AIML 428

- The peer review
  - Submit the excel file regularly
- Teaching evaluation is open and due in 7 days.
- Our class rep is Ye LI, Email: liye2@myvuw.ac.nz

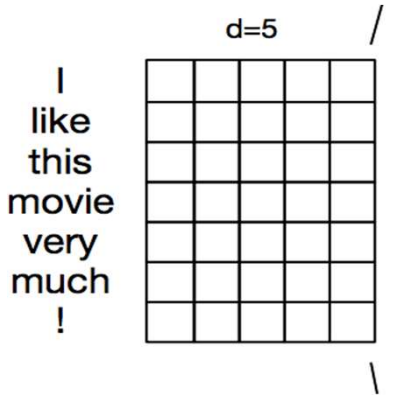
Today

- Summary of text classification with embeddings
- Limitations
- BERT

Text classification with word embedding

- The input to most NLP tasks are sentences or documents
- They can be represented as a matrix use word embeddings.
- Fit perfectly well with CNN, RNN.

  - RNN: Recurrent Neural Networks
    - LSTM: Long Short Term Memory
    - Bi-LSTM: Bidirectectional LSTM



- Each row of the matrix corresponds to one token.
  - Typically a word,
  - But can be a character, char n-gram
- Character level word embedding with CNN has shown good performance

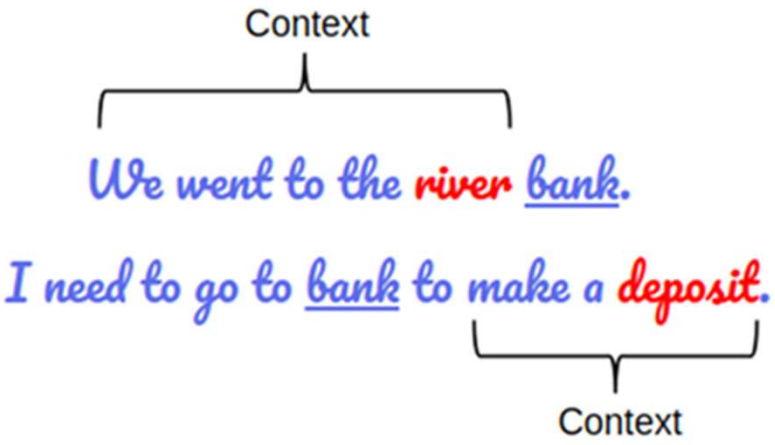
Discussion

- What are the advantages of this text representation?

- What are the limitations?

Limitations

- Disadvantages:
  - long training time
  - Interpretability
  - …
- Ambiguity:
  - Polysemy: same words having different meanings based on their context.



Sentence or document representations?

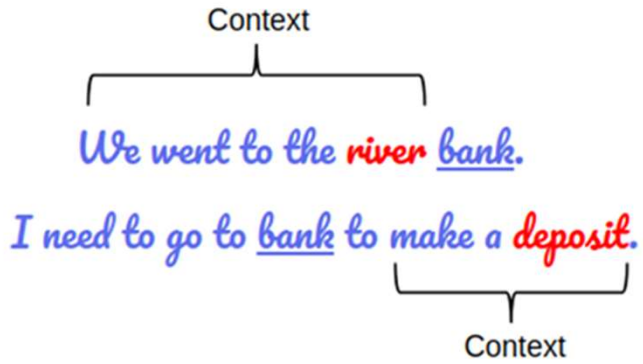
- Classic classifiers require document level representations:
  - Use average of word embeddings, tf-idf weighted average
  - Problems: “non-compositional” interpretations
    - An old cat/ an old city
    - A red flag/ a fake gun/ a potential suicide
    - A hot dog
- Doc2Vec
  - Train an embedding for each document.
  - Limited research shows not very good performance
- We need better ways for sentence representation

Research leading to BERT

- Contextualized Embedding uses bidirectional LSTM or transformer models to learn context-dependent word embeddings.
- It generates the representation of a word while modelling its syntax, semantics and polysemy. Ex: ELMo, ULM-FiT, GPT/GPT-2, BERT

ELMo. ULMFit

- ELMo to address Polysemy
  - training word embeddings using layers of complex **Bi-directional LSTM** architectures.
  - the same word can have multiple ELMo embeddings based on the context it is in.



