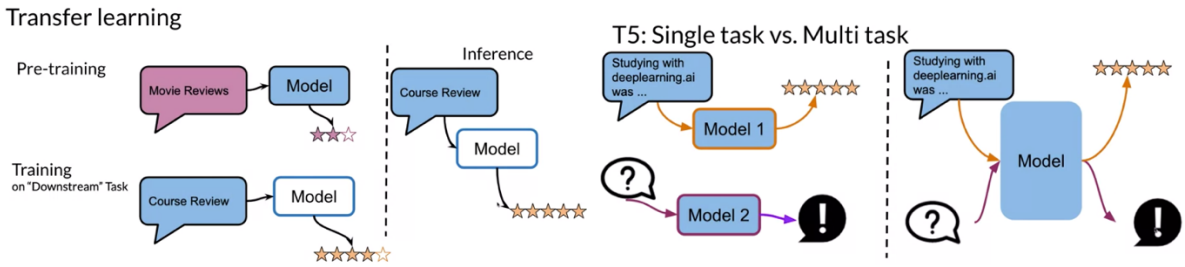
Question Answering

1. Question Answering – Given a question what is the answer given a context (BERT)
2. Transfer learning – Train for a specific task and apply it to a diff task (T5)
3. BERT – Looks at context from both directions of the word.



