08 – CONTEXTUAL WORD REPRESENTATION

MACHINE LEARNING FOR NATURAL LANGUAGE PROCESSING, AIMS 2024

Dr. Elvis Ndah

WORD REPRESENTATION AND CONTEXT

Same word different context

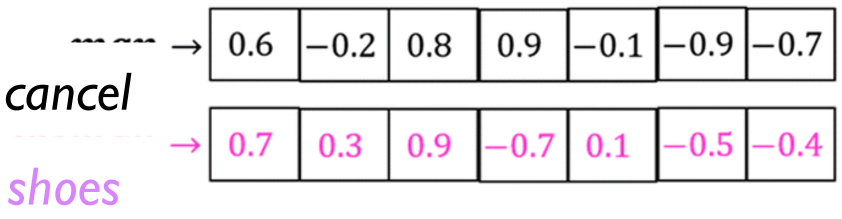
1. The verse broke. the verb *break* means *shattered to pieces*
2. Dawn broke. the verb *break* means *begin*
3. The news broke. the verb *break* means *to be known or published*
4. Sandy broke the world record. the verb *break* means *to be surpassed the previous level*
5. Sandy broke the law. the verb *break* means *act of transcreation.*
6. The burglar broke into the house. the verb *break* means *physical act of transcreation*

Same words different meanings and context

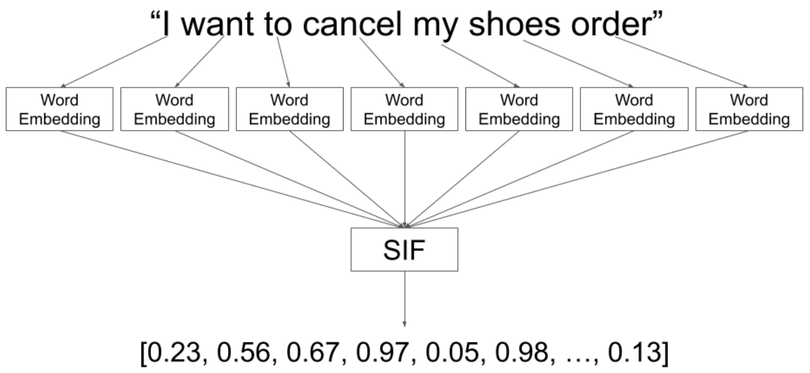
1. Flat tire/beer/note/surface
2. Throw a party/fight/ball/fit

WORD EMBEDDING (CO-OCCURRENCE)

- Word embeddings are the basis of deep learning for NLP



- Word embeddings (*word2vec*, *GloVe*) are often *pre-trained* on text corpus from co-occurrence statistics.



WORD EMBEDDING - WORD2VEC

- One vector for each word type

$$v(\text{bank}) = \begin{pmatrix} -0.224 \\ 0.130 \\ -0.290 \\ 0.276 \end{pmatrix}$$

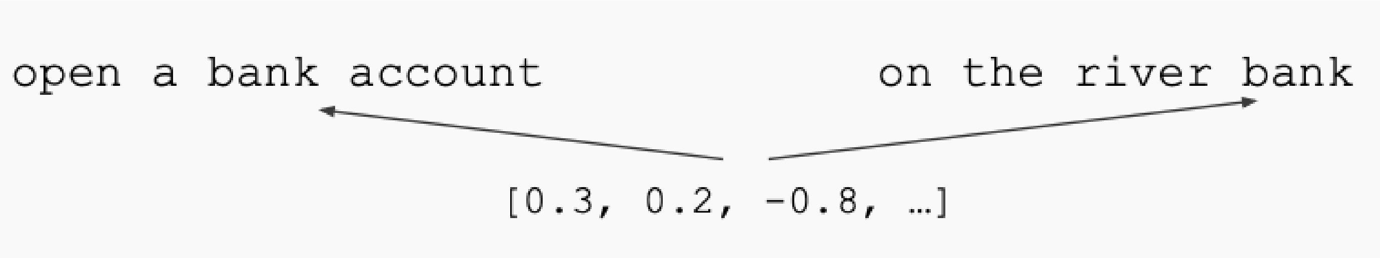
- Polysemous words, e.g. bank, mouse

    ***mouse^1**: … a mouse controlling a computer system*
    ***mouse^2**: … a quiet animal like a mouse*
    ***bank^1**: … a bank hold the investment in a custodial account…*
    ***bank^2**: … the bank on the east river*

- Words don't appear in isolation. The word use (e.g., syntax and semantics) depends on its context. Why not learn the representations for each word in its context?

