EEP 596: Adv Intro ML || Lecture 18

Dr. Karthik Mohan

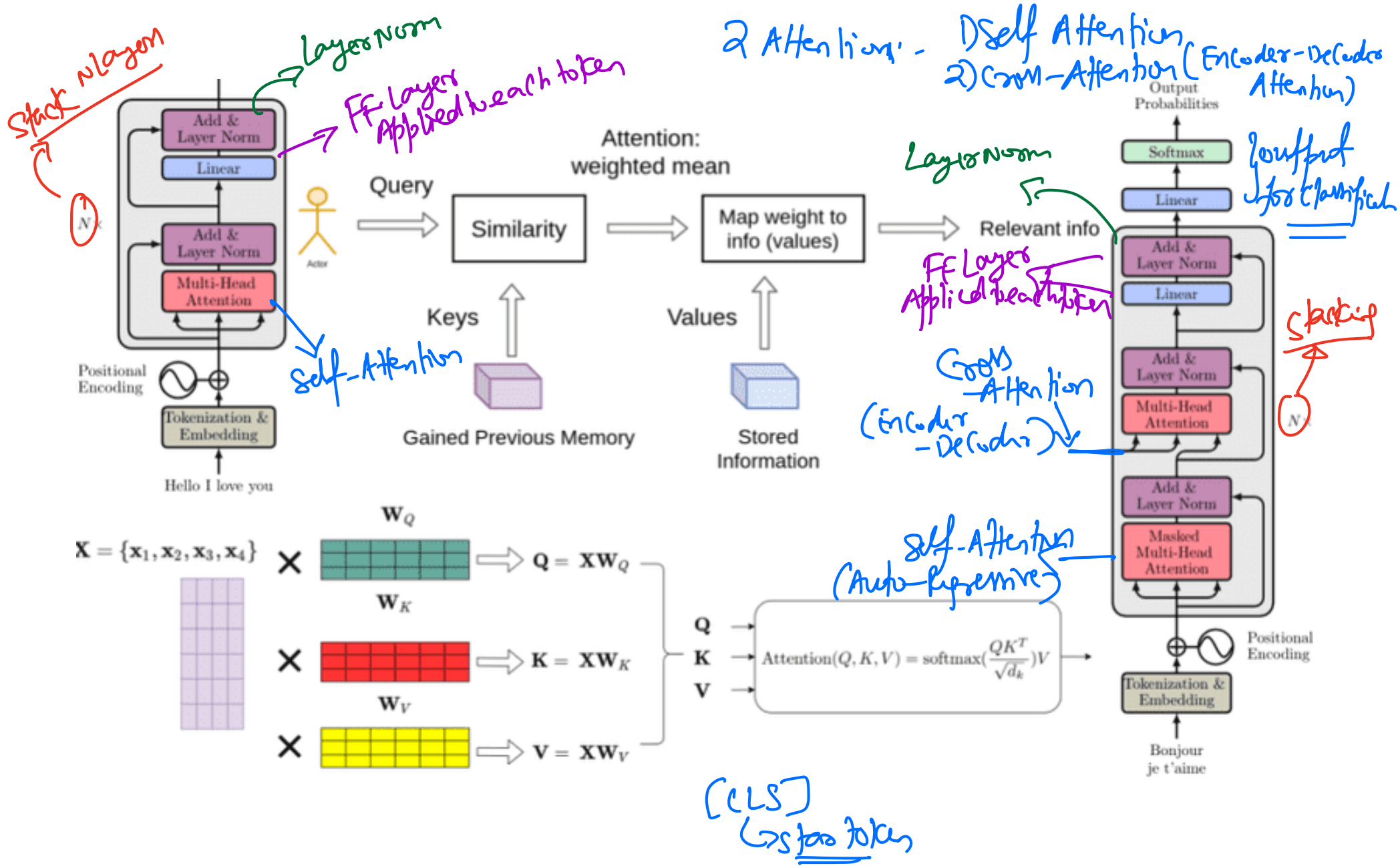
Univ. of Washington, Seattle

March 7, 2023