- ULMFiT
  - **transfer learning** in NLP.
  - pre-train language models: estimating the probability of the next word given the previous words.
  - fine-tuned with fewer data (e.g. less than 100 examples for document classification tasks).

- *Peters, Matthew; Neumann, Mark; Iyyer, Mohit; Gardner, Matt; Clark, Christopher; Lee, Kenton; Luke, Zettlemoyer (15 February 2018). "Deep contextualized word representations". arXiv:1802.05365v2*
- *Howard, Jeremy; Ruder, Sebastian (18 January 2018). "Universal Language Model Fine-tuning for Text Classification". arXiv:1801.06146v5*

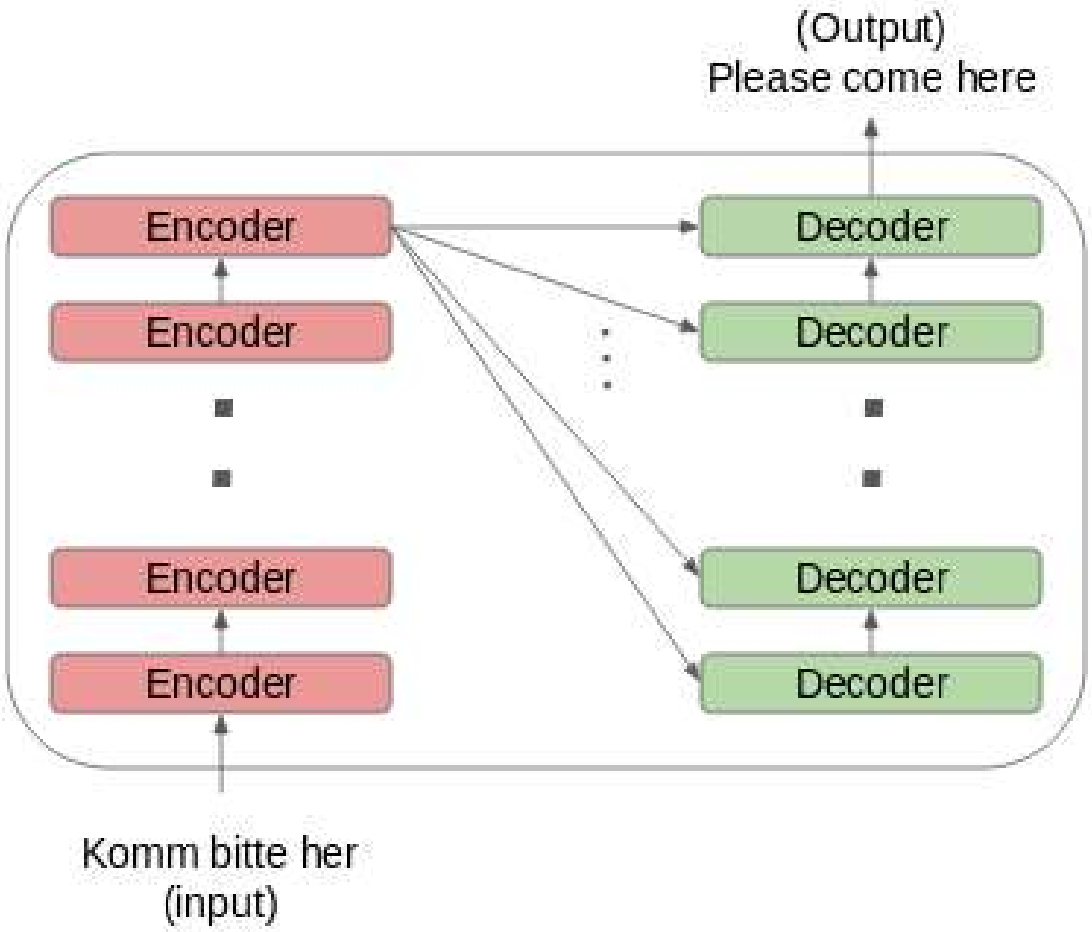
What is BERT?

