CMPE 297, INSTRUCTOR: JORJETA JETCHEVA

TRANSFER LEARNING FOR NLP, AND CONTEXTUAL EMBEDDINGS

OUTLINE

- Encoder-Decoder models
- Bidirectional transformers
- Contextual embeddings
- Transfer learning
- Pretaining and fine-tuning
- Masked language modeling

THE ENCODER-DECODER MODEL

- Encoder-decoder networks (aka sequence-to-sequence networks) are models that generate output sequences
- Example applications:
  - Machine translation
  - Summarization
  - Question Answering

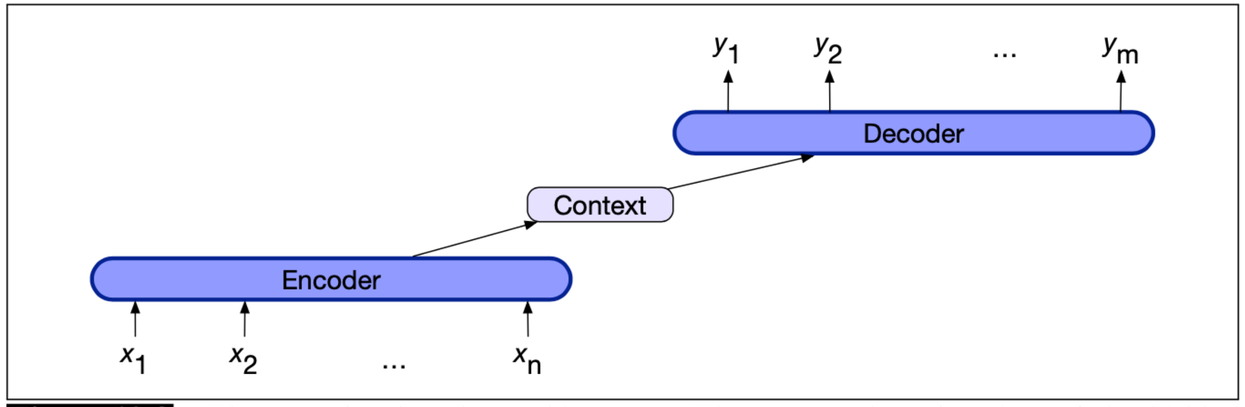


Figure 10.3 The encoder-decoder architecture. The context is a function of the hidden representations of the input, and may be used by the decoder in a variety of ways.

- The encoder creates a contextualized representation of the input (called context) and passes it to the decoder which then generates some task-specific output sequence

THE ENCODER-DECODER MODEL (CONT.)

- The encoder can be an RNN, LSTM or Transformer for example
- The decoder can also be an RNN, LSTM, Transformer, etc.
  - And can generate an arbitrary length output sequence

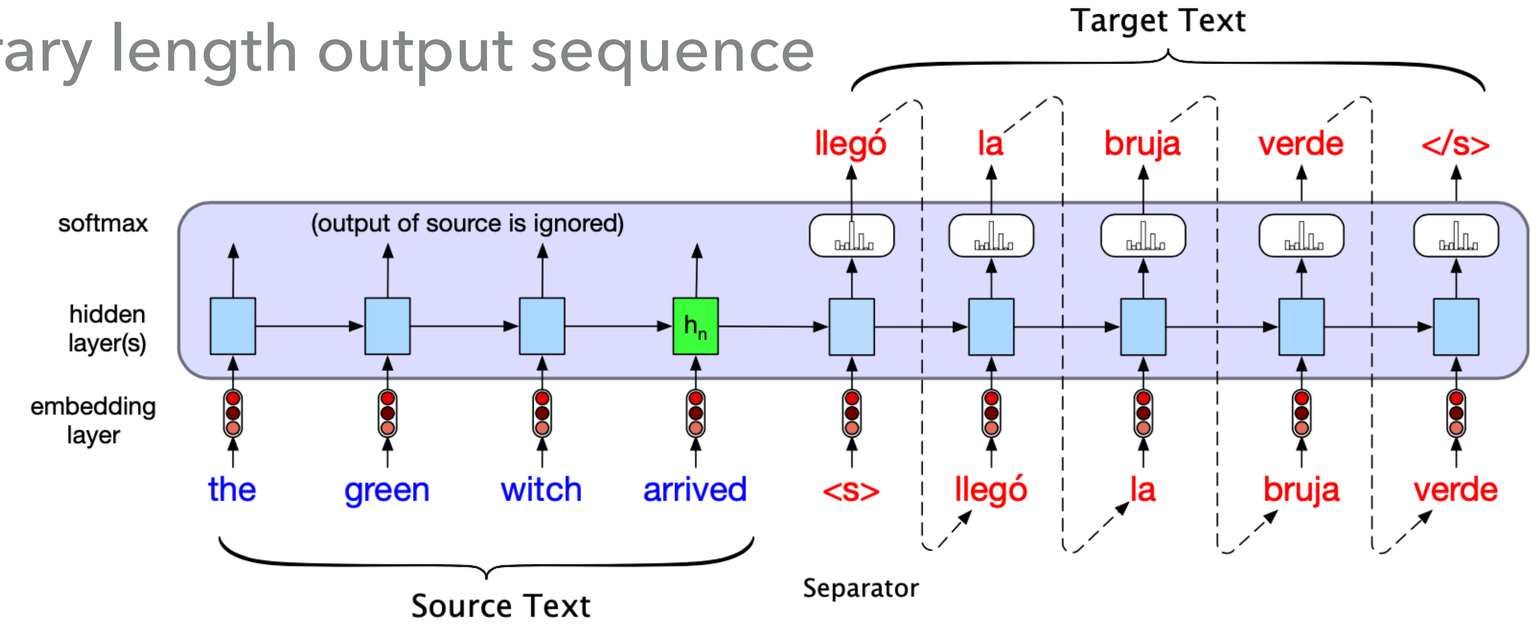
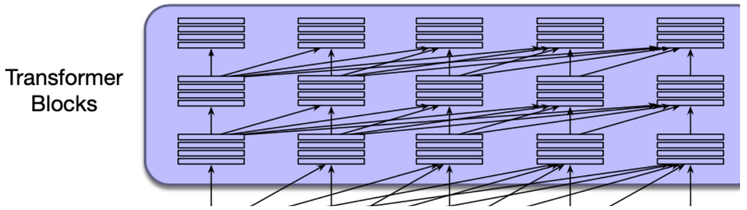