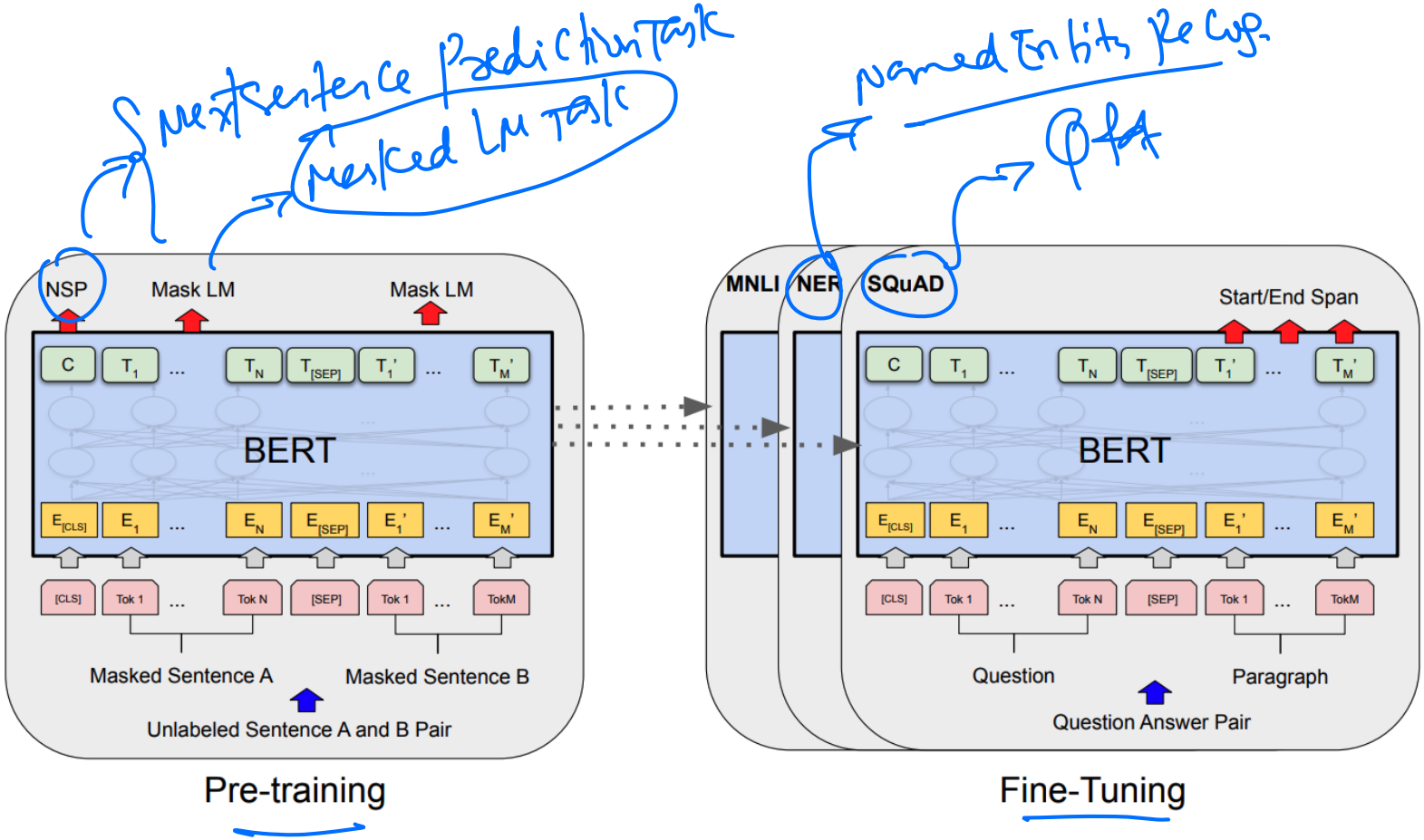
Today

- Transformers Recap
- Fine-tuning Transformers
- Demo on fine-tuning
- Course Recap