WORD EMBEDDING - WORD2VEC

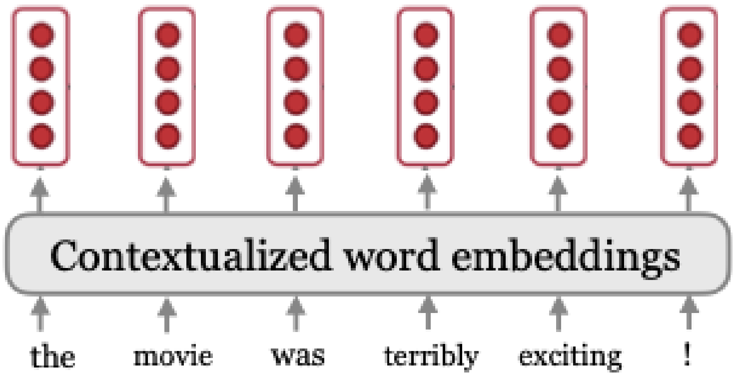
- **Main problem:**
  - Word embeddings do not consider language context
  - Just looks at co-occurrence statistics



- **Solution**: Train *contextual* representations on text corpus

CONTEXTUAL WORD EMBEDDING

- Build a vector for each word conditioned on its **context!**
- The representation of each token is a function of the entire input sentence



$$g : (w_1, w_2, \ldots, w_n) \longrightarrow \mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$$

CONTEXTUAL WORD EMBEDDING

Compute contextual vector

$$\mathbf{c}_k = f(w_k \mid w_1, w_2, \ldots, w_n) \in \mathbb{R}^d$$

$$f(\text{play} \mid \text{Elmo and Cookie Monster play a game})$$

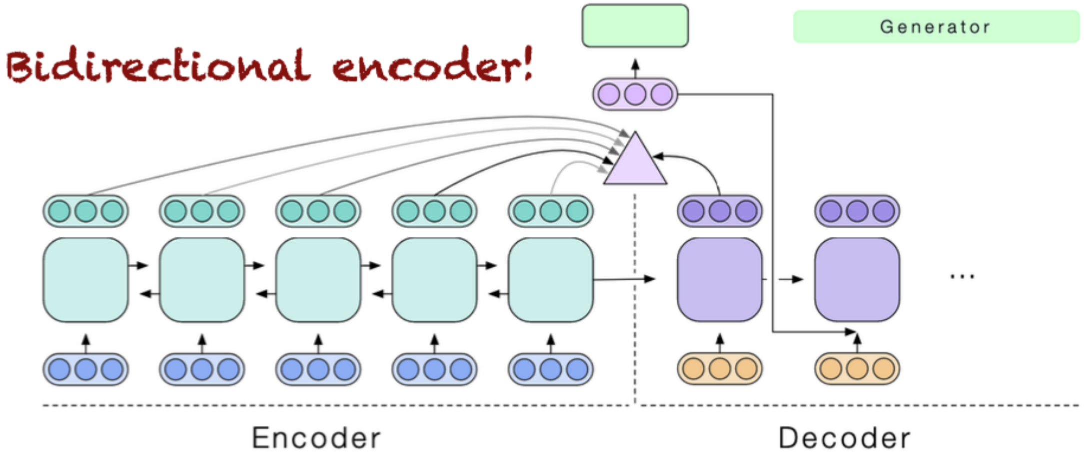
$$\neq$$

$$f(\text{play} \mid \text{The Broadway play premiered yesterday})$$

CONTEXTUAL WORD EMBEDDING - COVE

CoVe Architecture:

- Encoder: two-layer bidirectional LSTM
- Decoder: two-layer unidirectional LSTM

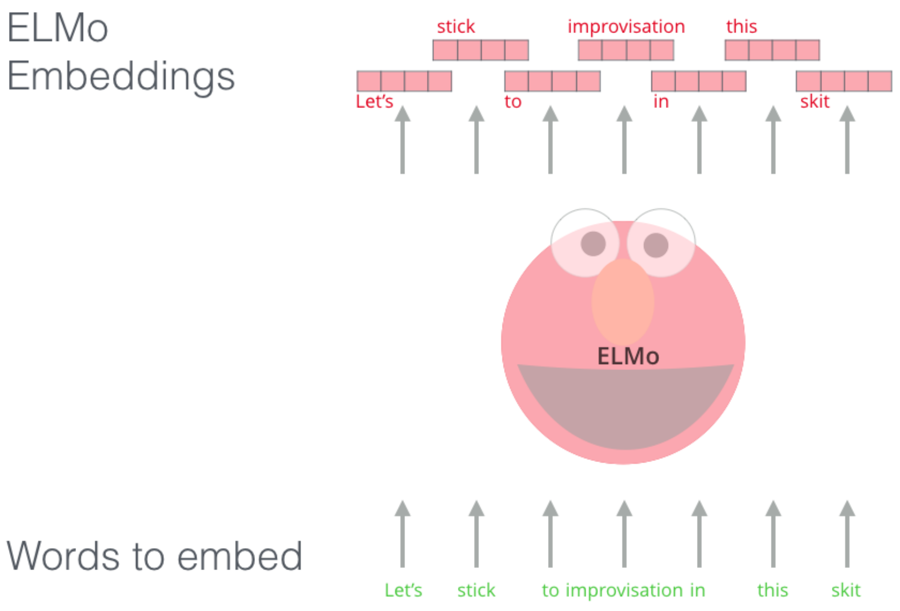


$\text{CoVe}(w) = \text{MT-LSTM}(\text{GloVe}(w))$ $\tilde{w} = [\text{GloVe}(w); \text{CoVe}(w)]$

CONTEXTUAL WORD EMBEDDING - ELMO

Main Idea:

- ELMo looks at the entire sentence before assigning embedding to each word.



CONTEXTUAL WORD EMBEDDING - ELMO

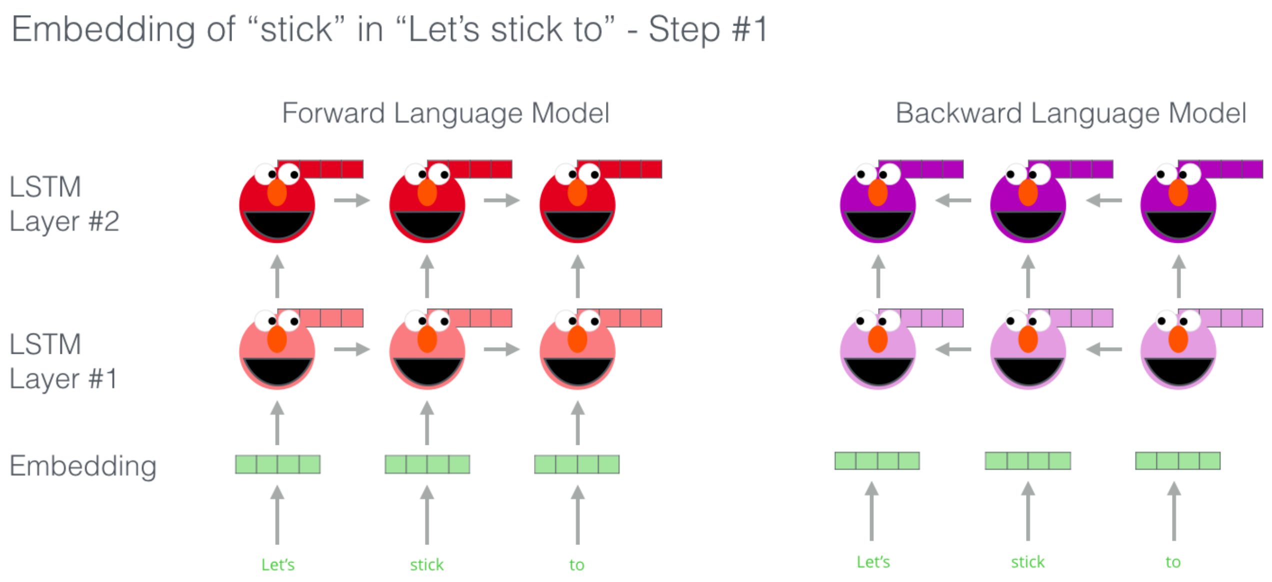
Main idea:

- Train a forward LSTM-based LM and a backward LSTM-based LM on some large corpus
- Use the hidden states of the LSTMs for each token to compute a vector representation of each word.
- bi-directional LSTM trained on a specific task to be able to create those embeddings.
- ELMo provided a pre-trained model to be fine-tuned in our domain of application.
    - ELMo LSTM would be trained on a massive dataset and then used as a component in other models that need to handle language.