Figure 10.4 Translating a single sentence (inference time) in the basic RNN version of encoder-decoder approach to machine translation. Source and target sentences are concatenated with a separator token in between, and the decoder uses context information from the encoder's last hidden state.

HOW DO HUMANS LEARN WORDS?

- Kids learn words at a rate that cannot be explained by explicit instruction and grounding of the words with real-life phenomena/experiences
- Hypothesis: Kids learn words (by reading extensively) based on the words they co-occur with (contextual learning based on the distributional hypothesis)
  - Distributional hypothesis: words which are synonyms (like oculist and eye-doctor) tend to occur in the same environment (e.g., near words like eye) and the difference in meaning between two words "corresponding roughly to the amount of difference in their environments" (Harris, 1954)
- This motivates the use of contextual embeddings: representations for words in context - each word will be represented by a different vector each time it appears in a different context

CONTEXTUAL EMBEDDINGS



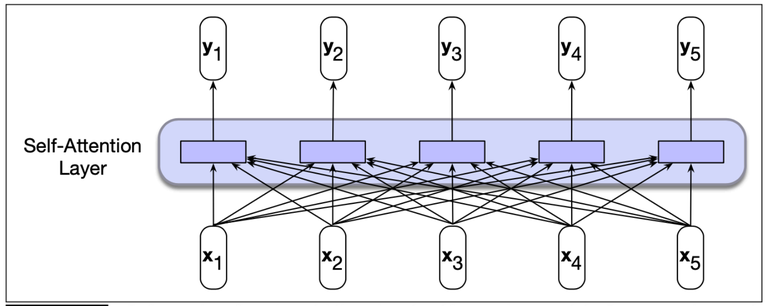


Figure 11.2 Information flow in a bidirectional self-attention model. In processing each element of the sequence, the model attends to all inputs, both before and after the current one.

- A pre-trained language model applied to a novel input sequence, creates contextual embeddings for each token in the sequence
- Contextual embeddings are vectors that reflect an aspect of the meaning of a token in the context of an input sequence
- Given an input sequence $x_1, \ldots, x_n$, the final output vector, y_i, from the final layer of the model can be thought of as representing each token x_i in the context of the sentence $x_1, \ldots, x_n$
- Alternatively, the vectors for y_i for each of the last four layers of the model can be averaged

BIDIRECTIONAL TRANSFORMERS

- Used in state-of-the-art NLP models like BERT, and its descendants (e.g., RoBERTa, SpanBERT, DNABert)
  - We use context from before and after!
  - BERT stands for Bidirectional Encoder Representations from Transformers
- Uses bidirectional self-attention – all inputs (not just those preceding an input) are used to create a contextualized version of the input
- The output vectors are output embeddings that are contextualized versions of the input embeddings
- Same computation as for regular transformers except there is no need to zero out the top part of the query-key comparison matrix