Transformers Architecture



BERT - Bi-directional Encoders from Transformers



BERT pre-training

Two Tasks

1. **Masked LM Model:** Mask a word in the middle of a sentence and have BERT predict the masked word
2. **Next-sentence prediction:** Predict the next sentence - Use both positive and negative labels. How are these generated?

BERT pre-training

Two Tasks

1. **Masked LM Model:** Mask a word in the middle of a sentence and have BERT predict the masked word
2. **Next-sentence prediction:** Predict the next sentence - Use both positive and negative labels. How are these generated?

ICE: Supervised or Un-supervised?

1. Are the above two tasks supervised or un-supervised?

BERT pre-training

Two Tasks

1. **Masked LM Model:** Mask a word in the middle of a sentence and have BERT predict the masked word
2. **Next-sentence prediction:** Predict the next sentence - Use both positive and negative labels. How are these generated?

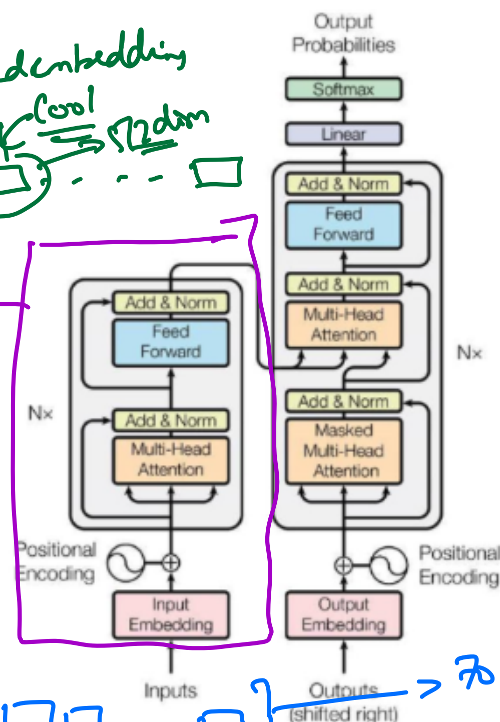
ICE: Supervised or Un-supervised?

1. Are the above two tasks supervised or un-supervised?

Data set!

English Wikipedia and book corpus documents!

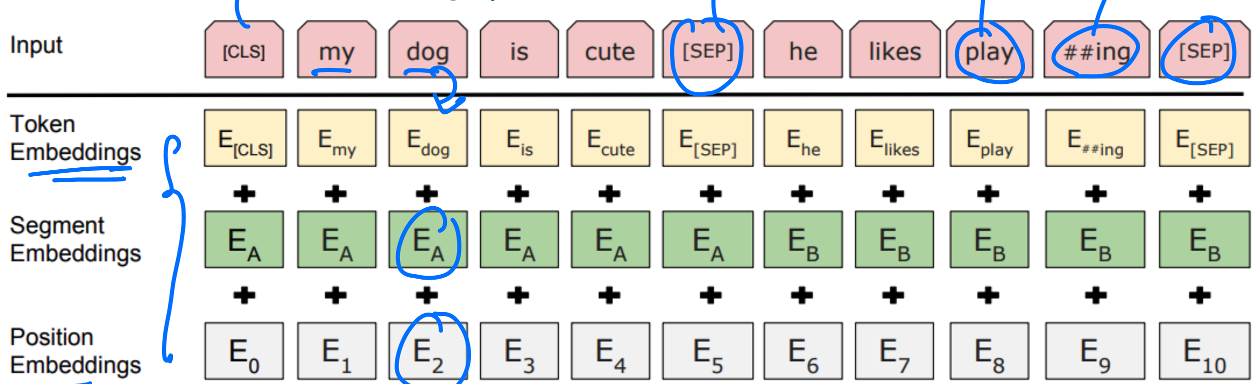
BERT Embeddings



Embeddings

1) Token Embedding:-
This from last hidden layer for "each token". E.g: "Cool"