- BERT stands for Bidirectional Encoder Representations from Transformers. It is designed to pre-train deep bidirectional representations from unlabelled text by jointly conditioning on both left and right context. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of NLP tasks.

Devlin et al., “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” arXiv preprint arXiv:1810.04805, 2018.

Transformer

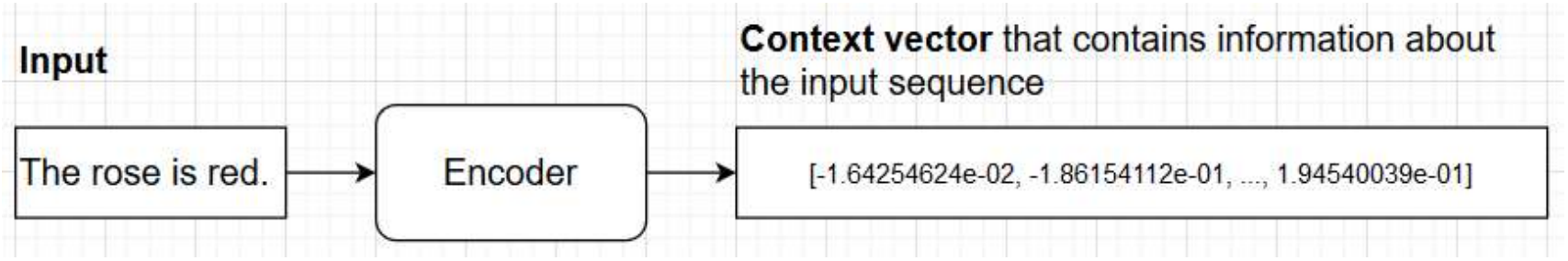
- https://www.analyticsvidhya.com/blog/2019/06/understanding-transformers-nlp-state-of-the-art-models/



What is BERT?

- BERT stands for Bidirectional Encoder Representations from Transformers.

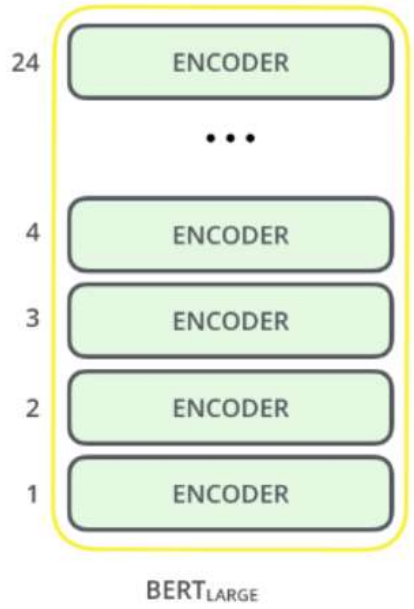
BERT is based on the Encoder of the Transformer architecture and is developed by Google.

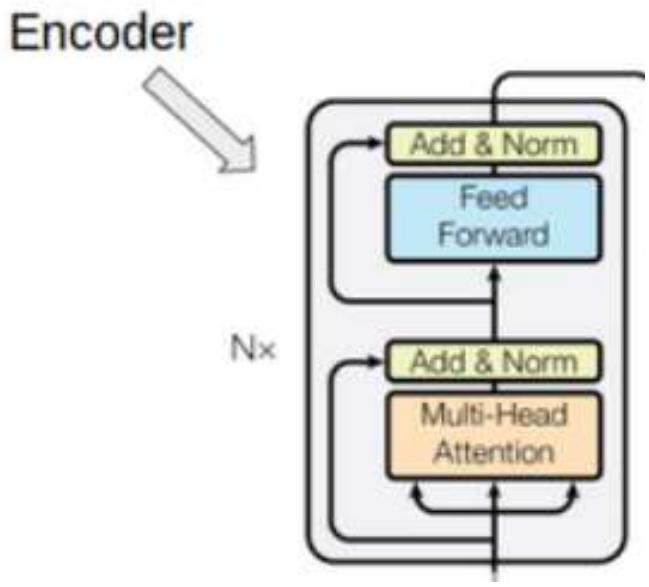


Devlin et al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," arXiv preprint arXiv:1810.04805, 2018.