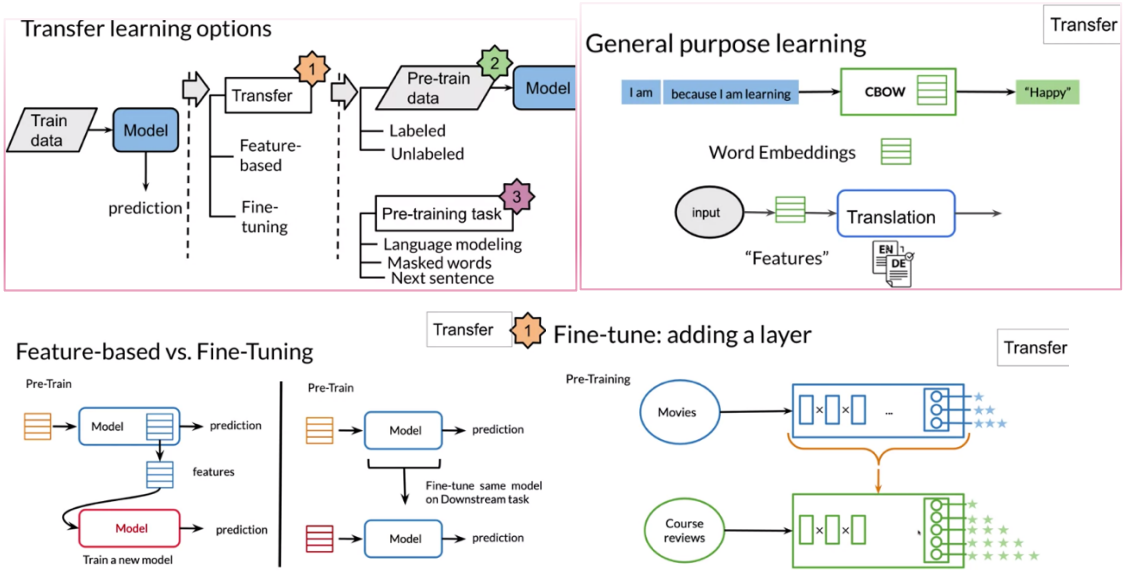
Transfer learning benefits

* Reduce training time

* Improve predictions

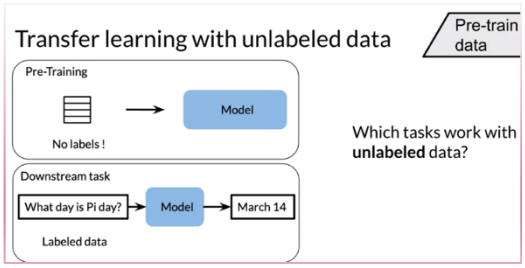
* Small dataset

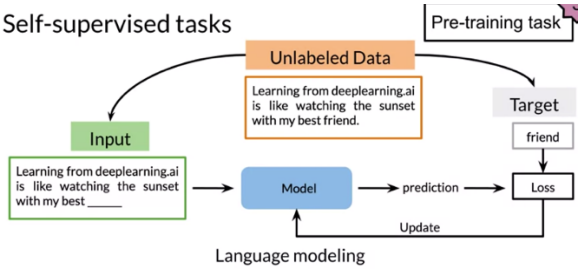
Transfer Learning in NLP

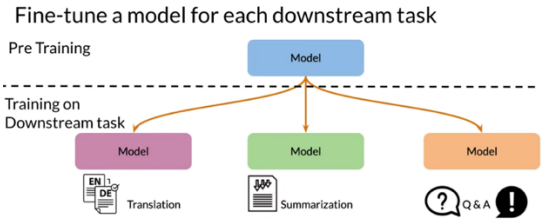


Larger the data better the performance

Self-supervised data, given unlabeled data you can create inputs or features. Then you create targets.



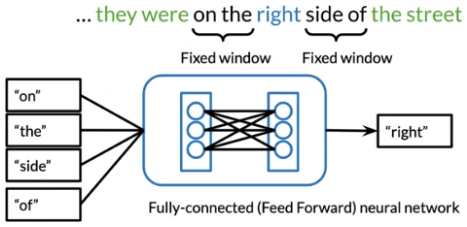




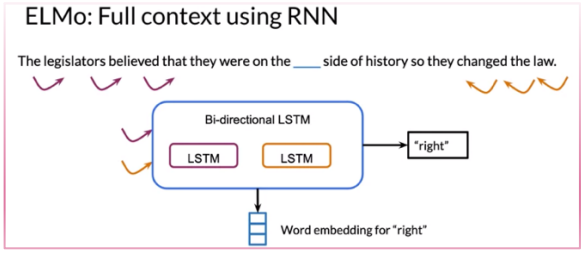
ELMo, GPT, BERT, T5

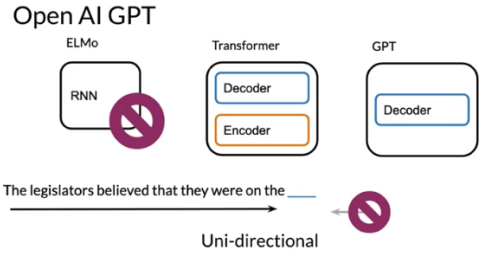
CBOW -> ELMo -> GPT -> BERT -> T5

Continuous Bag of Words



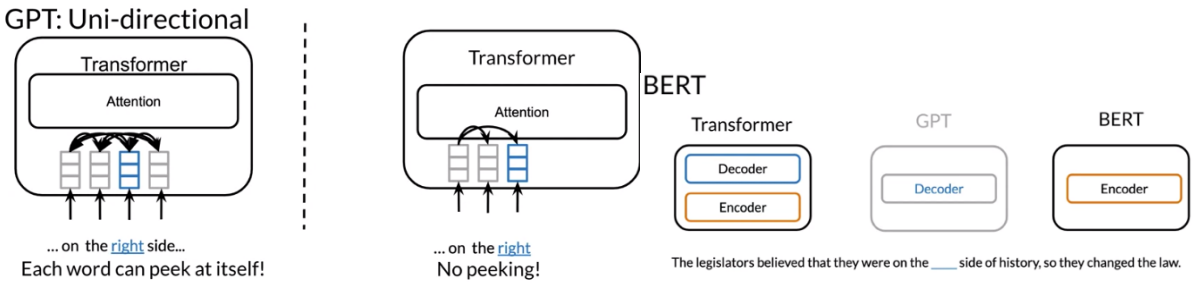
ELMo = Use all context words not just bigram on either side