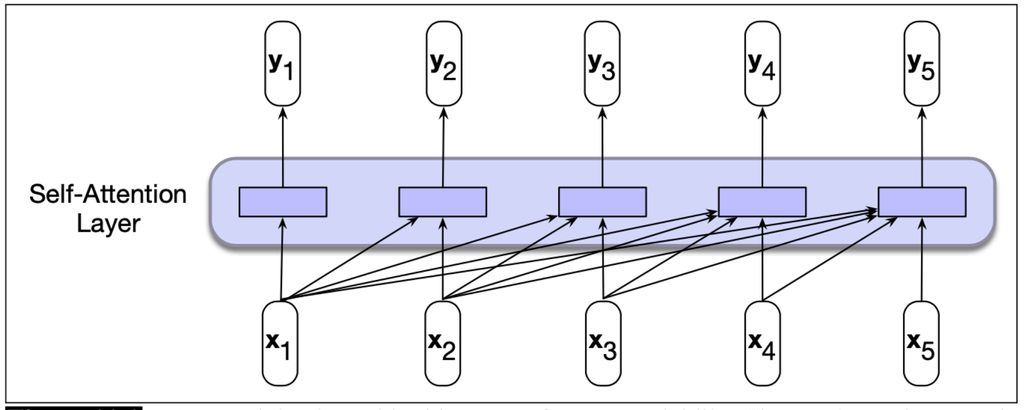


Figure 11.1 A causal, backward looking, transformer model like Chapter 9. Each output is computed independently of the others using only information seen earlier in the context.

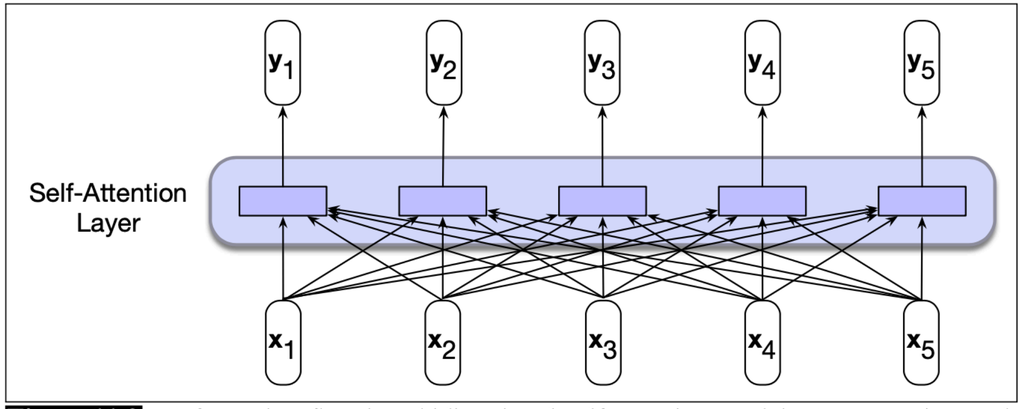


Figure 11.2 Information flow in a bidirectional self-attention model. In processing each element of the sequence, the m ter the current one.

q1•k1	q1•k2	q1•k3	q1•k4	q1•k5
q2•k1	q2•k2	q2•k3	q2•k4	q2•k5
q3•k1	q3•k2	q3•k3	q3•k4	q3•k5
q4•k1	q4•k2	q4•k3	q4•k4	q4•k5
q5•k1	q5•k2	q5•k3	q5•k4	q5•k5

ADDITIONAL DETAILS ABOUT THE ORIGINAL BERT MODEL

- Uses subword tokenization (WordPiece algorithm) - 30,000 tokens
- 12 layers of transformer blocks, with 12 multi-head attention layers each
- 100 million parameters total
- BERT has a fixed input size of 512 subword tokens (transformer memory requirements grow quadratically with the size of their input)

TRAINING BIDIRECTIONAL ENCODERS

- Because the model now has access to all words in the text, it no longer makes sense to train the model to predict the next word
- Instead we formulate the problem as a fill-in-the-blank task (called cloze task)

Please turn your homework ___

vs. Please ___ your ___ in (cloze)

The logic behind this is that we train the model on all words, then we hide the words and let the model decide what word could be used to fill in the blank, using probability distrubution. it then understands what word/token could be used.