What is BERT?

- BERT stands for Bidirectional Encoder Representations from Transformers.





Vaswani et al.,
"Attention Is All You Need,"
https://arxiv.org/pdf/1706.0372.pdf

What is BERT?

- BERT stands for Bidirectional Encoder Representations from Transformers. It is designed to pre-train deep bidirectional representations from unlabelled text by jointly conditioning on both left and right context. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of NLP tasks.

Devlin et al., “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” arXiv preprint arXiv:1810.04805, 2018.

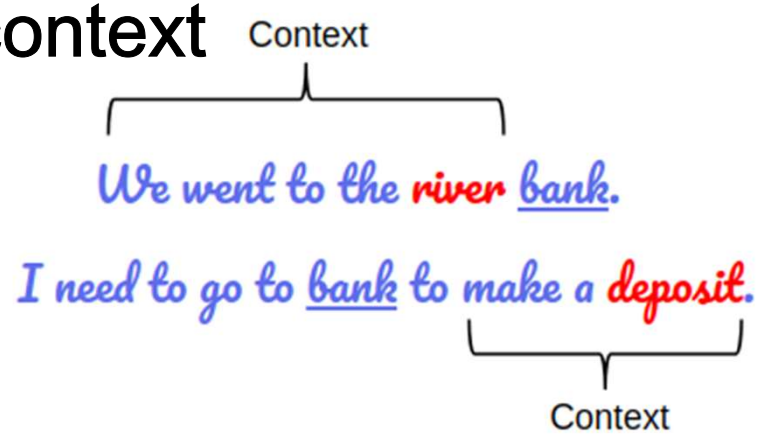
What is BERT? (not used)

- BERT stands for Bidirectional Encoder Representations from Transformers. It is designed to pre-train deep bidirectional representations from unlabelled text by jointly conditioning on both left and right context. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of NLP tasks.

Devlin et al., “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” arXiv preprint arXiv:1810.04805, 2018.

What is BERT?

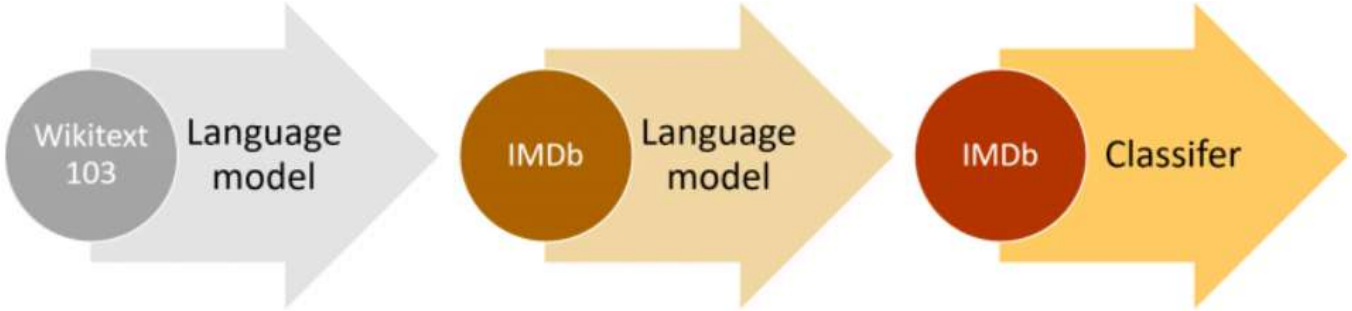
Left and right context



- If we try to predict the nature of the word “bank” by only taking either the left or the right context, then we will be making an error in at least one of the two given examples.
- What BERT does is to consider both the left and the right context before making a prediction.

What is BERT?

- Transfer Learning in NLP = Pre-training + Fine-tuning



and right context. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of NLP tasks.

