Using BERT for Sentiment Analysis

Outline

Input is several sentences	Split sentences	Split sentences to words/tokens	Represent each token (position) in its given context	Represent sentence by concatenating token-level representations	Use this representation to perform specific task
	Preprocessing	Preprocessing	The model represents words is trained by masking and next-sentence		Fine-tuning

Outline

Data Preprocessing: Get data into format BERT understands

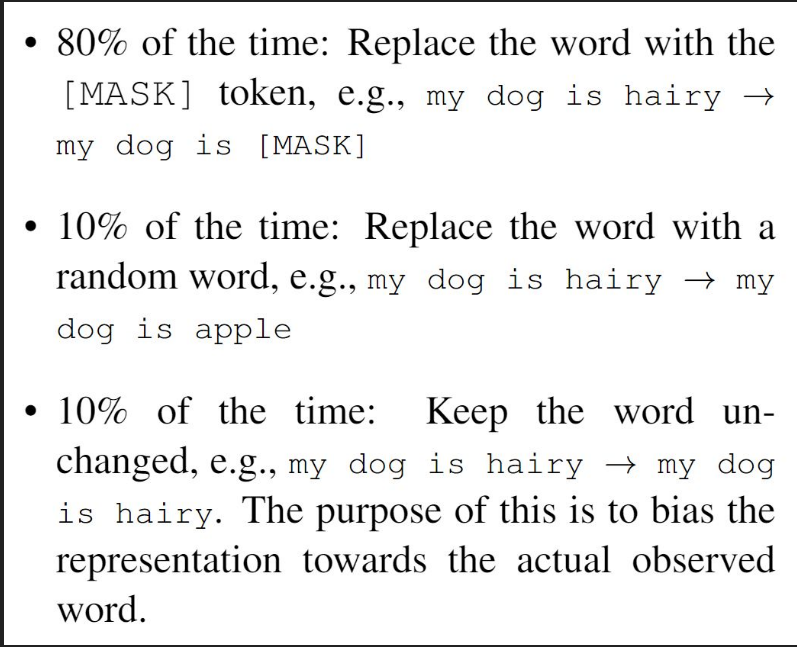
- Lowercase everything
- Tokenizes sentence to words, then words to WordPieces
  - (“calling” => [“call”, “##ing”])
- WordPieces are within corpus of 30,000 tokens
  - Constrained vocabulary of units so it’s easier to process on
  - Handles rare words more effectively
- Add “CLS” and “SEP” tokens

Pretraining

- Data: BooksCorpus and English Wikipedia (massive continuous text)
- Fine contextual understanding (word-level)
- Coarse contextual understanding (sentence-level)

Pretraining, Fine

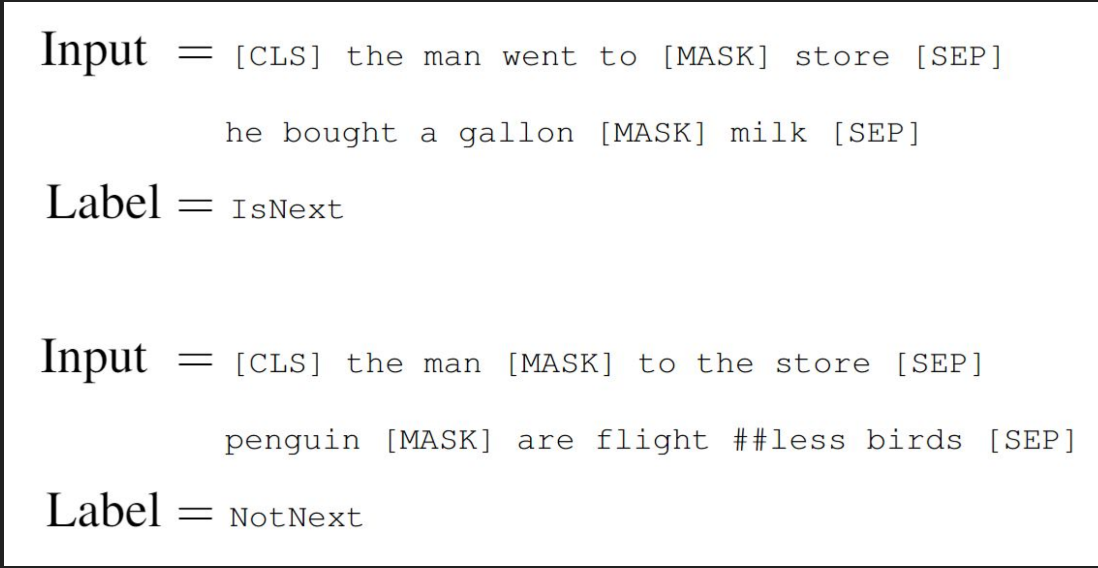
- Consider “bank” in “river bank” and “bank account”
  - Word2Vec represents them the same way (context-free)
  - BERT uses contextual representations of words
- Train model to predict words based on *entire general* context (bidirectional, random masking) instead of the *previous specific* context (left-directional)
- 15% chance a word gets chosen