2) Sentence Embedding:-
a) CLS Embedding as sentence embedding
b) Mean pooling on token embeddings



Fine-Tuning Transformers for down-stream tasks

A methodology for fine-tuning transformers for classification tasks

1. **Pick Base pre-trained Architecture:** Pick a base pre-trained architecture as a starting point for your fine-tuning. Example: `bert-base-uncased` is one such pre-trained model that can be loaded through `Hugging Face Transformers Library`

Fine-Tuning Transformers for down-stream tasks

A methodology for fine-tuning transformers for classification tasks

1. **Pick Base pre-trained Architecture:** Pick a base pre-trained architecture as a starting point for your fine-tuning. Example: `bert-base-uncased` is one such pre-trained model that can be loaded through `Hugging Face Transformers Library`
2. **Extract output from pre-training:** How do you want to use the output from pre-training going into *fine-tuning*? a) Extract embedding from the first token, `CLS` b) Average embeddings of all tokens as a starting point (mean pooling).

Fine-Tuning Transformers for down-stream tasks

A methodology for fine-tuning transformers for classification tasks

1. **Pick Base pre-trained Architecture:** Pick a base pre-trained architecture as a starting point for your fine-tuning. Example: `bert-base-uncased` is one such pre-trained model that can be loaded through `Hugging Face Transformers Library`
2. **Extract output from pre-training:** How do you want to use the output from pre-training going into *fine-tuning*? a) Extract embedding from the first token, `CLS` b) Average embeddings of all tokens as a starting point (mean pooling).
3. **Add fine-tuning layers:** Add fine-tuning layers on top of the pre-trained layers. Example, starting with the pooled embeddings, construct one or more dense layers (`Feed-Forward NN style`) to extract finer representations of the input. Add the output layer and its activation (typically softmax for classification tasks).

Fine-Tuning Transformers for down-stream tasks

A methodology for fine-tuning transformers for classification tasks

1. **Pick Base pre-trained Architecture:** Pick a base pre-trained architecture as a starting point for your fine-tuning. Example: `bert-base-uncased` is one such pre-trained model that can be loaded through `Hugging Face Transformers Library`
2. **Extract output from pre-training:** How do you want to use the output from pre-training going into *fine-tuning*? a) Extract embedding from the first token, `CLS` b) Average embeddings of all tokens as a starting point (mean pooling).
3. **Add fine-tuning layers:** Add fine-tuning layers on top of the pre-trained layers. Example, starting with the pooled embeddings, construct one or more dense layers (`Feed-Forward NN style`) to extract finer representations of the input. Add the output layer and its activation (typically softmax for classification tasks).
4. **Set training schedule, hyper-parameters, etc:** Set up optimizer (e.g. ADAM), hyper-parameters, training schedule, etc for training.

ICE #1

BERT Embeddings and Emotion Detection

Let's say you want to do emotion detection by fine-tuning BERT (Encoding Transformer) on a data set. One of the outputs of the *BERT pre-trained model* for a given input is the *last-hidden-state*. This includes an embedding for every token that was passed into BERT.
Let's say you are going to start with the last hidden layer and use that as input for your *fine-tuned* model. This ICE is about the dimensionality of the inputs and outputs. Let's say you have sentences of the kind: "I am looking forward for today! It's going to be a big day" This sentence conveys excitement. There are 13 words in this input and using *word-piece tokenization*, you arrive at 20 sub-tokens as input into the BERT model. The last hidden layer includes an embedding for every single token. Let's say the embedding dimension for a token is 768.