- We corrupt the input sequence by masking out words and the model generates a probability distribution over the vocabulary for each missing word/token

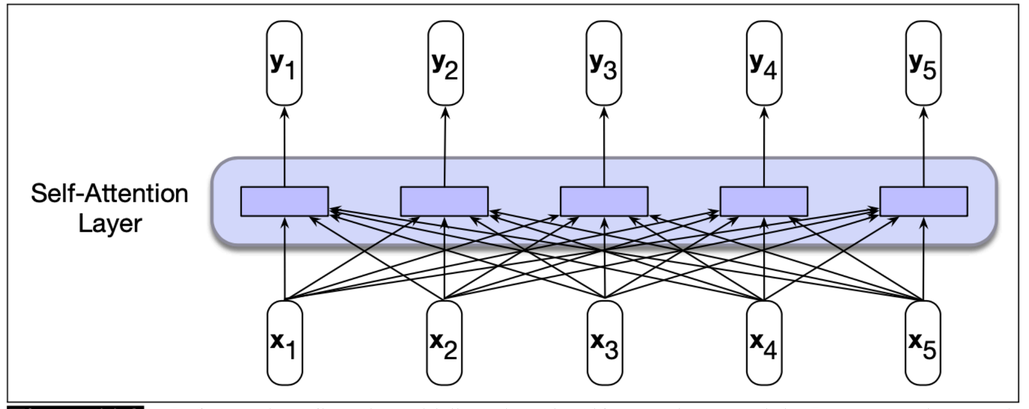


Figure 11.2 Information flow in a bidirectional self-attention model. In processing each element of the sequence, the model attends to all inputs, both before and after the current one.

N	q1•k1	q1•k2	q1•k3	q1•k4	q1•k5
	q2•k1	q2•k2	q2•k3	q2•k4	q2•k5
	q3•k1	q3•k2	q3•k3	q3•k4	q3•k5
	q4•k1	q4•k2	q4•k3	q4•k4	q4•k5
	q5•k1	q5•k2	q5•k3	q5•k4	q5•k5

N

MASKING WORDS

Basically, we are challenging the model, removing some words or replacing it with random words that dont make sense, And force it check whether its correct or not.

- In Masked Language Modeling (MLM), we pick a random sample of the tokens and do one of the following:
  - Replace each one with the unique token [MASK]
  - Replace it with a random token from the vocabulary (randomly sampled based on token unigram probabilities)
  - Leave the token unchanged
- In BERT 15% of the input tokens are sampled for learning and of those:
  - 80% are repacked with [MASK]
  - 10% are replaced with randomly selected tokens
  - 10% are left unchanged

80% of that sampled 15% is replaced

MASKING WORDS – EXAMPLE

- Note that subword tokens are typically used (example in figure shows word tokens)
- Positional embeddings are combined with the subword tokens
- Some words are masked, one is replaced with a random word (“apricot”)
- Each masked/replaced word is predicted using a feed-forward classification layer with weight matrix W_V
- Output is a probability distribution over the vocabulary
- Cross-Entropy loss is computed relative to the ground truth token
- Weights are updates based on average loss over the sampled learning items