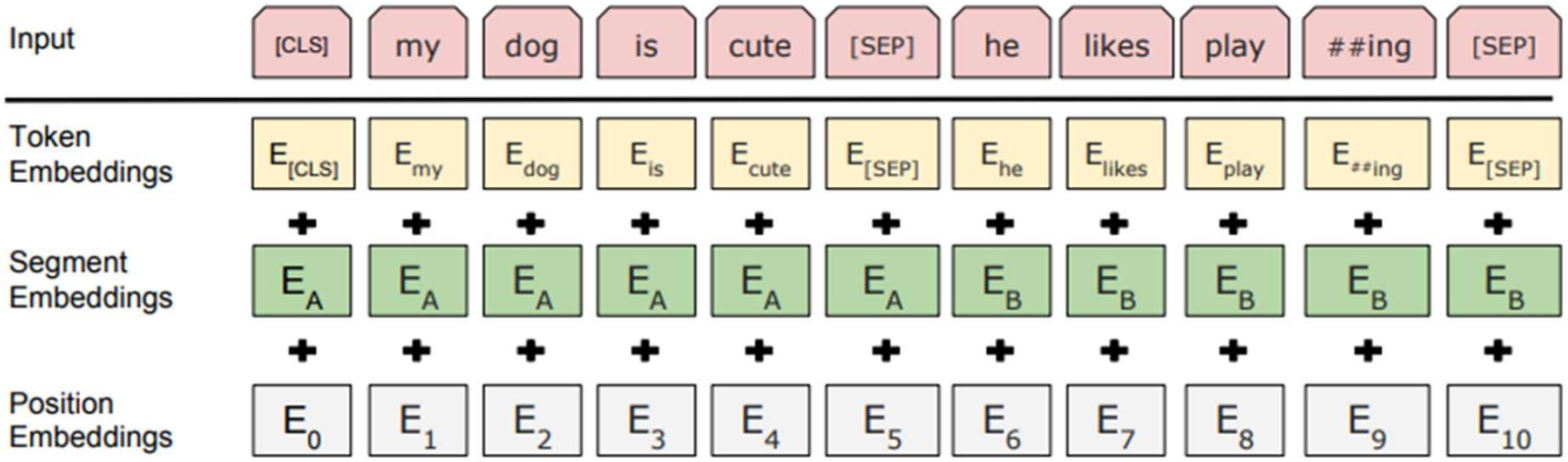
Devlin et al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," arXiv preprint arXiv:1810.04805, 2018.

How was BERT pre-trained?

- Text Preprocessing
- Pre-training Tasks
  - Masked Language Modelling
  - Next Sentence Prediction

How was BERT pre-trained?

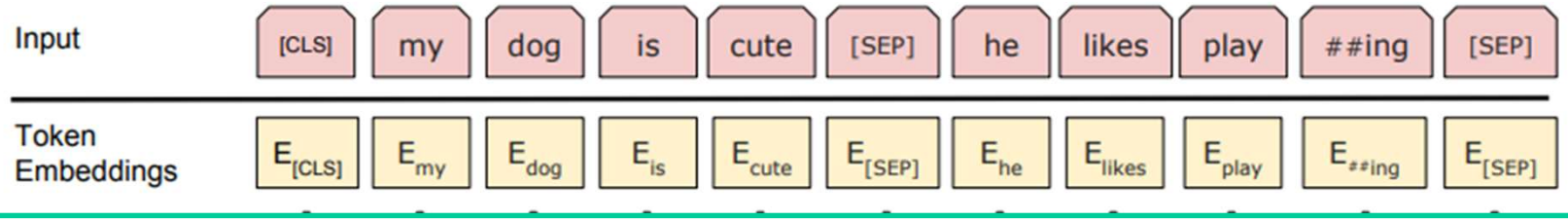
Text Preprocessing



- [CLS]: a special token added in front of every input example
- [SEP]: a special token separating sentences

How was BERT pre-trained?