ICE #1 continued

BERT Embeddings and Emotion Detection

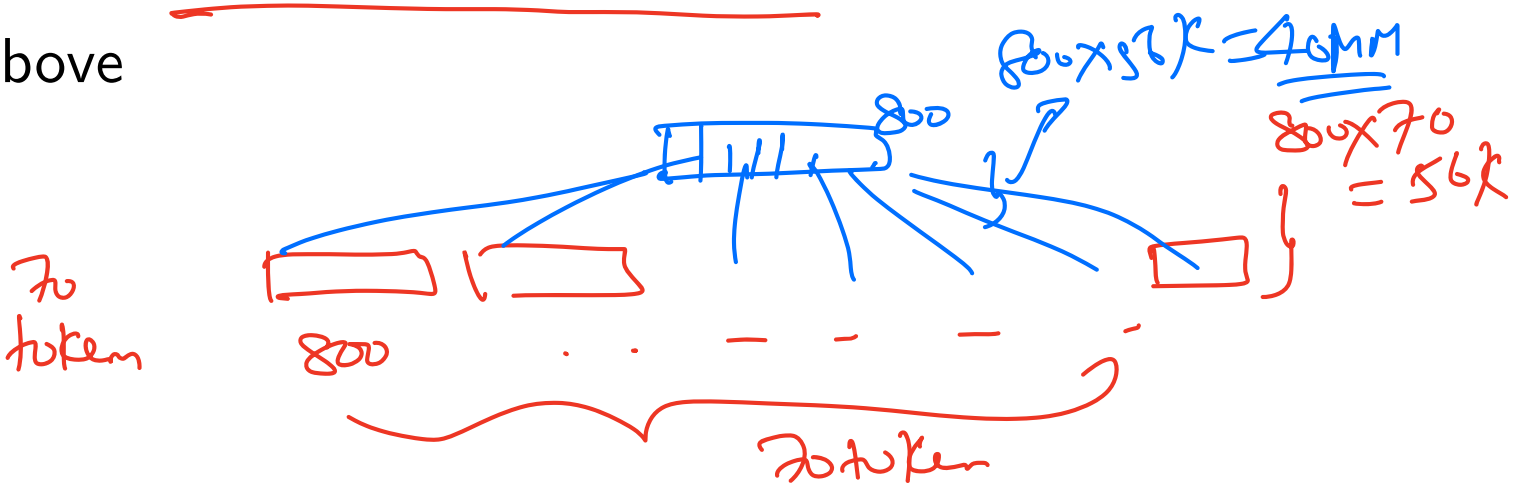
There are 13 words in this input and using *word-piece tokenization*, you arrive at 20 sub-tokens as input into the BERT model. The last hidden layer includes an embedding for every single token. Let's say the embedding dimension for a token is 768. For the purpose of emotion detection - You can either use the *CLS* token (Start token) embedding (also called the pooled embedding) or you can take the average of the embeddings of the tokens in the last hidden layer of BERT. What's the dimension of the pooled BERT embedding of this particular input example and what's the total dimension of the last hidden layer?

1. 768 and 9984
2. 15360 and 768
3. 9984 and 768
4. 768 and 15360

ICE #2

Why does pooling of the output need to be done for sequence classification (e.g. emotion detection)?

1. Reduces the dimensionality
2. Averages context from all the tokens
3. Computational concerns for training the fine-tuned model
4. All of the above