- 80% of the time: Replace the word with the `[MASK]` token, e.g., `my dog is hairy` → `my dog is [MASK]`
- 10% of the time: Replace the word with a random word, e.g., `my dog is hairy` → `my dog is apple`
- 10% of the time: Keep the word unchanged, e.g., `my dog is hairy` → `my dog is hairy`. The purpose of this is to bias the representation towards the actual observed word.

Pretraining, Coarse

- Train model to predict if some sentence is *next* sentence given *current* sentence



```
Input = [CLS] the man went to [MASK] store [SEP]
        he bought a gallon [MASK] milk [SEP]
Label = IsNext

Input = [CLS] the man [MASK] to the store [SEP]
        penguin [MASK] are flight ##less birds [SEP]
Label = NotNext
```

Transformers

- Recurrence allows system to encode what was read before to better understand what is being read now
  - However, has bad long-range dependency, is greatly biased towards remembering words it saw most recently
- Use attention to **choose** which of prior words are **most important** to understanding the context as we continue reading
  - Far less bias on words that were just read, better long-range dependency
- Key Idea: Multi-Headed Block
  - One attention block captures some global context via weighted sum of hidden states
  - Use multiple attention blocks to capture many diff. contexts

Transformers High Level

- Machine Translation uses basic-encoder decoder model:
  - Map sequences to sequences via intermediate representation.Maximize probability of correct prediction at each step
- Intuitively the attention mechanism allows the decoder to “look back” at entire sequence sentence
- Attention gives encoder access to all hidden states
- Encoder focuses and ignores hidden states by using weights
  - Decoder passed weighted sum to predict next word
  - Computed originally w/ feed forward neural network
- 3 types of dependencies
  - Input and output
  - Input
  - output

Transformers low level

- Novelty of transformer is to allow decoder to see entire input sequence instead of left to right
- Multi-Head
    - Encoder hidden states(valued)
    - Decoder hidden state(query)
    - Block computes multiple attention weighted sums over values
        - Uses different linear transformations
- Scaled Dot Product Attention:
    - Attention(Q,K,V) = softmax(QK^T/sqrt(d_k))V
    - Transfomer rescales dot product to prevent from exploding to huge values
-

Training - Fine-Tuning

- Add a output layer to overall BERT architecture so that output is in correct format for task
- Train whole model for specific task on associated input/output using pre-trained BERT parameters as initial parameters
- Modify batch size, learning rate, # of training epochs for task
    - Hyperparameters found using a search method for specific task
- Our case we use:
    - Softmax to get probabilities of each outcome
    - Dropout with 90% keep rate to prevent overfitting