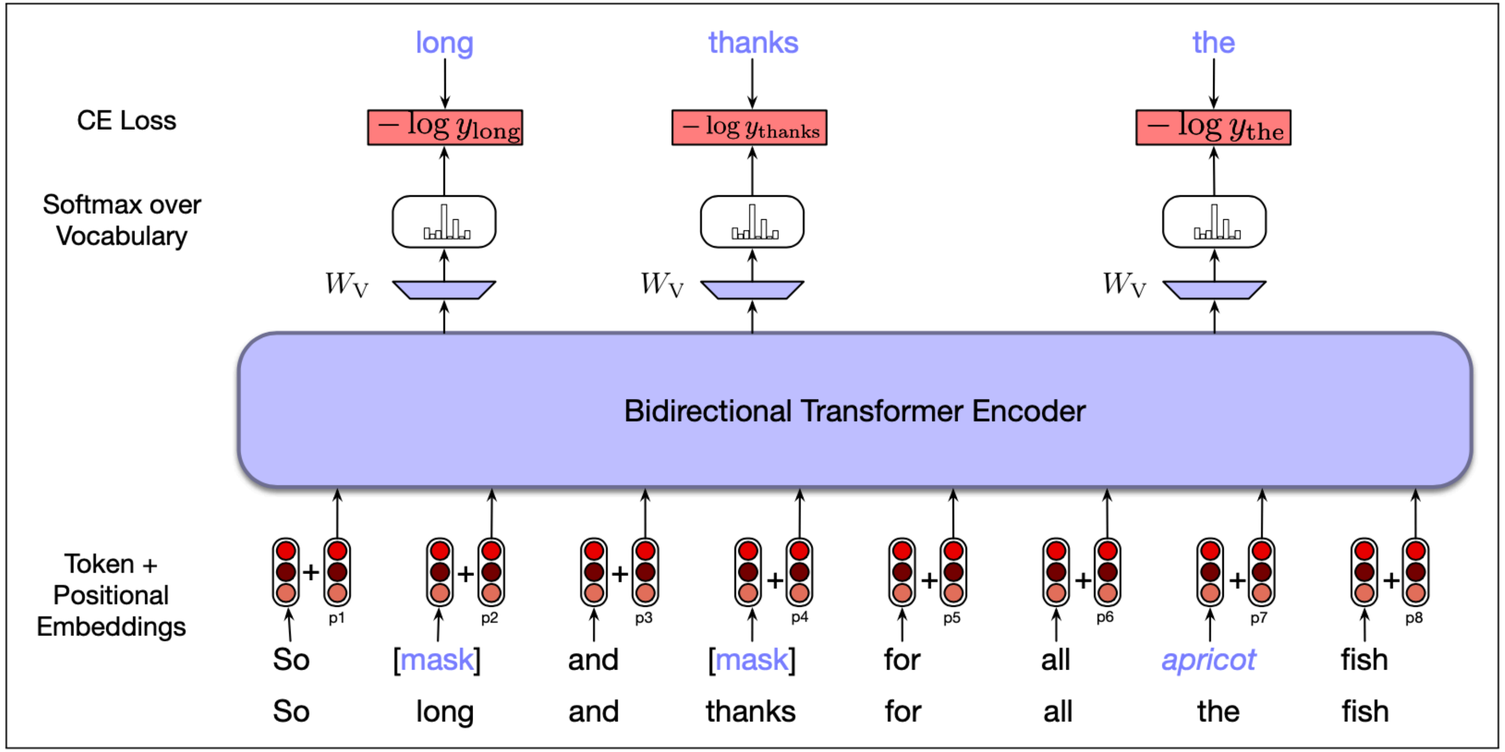


Figure 11.5 Masked language model training. In this example, three of the input tokens are selected, two of which are masked and the third is replaced with an unrelated word. The probabilities assigned by the model to these three items are used as the training loss. (In this and subsequent figures we display the input as words rather than subword tokens; the reader should keep in mind that BERT and similar models actually use subword tokens instead.)

MASKING SPANS

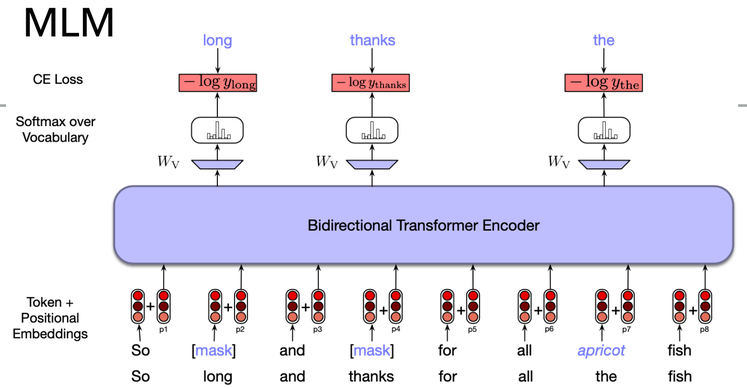
- Identification and classification of phrases for applications such as question-answering, requires learning to predict a span of text
- A span is a continuous sequence of one or more words from the training text (prior to tokenization)
- Span-based masking involves masking entire spans

In SpanBert (Joshi et al., 2020):

- A span length is sampled from a geometric distribution with upper bound of 10
- The location of the span is chosen uniformly among possible spans within a particular text
- All tokens in a span undergo the same replacement strategy, and 80% of spans are replaced by [MASK], 10% by randomly sampled vocabulary words, 10% are left as is (same as BERT)
- Total token substitution is limited to 15% of the training sequence input

MASKING SPANS (CONT.)

- The MLM predicts the loss for a specific word (e.g., “thanks” in the example)
- That is added to the loss for the prediction of each word relative to the text before and after the span (e.g., “long” and “all”)
  - FFN network has the word before and the word after the whole span, and the positional embedding for the word we are predicting
  - The FFN-based portion is referred to as Span Boundary Objective (SBO)



