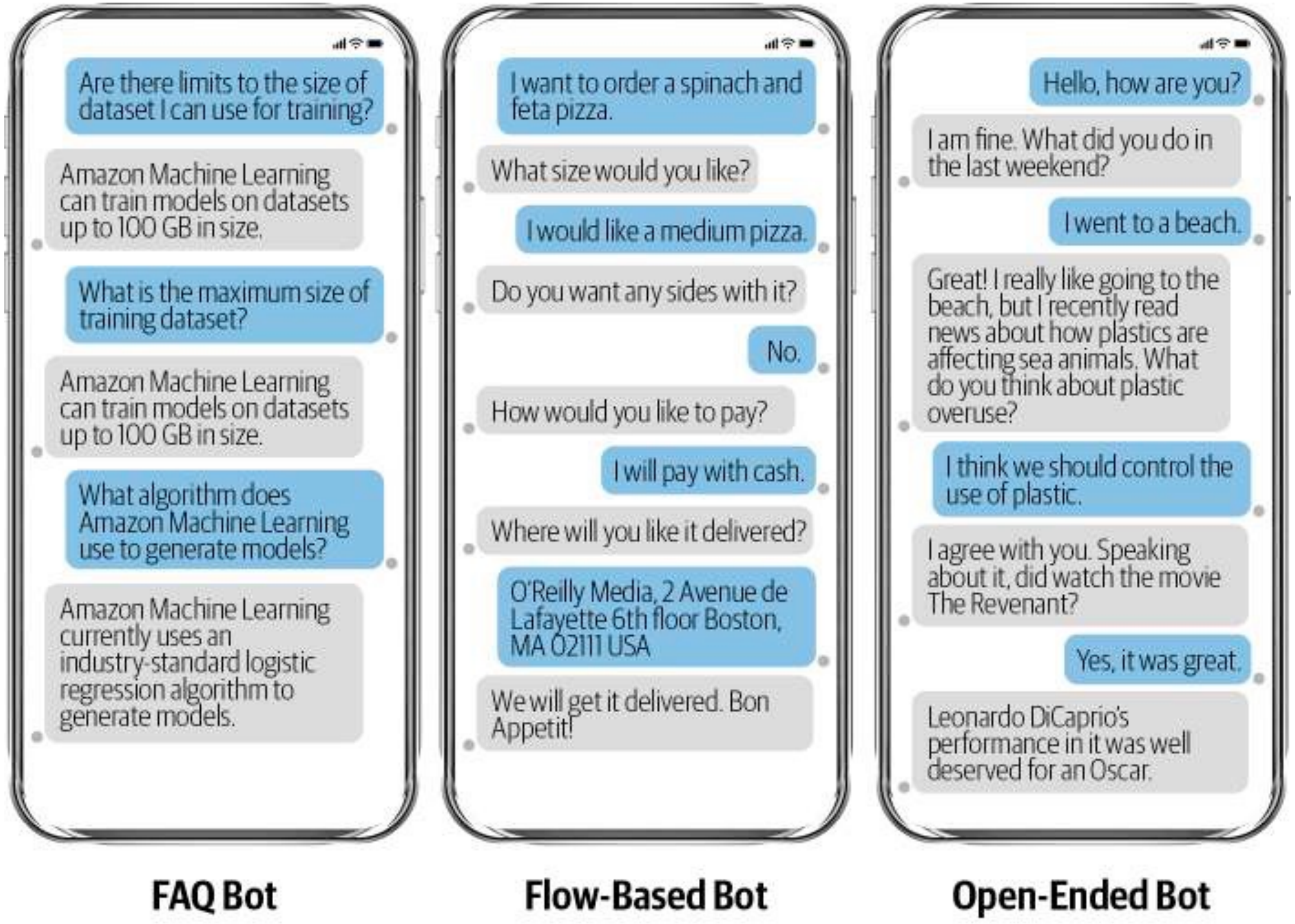
- Information Extraction
  - Key phrase Extraction (KPE)
  - Named Entity Recognition (NER)
  - Named Entity Linking (NEL)
  - Relationship Extraction
- Chatbots
- Topic Modeling –
  - Generally done with Latent Dirichlet Allocation (LDA) and not DL. LDA is a special case of Naïve Bayes with an assumption of Dirchlet Process prior

Attention Mechanism Video



https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Chatbots



RASA Chatbot Framework

Understand intent, take action based on rules

NLP task	Use	Nature of data
Search	Find relevant content for a given user query.	World wide web/large collection of documents
Topic modeling	Find topics and hidden patterns in a set of documents.	Large collection of documents
Text summarization	Create a shorter version of the text with the most important content	Typically a single document
Recommendations	Showing related articles	Large collection of documents
Machine translation	Translate from one language to another	A single document
Question answering system	Get answers to queries directly instead of a set of documents.	A single document or a large collection of documents

Top NLP Libraries

- NLTK - tokenization, lemmatization, stemming, parsing, POS tagging, etc. This library has tools for almost all NLP tasks. Supports large number of languages
- Spacy – The main competitor for NLTK. These two libraries can be used for the same tasks. Limited language support
- Gensim – Topic and vector space modelling, document similarity
- Polyglot – similar to NLTK and has support for a large number of languages. But slow and not enough support.

Automatic Speech Recognition Models

- https://openai.com/research/whisper
- Wave 2 Vec 2
- HuBERT
- Neural Speech Models – Textless Speech Models

Additional Material

Word Piece Tokenization – Byte Pair Encoding

- ("hug", 10), ("pug", 5), ("pun", 12), ("bun", 4), ("hugs", 5)
  - Split: ("h" "##u" "##g", 10), ("p" "##u" "##g", 5), ("p" "##u" "##n", 12), ("b" "##u" "##n", 4), ("h" "##u" "##g" "##s", 5)
- The most frequent pair is ("##u", "##g") (present 20 times), but the individual frequency of "##u" is very high, so its score is not the highest (it's 1 / 36).
- All pairs with a "##u" actually have that same score (1 / 36), so the best score goes to the pair ("##g", "##s") — the only one without a "##u" — at 1 / 20, and the first merge learned is ("##g", "##s") -> ("##gs").
- Continue merging until we reach the desired vocabulary size