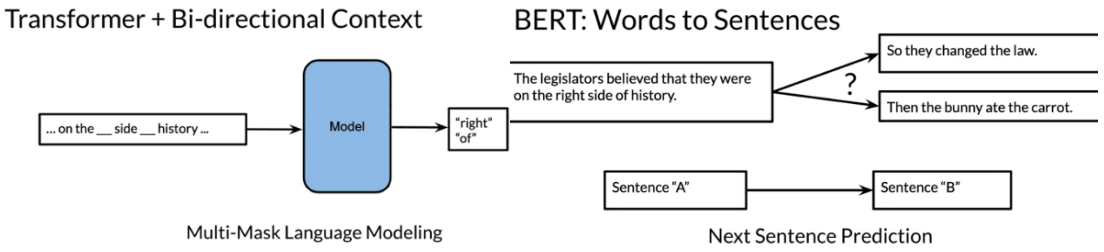


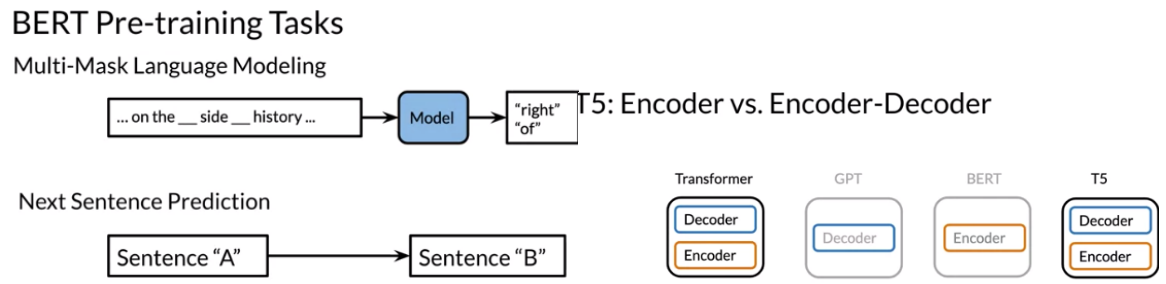


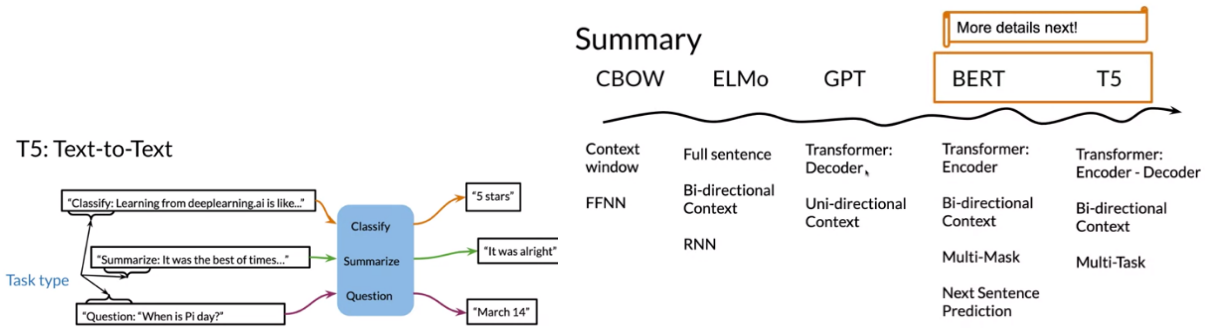
Open AI GPT

GPT is unidirectional. Uses just the decoder stack, does not have an encoder.









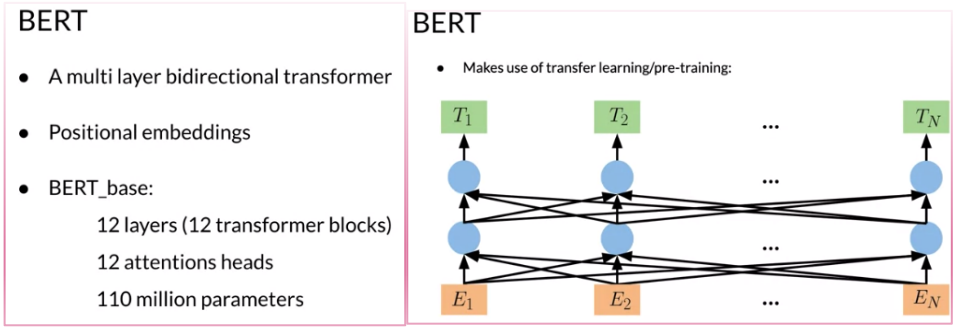
Bidirectional Encoder Representations from Transformers (BERT)

*. Input embedding

* Go through some transformer black.

* During pre-training model is trained on unlabeled data

* For fine-tuning, BERT is initialized with pre-trained params and all params are fine tuned using labeled data from downstream tasks.