CONTEXTUAL WORD EMBEDDING - ELMO

ELMo Architecture:

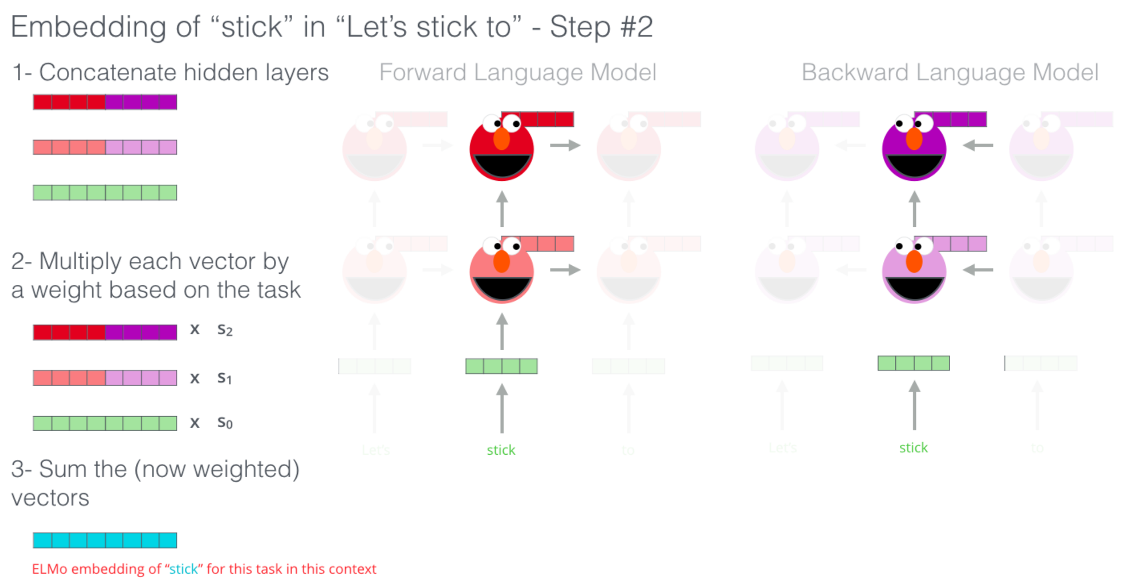
- Trained to predict next word in a sequence (language modelling)
  - allows the model to pick up on language patterns.
- Trains bi-directional so that the language model have a sense of the next word and the previous word.



CONTEXTUAL WORD EMBEDDING - ELMO

ELMo contextual embedding is obtained by:

- Concatenating the hidden layers and initial embedding
- Weighted summation



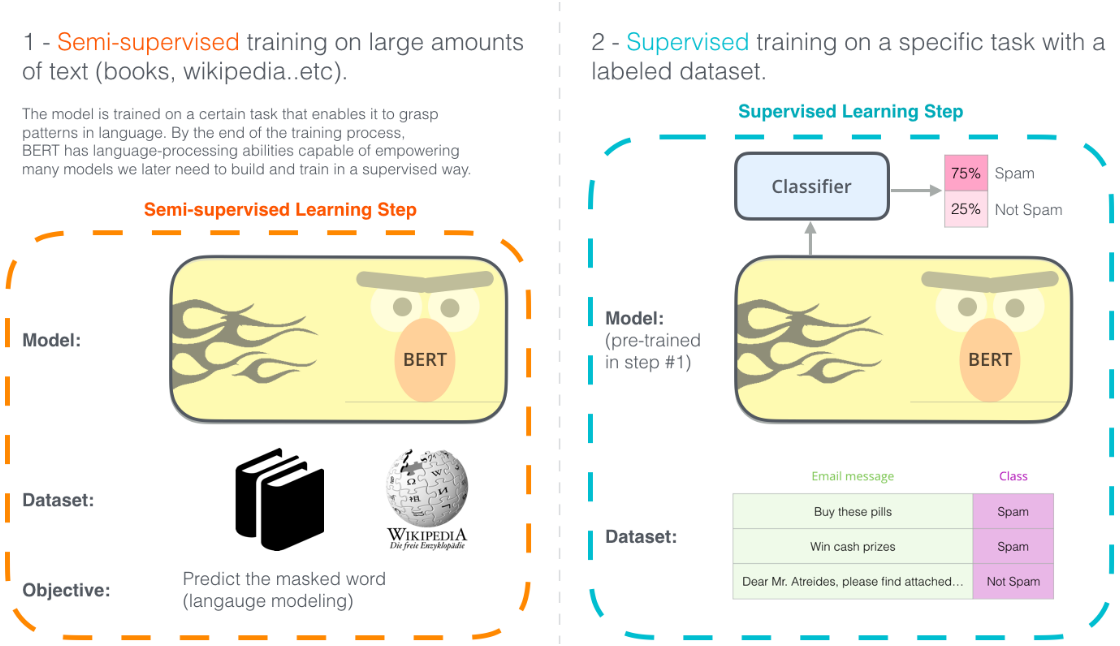
LIMITATIONS OF ELMO/COVE

- Task-specific architectures: Contextualized word embeddings are used as an augmentation to static word embeddings.
- Trained on single sentences.
- Training corpus is much smaller than those used for training word2vec/GloVe vectors.