Text Preprocessing

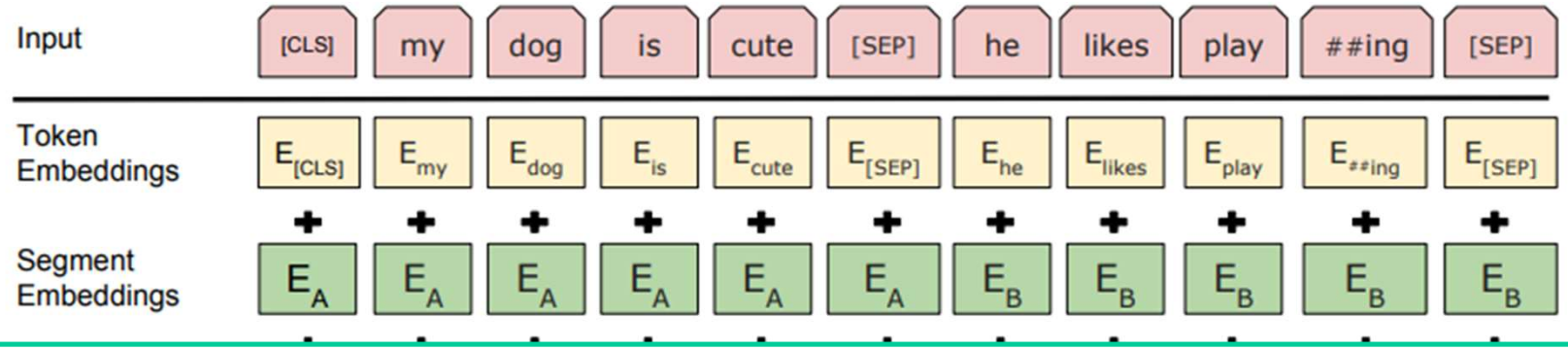


- **Token Embeddings:** These are the embeddings learned for the specific token from the WordPiece token vocabulary.

- [CLS]: a special token added in front of every input example
- [SEP]: a special token separating sentences

How was BERT pre-trained?

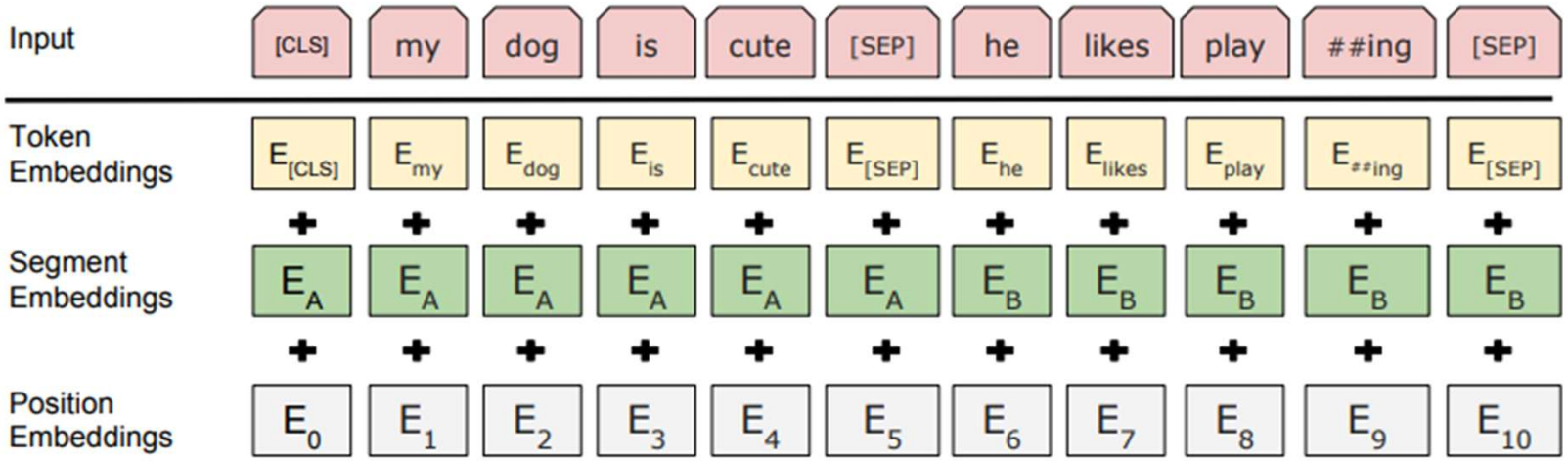
Text Preprocessing



- **Segment Embeddings:** in the above example, all the tokens marked as E_A belong to sentence A (and similarly for EB).
- [SEP]: a special token separating sentences

How was BERT pre-trained?