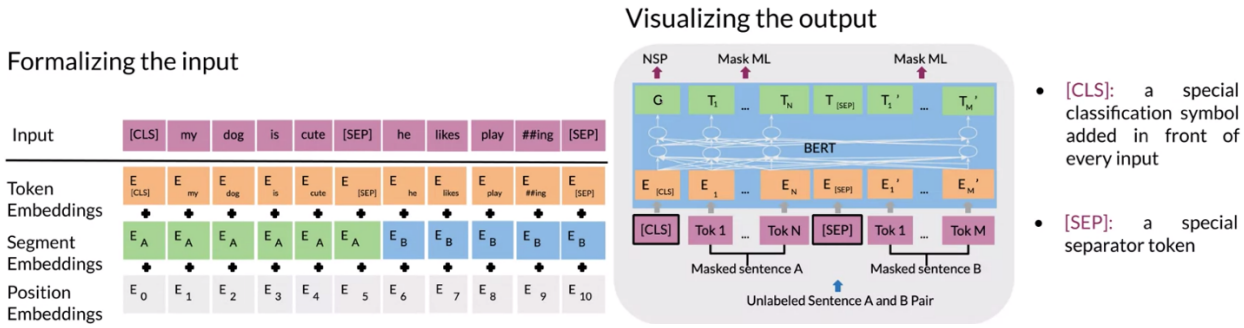


Before feeding the word sequences to the BERT model, we mask 15 percent of the words, and then, the training data generator chooses 15 percent of these positions at random for prediction. Then, if the ith token is chosen, we replace the ith token with one, the mask token 80 percent of the time, and then, two, a random token 10 percent of the time, and then, three, the unchanging ith token 10 percent of the time. In this case, then TI, which you've seen in the previous slide, will be used to predict the original token with cross-entropy loss. In this case, this is known as the masked language model. add the dense layer after the TI token and use it to classify after the encoder outputs. You just multiply the output's vectors by the embedding matrix, and then, to transform them into vocabulary dimension and you add a softmax at the end.

Summary

- Choose 15% of the tokens at random: mask them 80% of the time, replace them with a random token 10% of the time, or keep as is 10% of the time.
- There could be multiple masked spans in a sentence
- Next sentence prediction is also used when pre-training.

BERT Objective



- [CLS]: a special classification symbol added in front of every input
- [SEP]: a special separator token