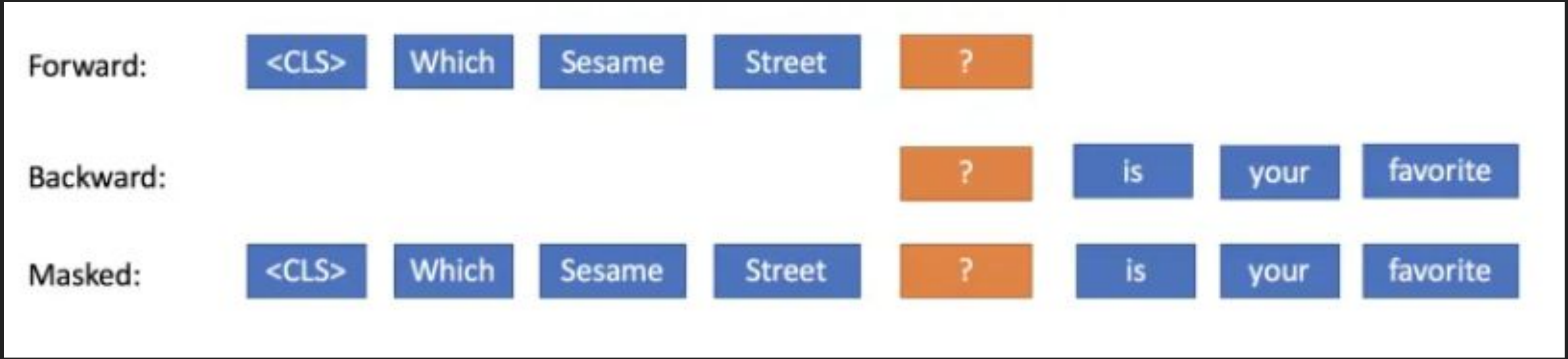
Results

- Project 1 training data accuracy
  - Positive: 0.89 on 25,000 instances
  - Negative: 0.89 on 25,000 instances
  - Total: 0.89 on 25,000 instances
- Project 1 testing accuracy: 86.08%
- Project 1 hidden accuracy: 86.67%
- Project 2 accuracy: 74.39%

Bidirectional Encoder Representations from Transformers

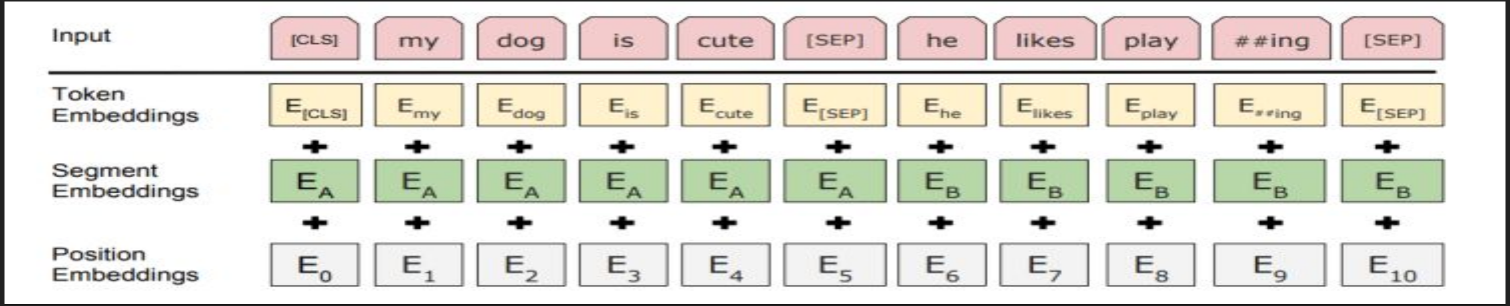
- Two main types of applying generic NLP models to specific tasks:
  - Feature based: Contextual representations of words as additional **features** (i.e. ELMo), task specific architectures
  - Fine-tuning based: Fine-tuning with few additional parameters, tuning of *all* parameters for task
- Both use unidirectional language models to learn representations
  - A word is represented by taking context of words that come before (but not after)

- BERT is a fine-tuning model
- Uses bi-directional representation of words
  - Word is represented by the context it appears in relative to words that come before **and** after
- This is in contrast to other models which generally only use information from right to left(instead of bidirectionally)



Preprocessing

- As input BERT takes token embeddings as well as additional meta-embeddings
- BERT addresses transformed positional lacking by using positional embeddings
- Takes segment embeddings as input(pairs of sentence to improve contextualization)


Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Masked Language Model

- Naively replace percentage with mask token and try to predict masked token
- BERT usually does this with 15%
- Additionally some random swapping of words done (~10%) of selected tokens to try to predict correctness regardless of what token is present

Next Sentence Prediction Training

- Done for sentence relation training. Sentences separated by [SEP] token. 50% of time second sentence is correct successor
- BERT must predict whether second sentence random

Fine Tuning

- BERT encoder produces sequence of hidden states
  - Need to be reduced to single vector for classification
- BERT just takes hidden state corresponding to first token
- Sentence representation can then be fed into any classifier, can be tuned w/ r on top of BERT

Hyperparameters

- Dropout: .1
- Batch: 32, 16
- Optimizer: Adam
- LR: 5e-5, 3e-5,2e-5
- Epochs: 3,4

Pretranining:

- Seq len: 256
- Batch: 512

Transformer Architecture

- RNNs used for a long time to do machine translation(mapping sentences to sentences). Problematic because they do not do well with large texts. Bottlenecked in intermediate representation
- Transformer composed of multiple attention blocks
- Stacks a layer that maps sequences to sequences
- Cannot take order of inputs into account(will treat the first and last toekns of inputs same if same word)
- BERT addressed by