Text Preprocessing



- **Position Embeddings:** BERT uses positional embeddings to express the position of words in a sentence.

How was BERT pre-trained?

Pre-training Tasks

- Masked Language Modelling
  - I am studying at Victoria [MASK] of Wellington.
  - We'll then train the model in such a way that it should be able to predict "University" as the missing token.
  - A multi-class classification task
  - Understand the relationship between words.

How was BERT pre-trained?

Pre-training Tasks

- Next Sentence Prediction
  - Given two sentences – A and B, is B the actual next sentence that comes after A in the corpus, or just a random sentence?
  - A binary classification task
  - Understand the relationship between sentences.

References

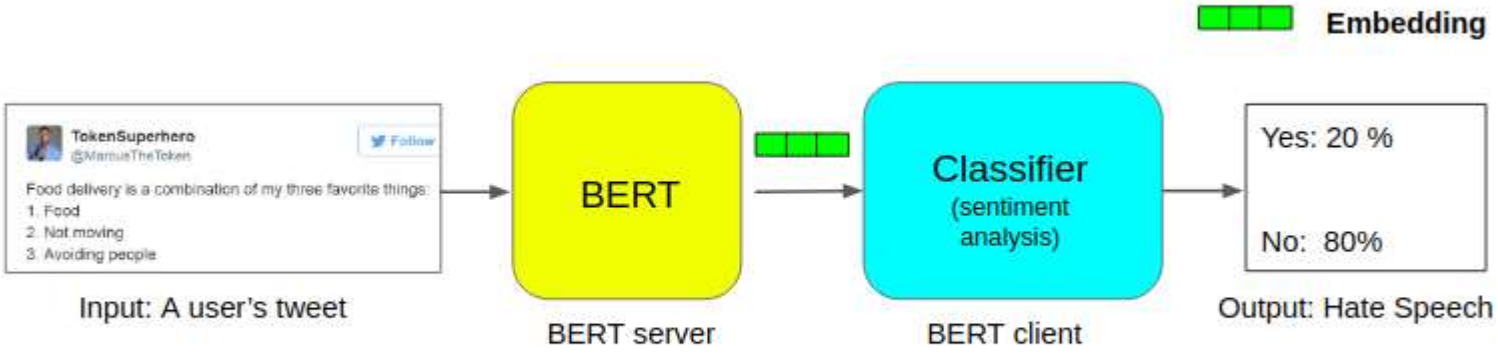
- Devlin et al., “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” arXiv preprint arXiv:1810.04805, 2018.
- Vaswani et al., “Attention Is All You Need,” arXiv preprint arXiv:1706.03762, 2017.
- “Demystifying BERT: A Comprehensive Guide to the Groundbreaking NLP Framework,” https://www.analyticsvidhya.com/blog/2019/09/demystifying-bert-groundbreaking-nlp-framework/, [Accessed: 22-Mar-2020].

BERT and text classification:

- The pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of NLP tasks.

Typically,

- Each sentence is changed to a vector
- List of sentences is changed to a matrix, can be directly fit into any classifier.
- Many extensions: Sentence-BERT is much faster



BERT for text classification

- Use CLS token as sentence representation
- Use the output of the final encoder layer
- Use the output of any encoder layer
- Use the pretrained token embeddings from BERT

- You may try simple classifiers
  - LR
  - MLP

- Or CNN

Python implementation

- Code from Tobias Tuan Ha
- Sentence embedding using RoBERTa
- https://colab.research.google.com/drive/13ZKL3b18j3lvLXtGqYY4VuGIwRnUFKmL

- BERT for text classification tutorial (with IMDB)
  - https://www.tensorflow.org/tutorials/text/classify_text_with_bert
  - BERT encoder as a kerasLayer

- More recent tutorial
  - https://curiousily.com/posts/sentiment-analysis-with-bert-and-hugging-face-using-pytorch-and-python/