BERT Objective

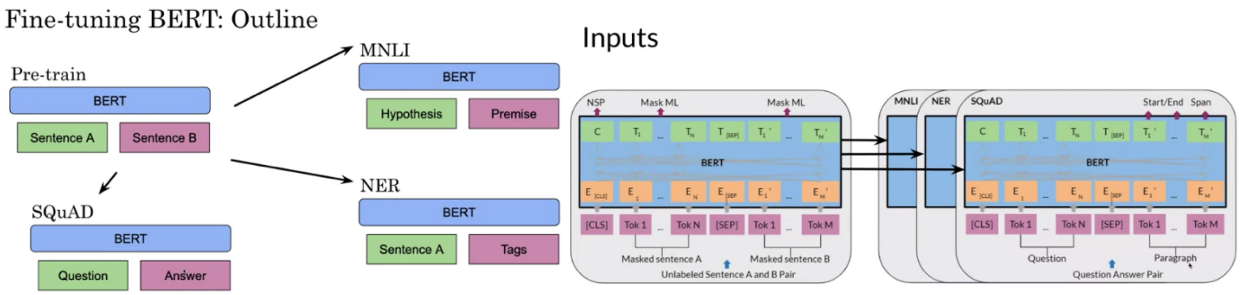
Objective 1:
Multi-Mask LM

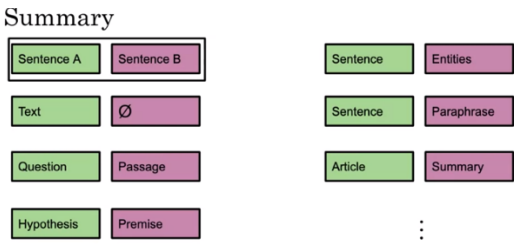
Loss: Cross Entropy Loss

Objective 2:
Next Sentence Prediction

Loss: Binary Loss

Fine-tuning BERT





Transformer: T5

Transfer learning + Mask language modeling

Uses transformers when training

Applications

* Classification

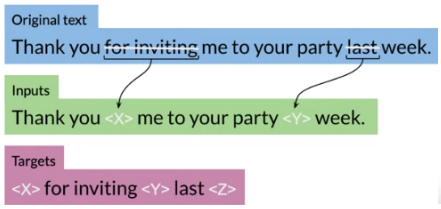
* Machine Translation

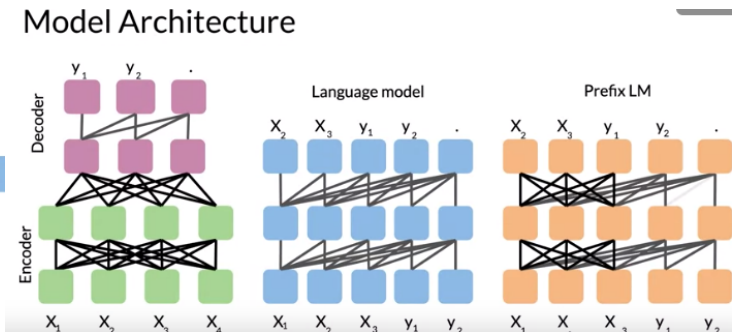
* Question and Answering

* Summarization

* Sentiment

Transformer - T5 Model





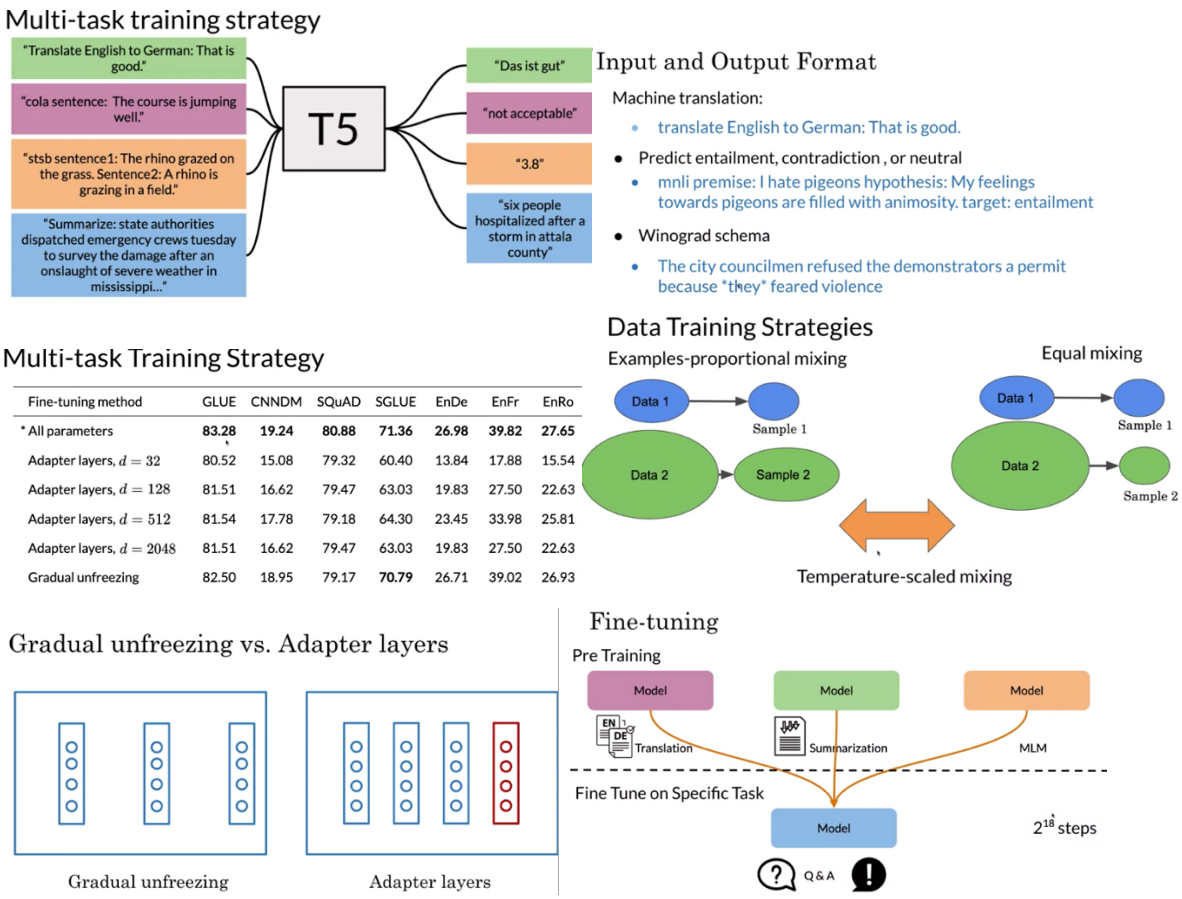
Model Architecture

* Encoder/Decoder

* 12 transformer blocks each

* 220Million params

Multi-Task Training Strategy



GLUE Benchmark – General Language Understanding Evaluation

GLUE is a collection of resources training, evaluating, and analyzing, natural language understanding systems.

* Drive research

* Model agnostic

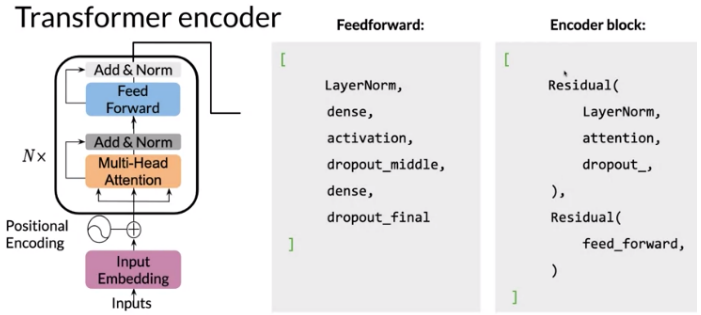
* Makes use of transfer learning

- A collection used to train, evaluate, analyze natural language understanding systems
- Datasets with different genres, and of different sizes and difficulties
- Leaderboard