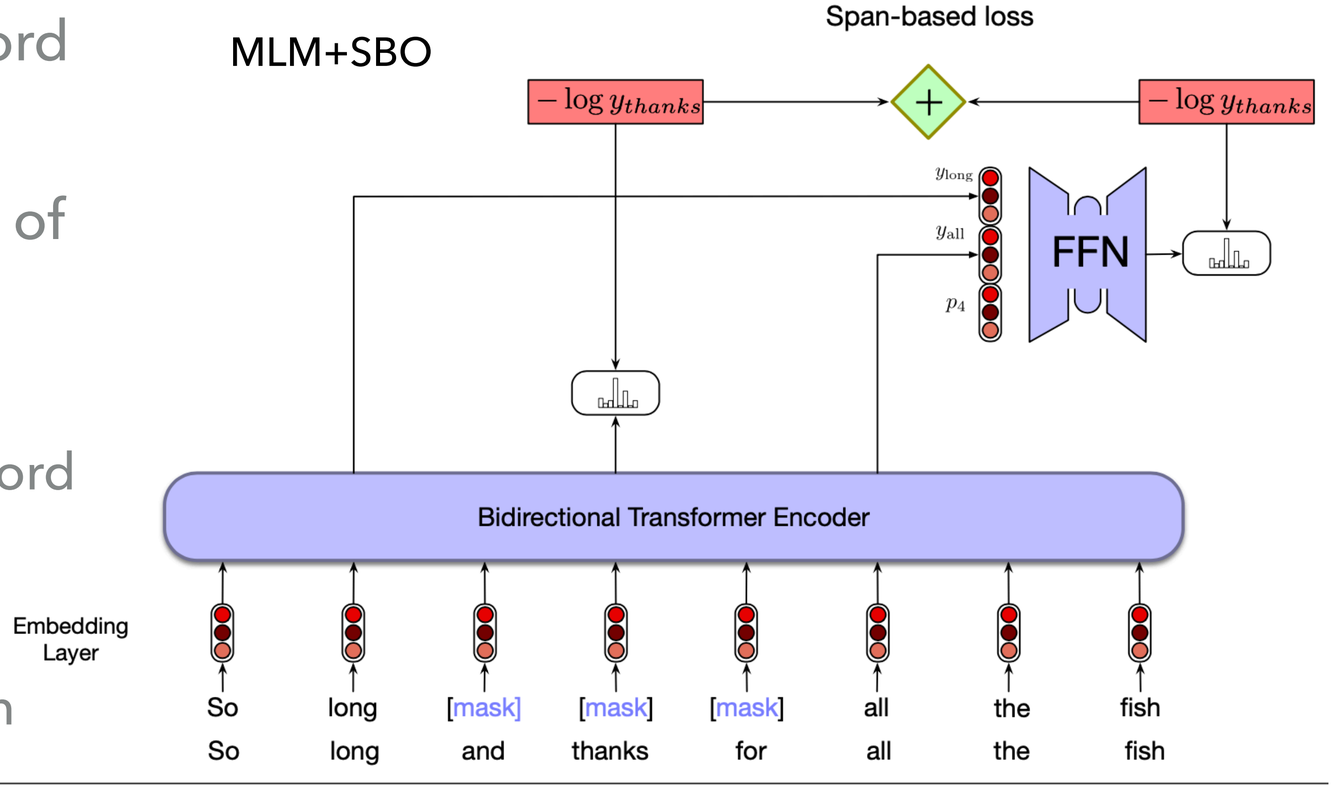


Figure 11.6 Span-based language model training. In this example, a span of length 3 is selected for training nd all of the words in the span are masked. The figure illustrates the loss computed for word *thanks*; the loss or the entire span is based on the loss for all three of the words in the span.

NEXT SENTENCE PREDICTION (NSP)

- Determining the relationships between pairs of sentences is important for a number of applications, including:
  - Paraphrase detection (do 2 sentences have similar meanings)
  - Entailment (do 2 sentences entail or contradict each other)
    - Entailment means that the truth of one text fragment (e.g., sentence) follows from another text fragment (e.g., sentence)
    - Example:

      a: I'm confused.

      b: Not all of it is very clear to me.
  - Discourse coherence (do 2 neighboring sentences form a coherent discourse)

NEXT SENTENCE PREDICTION (NSP) WITH BERT

- BERT asked to predict whether a pair of sentences are actually adjacent in the training corpus or unrelated
- 50% of training pairs were positive pairs
- In the other 50%, the second sentence of a pair was randomly chosen from some other part of the corpus
- NSP loss is based on how well the model can differentiate true pairs from random pairs

NEXT SENTENCE PREDICTION WITH BERT

CLS at the front of sequence.
SEP would tell us where the sentence.
From the diagram s1 , s2 sentence 1 and sentence 2

- After tokenizing the input, two tokens are added:
  - a [CLS] token is pre-pended to the input sequence pair
  - A [SEP] token is placed between the two sentences