BERT

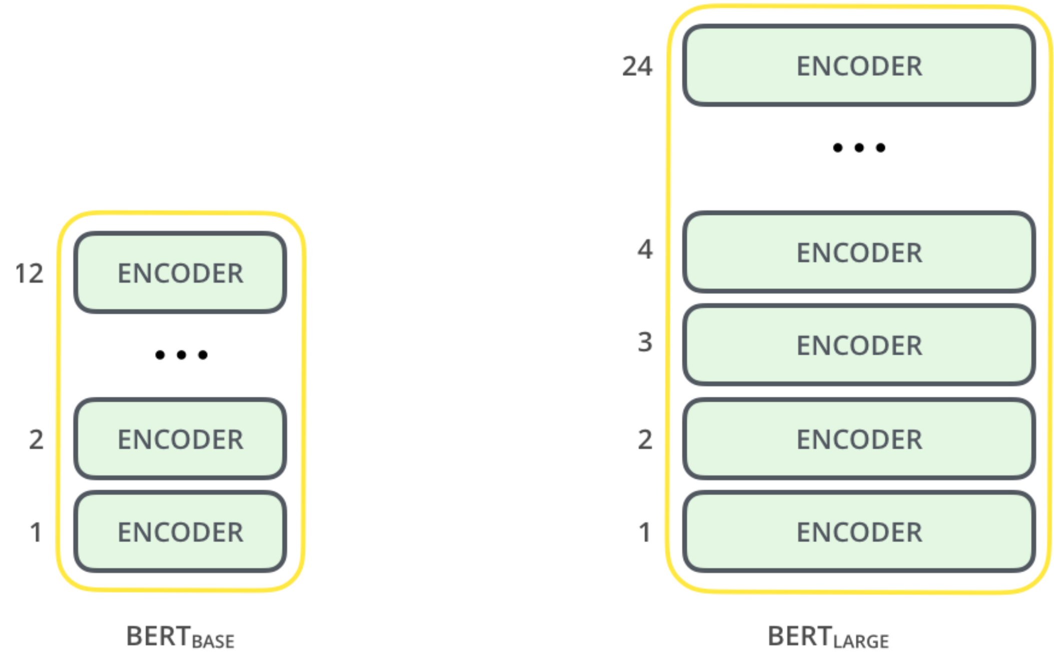
Two steps of how BERT is developed

- Step 1: trained on unannotated data
- Step 2: fine-tuned for specific task



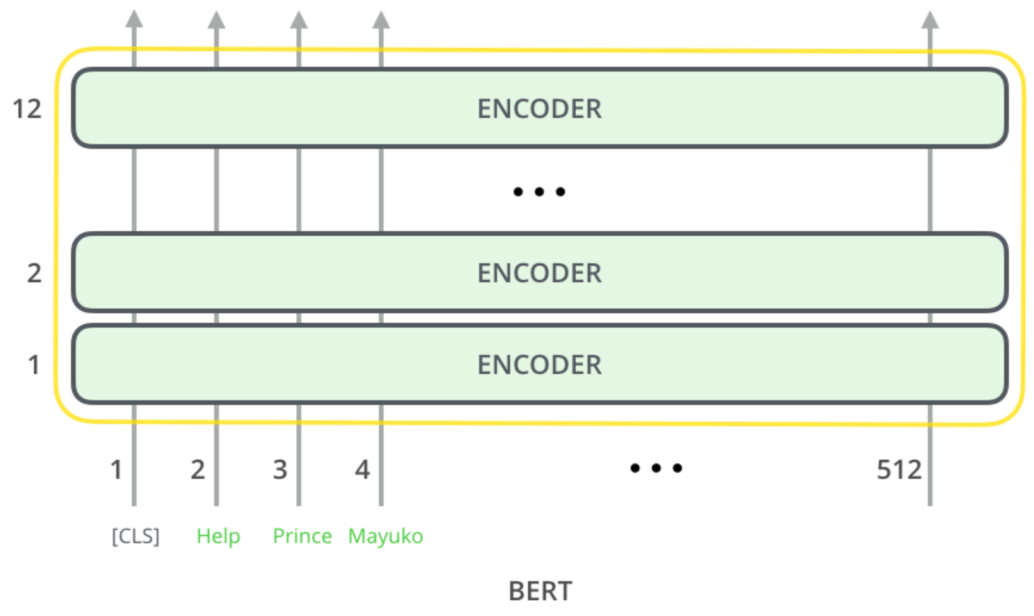
BERT – MODEL ARCHITECTURE

- BERT is a trained Transformer Encoder stack
- The paper presented two models
    - $\text{BERT}_{\text{BASE}}$
        - Stack of 12 Encoder layers
        - Feedforward NN 768 hidden units
        - 12 Attention heads
    - $\text{BERT}_{\text{LARGE}}$
        - Stack of 24 Encoder layers
        - Feedforward NN 1024 hidden units
        - 16 Attention heads



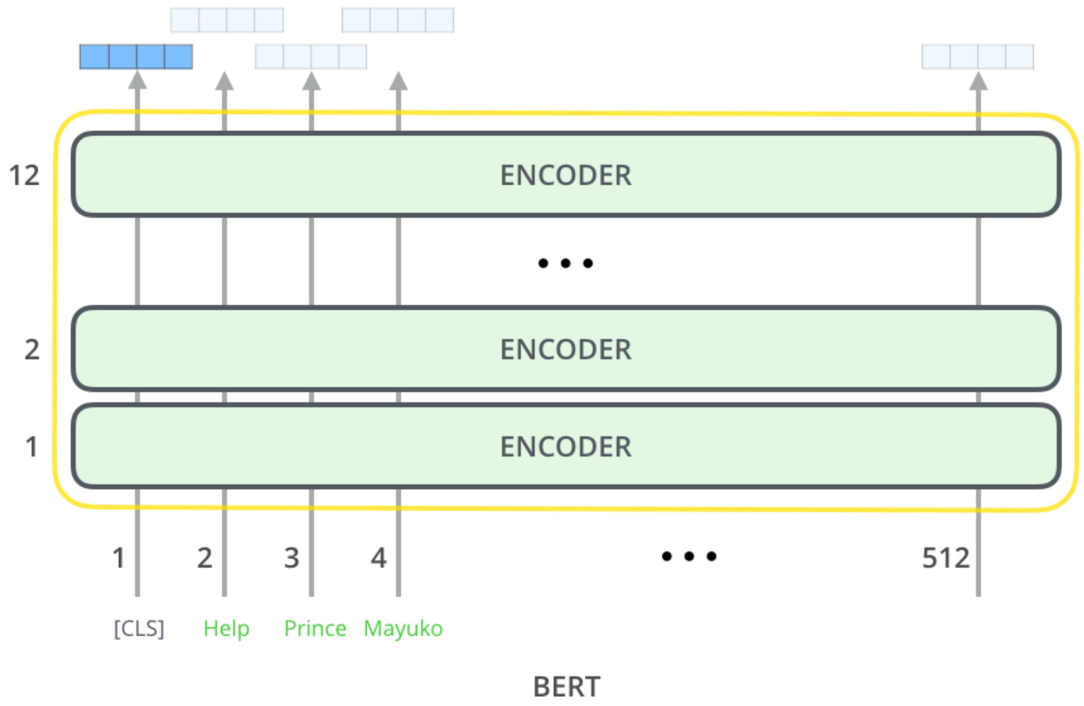
BERT – MODEL INPUT

- The first input token is a [CLS] token
- BERT takes as input the entire sequence of words
- Each layer applies self-attention and passes its result through a feedforward network and hands it to the next encoder layer



BERT – MODEL OUTPUT

- Each position outputs a vector of size *hidden_size* (768 in BERT Base)
- Vector for each word represents the contextual word embedding for that word



BERT – TRAINING

- BERT is based on the Transformer architecture
  - the attention mechanism learns the contextual relationship between words (or sub words) in a text.