Fine-Tuning BERT for Sentence Paraphrasing Demo

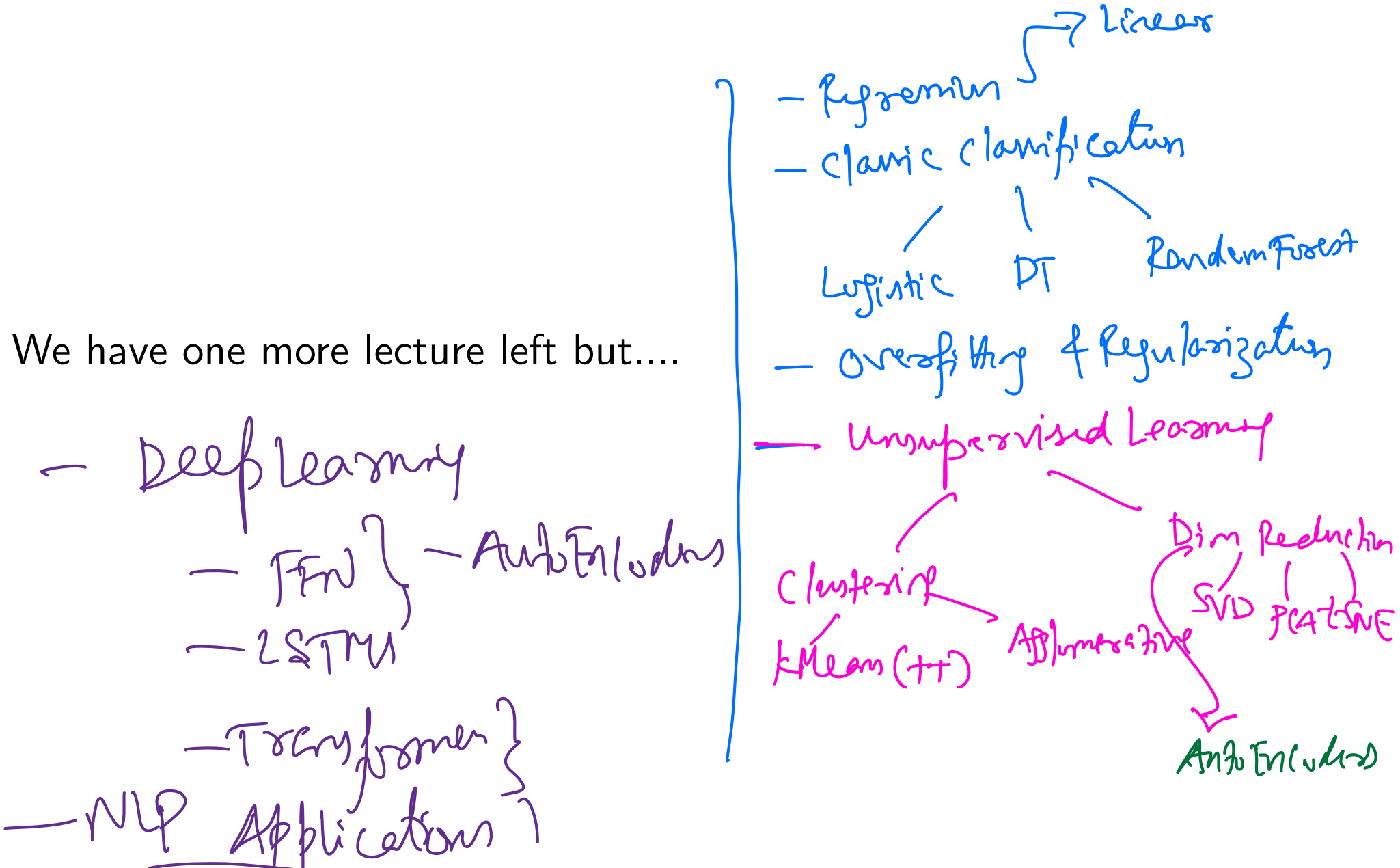
Demo Notebook available on course page

Breakout

Discuss your modeling approach for Kaggle 1 and Kaggle 2. What models/archs would you use? How can you build off Kaggle 1 results to build a model for Kaggle 2?

Course Wrap-up

We have one more lecture left but....



Final Course Survey

Course Feedback

Let's take next 15 minutes for sharing course feedback. What worked for you, what did you like, and what could have been better?