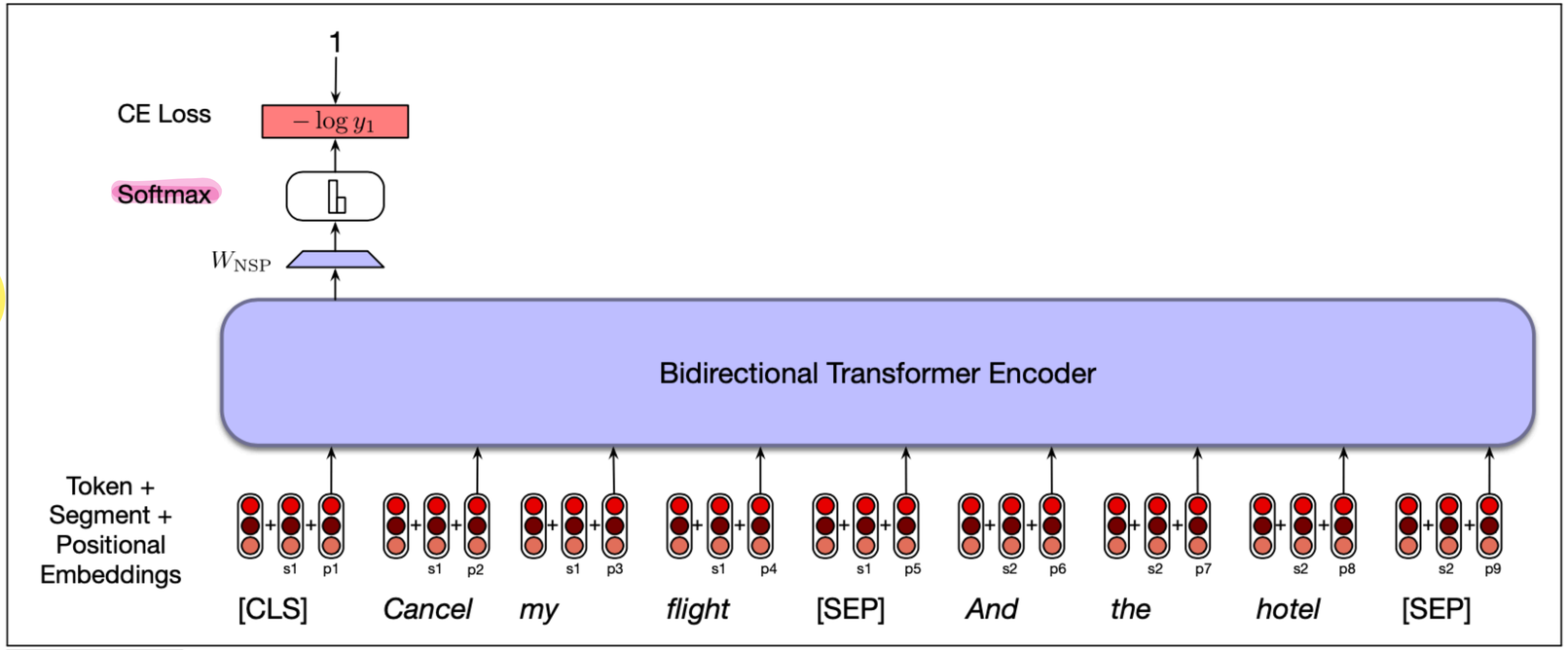


Figure 11.7 An example of the NSP loss calculation.

- Then two positional embeddings are added (one for each sentence) to help the model distinguish between the two sentences
- The output vector associated with CLS represents the next sentence prediction and is passed to a FFN which does binary classification: pair is adjacent or not

Content Embeddding stored in CLS token is sent to the feedfoward network

BERT TRAINING

- The original BERT was trained on 3.3 million words
  - 800 million word corpus of book texts (BookCorpus) and
  - 2.5 billion word corpus from the English Wikipedia
- Next sentence prediction was performed but sentences were sampled so that each pair would result in a less than 512 token length input
- It took 40 epochs (passes over the training set) for the model to converge

TRANSFER LEARNING THROUGH FINE TUNING

We pretrain a lang model to learn to speak english, THEN we speicilaize it for speicifc task.
Then we fine tine it, after learning english , learning medical applcations, much less data, much less training;

- Pre-trained language models extract generalizations from large text corpuses
- We can use this "knowledge" for downstream applications by interfacing the pre-trained model with the downstream application through a process called fine tuning
- Fine tuning
  - Involves adding a small set of application-specific parameters to the model
  - Uses labeled data to train the application-specific model parameters
  - The pre-trained model may be frozen, or allow small adjustments during this training

TRANSFER LEARNING THROUGH FINE TUNING: SEQUENCE CLASSIFICATION

- Example to keep in mind: sentiment analysis
- A [CLS] token is is added to the vocabulary and is pre-pended to all input sequences during training and prediction
- The output vector for the [CLS] input serves as a representation for the entire input sequence
- This vector is then passed onto a feedforward neural network (or other model, e.g., logistic regression) that makes the final classification decision
- Training updates the FFN weights, and can be allowed to adjust the pre-trained model
- Recall that in RNNs we used the hidden layer associated with the final input token as input to the FFN