- BERT uses only the encoder mechanism of the Transformer
  - This is because its goal is to generate a language model

- Most language model are bi directional thus limiting the ability to learn complete context. BERT using two training strategies to overcome this
  - MASK Language Model (MLM)
  - Next sentence prediction (NSP)

BERT – TRAINING

MASK LM training:

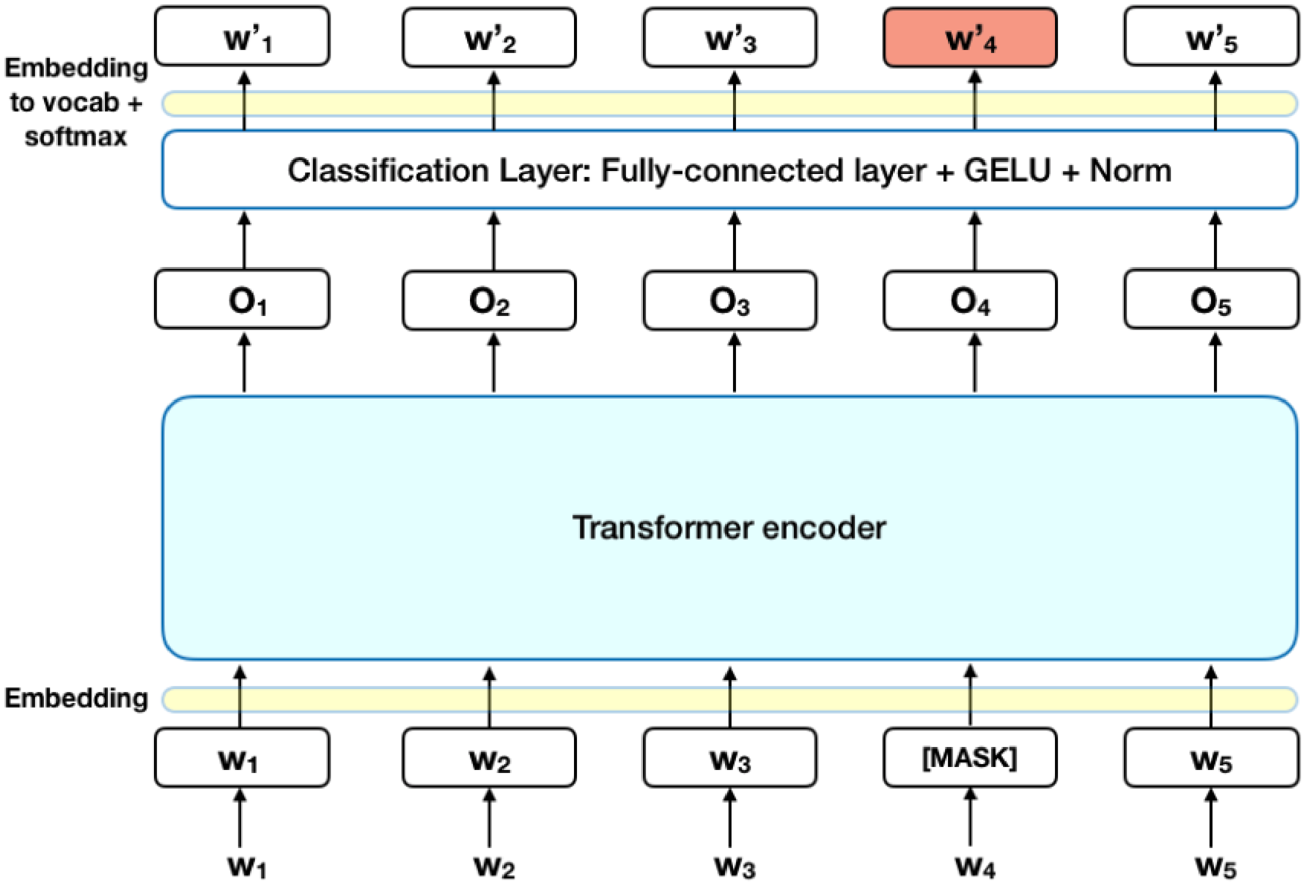
- BERT mask about 15% of words in each sentence before feeding into the model.
- The model then attempts to predict the original value of the masked words based on context provided by the other non-masked words in the sentence.