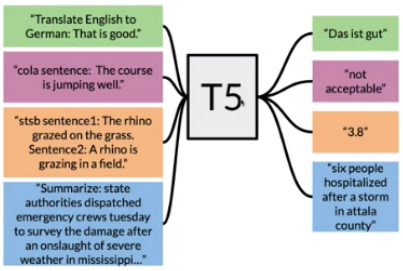
Tasks Evaluated on

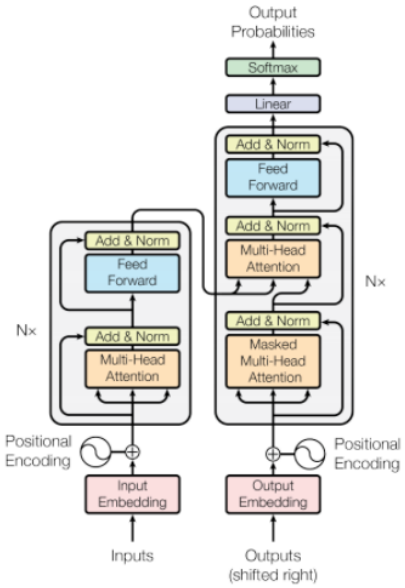
- Sentence grammatical or not?
- Sentiment
- Paraphrase
- Similarity
- Questions duplicates
- Answerable
- Contradiction
- Entailment
- Winograd (co-ref)

Question Answering



- Load a pre-trained model
- Process data to get the required inputs and outputs: "question: Q context: C" as input and "A" as target
- Fine tune your model on the new task and input
- Predict using your own model





```
from unicodedata import normalize
def get_hex_encoding(s):
    return ' '.join(hex(ord(c)) for c in s)
# Replace tabs and  newlines and white space with '\u2581'

BPE (Byte pair encoding) algo
import ast
def convert_json_examples_to_text(filepath):
```

```
    example_jsons = list(map(ast.literal_eval, open(filepath))) # Read in the json
from the example file
    texts = [example_json['text'].decode('utf-8') for example_json in example_jsons] #
Decode the byte sequences
    text = '\n\n'.join(texts)       # Separate different articles by two newlines
    text = normalize('NFKC', text)  # Normalize the text
    with open('example.txt', 'w') as fw:
        fw.write(text)
    return text
```