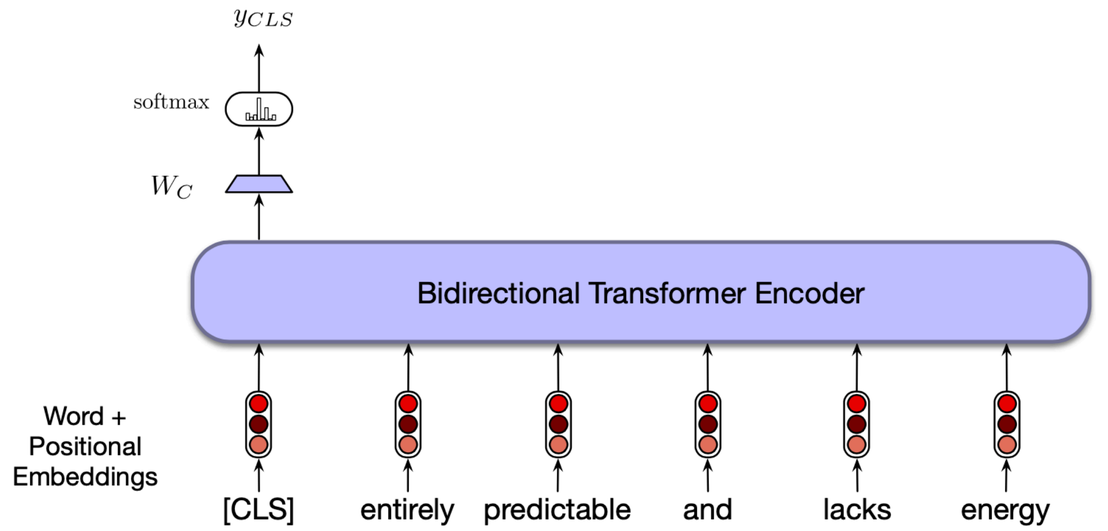
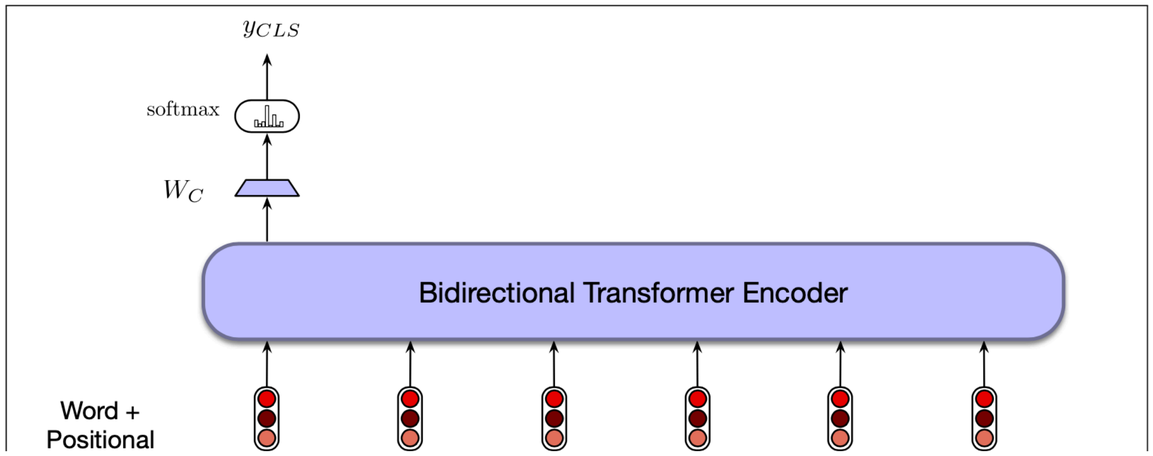


Figure 11.8 Sequence classification with a bidirectional transformer encoder. The output vector for the [CLS] token serves as input to a simple classifier.

TRANSFER LEARNING THROUGH FINE TUNING: PAIR-WISE SEQUENCE CLASSIFICATION

Are 2 sentences connected? Multiclass classification problem.

- Pair-wise sequence classification is used in applications like paraphrase detection and logical entailment
- We pass premise/hypothesis (i.e, first/second sentence) pairs through a bidirectional encoder with a pre-pended [CLS] and [SEP] between the sentences and the rest is the same as for the sequence classification example



Example from the Multi- Genre Natural Language Inference (MultiNLI) dataset:

Neutral

a: Jon walked back to the town to the smithy.

b: Jon traveled back to his hometown.

Contradicts

a: Tourist Information offices can be very helpful.

b: Tourist Information offices are never of any help.

Entails

a: I'm confused.

b: Not all of it is very clear to me.