- Thus BERT has an incorporated
  1. a classification layer on top of the encoder output
  2. Multiplying the output vectors by the embedding matrix, transforming them into the vocabulary dimension
  3. Calculating the probability of each word in the vocabulary

BERT loss function is estimated based only on the prediction of the masked words and ignores the non-masked words.

BERT – TRAINING

MASK LM training:



BERT – TRAINING

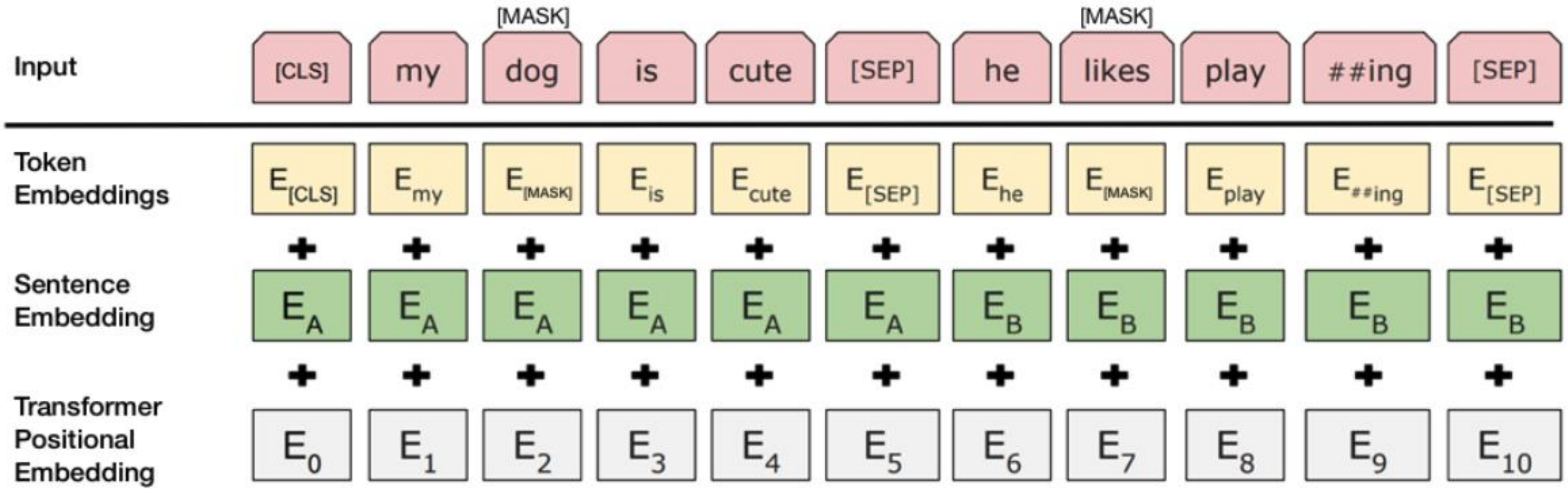
Next Sentence prediction:

- Model receive pairs of sentences and aims at predicting if the second sentence in the subsequent sentence in the original document.
    - 50% of the inputs are consecutive sentence as in the original document
    - The other 50% have random consecutive sentence pairs from the corpus

- To help the model distinguish between two sentences
    1. Add [CLS] token at the beginning of the first sentence
    2. Add [SEP] token at the send of each sentence
    3. Sentence embedding indicting sentence A or B is added to each token
    4. Positional embedding is added to each token to indicate its position

BERT – TRAINING

Next Sentence prediction:



BERT – TRAINING

Next Sentence prediction:

- Model receive pairs of sentences and aims at predicting if the second sentence in the subsequent sentence in the original document.
  - 50% of the inputs are consecutive sentence as in the original document
  - The other 50% have random consecutive sentence pairs from the corpus

- To help the model distinguish between two sentences
  1. Add [CLS] token at the beginning of the first sentence
  2. Add [SEP] token at the send of each sentence
  3. Sentence embedding indicting sentence A or B is added to each token
  4. Positional embedding is added to each token to indicate its position

REFERENCES