Contextualized Representations

Instructor: Yoav Artzi
Overview

- Motivation
- Context-dependent Representations with BERT
- Advanced tokenization for BERT (and elsewhere)
- Cross-modality representations
Motivation

• Word embeddings (e.g., word2vec, GloVe):
  • Learn a vector for each word type
  • Always the same vector
  • Problem: each vector likely mixes multiple senses, regardless of how the specific instance of the word is used
Motivation

• Instead of a single vector: learn a different vector for each use of a word type

• Challenge: how do we define the space of uses? Isn’t it too large?

• Solution: use sentence encoders to create a custom vector for every instance of a word
Several Approaches

- Central Word Prediction Objective (context2vec) [Melamud et al. 2016]
- Machine Translation Objective (CoVe) [McMann et al. 2017]
- Bi-directional Language Modeling Objective (ELMo) [Peters et al. 2018]
- Bidirectional Encoder Representations from Transformers (BERT) [Devlin et al. 2018]
- Robustly Optimized BERT (RoBERTa) [Liu et al. 2019]
BERT

- Model: multi-layer self-attention (Transformer)
- Input: a sentence or a pair of sentences with a separator and subword representation
- Why do we need positional embedding?

[figure from Devlin et al. 2018]
Sub-word Tokenization

- BERT uses Word Piece tokenization
- Related models (e.g., for MT, language modeling, etc) use either Word Piece or Byte Pair Encoding tokenization
- Advantage: no unknown words problem
- Package: https://github.com/huggingface/tokenizers
Byte Pair Encoding (BPE) Tokenization

1. Start with every individual byte (basically character) as its own token

2. Count bigram token cooccurrences over words (potentially: weight according to corpus frequencies)

3. Merge the most frequent pair of adjacent characters to create a new token

- Vocabulary size is controlled by the number of merges
- With ~8000 tokens we get many whole words in English

[Sennrich et al. (2016)]
Word Piece Tokenization

1. Initialize with tokens for all characters

2. While vocabulary size is below the target size:
   
   1. Build a language model over the corpus (e.g., unigram language model)
   
   2. Merge pieces that lead to highest improvement in language model perplexity

   • Need to choose a language model that will make the process tractable
   
   • Often a unigram language model (e.g., SentencePiece library)
   
   • Particularly suitable for machine translation

[Schuster and Nakajima (2012), Wu et al. (2016), Kudo and Richardson (2018)]
BERT

- Model: multi-layer self-attention (Transformer)

[figure from Devlin et al. 2018]
• Model: multi-layer self-attention (Transformer)

• BERT Base: 12 layers, 768-dim per word-piece token, 12 heads. Total parameters = 110M

• BERT Large: 24 layers, 1024-dim per word-piece token, 16 heads. Total parameters = 340M

• RoBERTa: much more data (160GB of data instead of 16GB)
Training BERT

- Two objectives: masked language modeling and next sentence prediction
- Data: BookCorpus + English Wikipedia
- Later development with RoBERTa:
  - Much more data
  - Removed the next sentence prediction objective
  - Dynamic masking
Masked Language Modeling

• Similar to predicting the next word for language modeling, but adapted for non-directional self-attention

• The BERT recipe: mask and predict 15% of the tokens
  • For 80% (of 15%) replace with the input token with [MASK]
  • For 10%, replace with a random token
  • For 10%, keep the same
Next Sentence Prediction

• Input: [CLS] Text chunk 1 [SEP] Text chunk 2

• 50% of the time, take the true next chunk of text, 50% of the time take a random other chunk. Predict whether the next chunk is the “true” next
Using BERT

- Use the pre-trained model as the first “layer” of your final model
- Train with fine-tuning using your supervised data
- Fine-tuning recipe: 1-3 epochs, batch size 2-32, learning rate 2e-5 - 5e-5
  - Large changes to weights in top layers (particularly in last layer to route the right information to [CLS])
  - Smaller changes to weights lower down in the transformer
  - Small learning rate and short fine-tuning schedule mean weights don’t change much
- More complex recipes exist
Sentence Classification with BERT

- CLS token is used to provide classification decision

- Example tasks:
  - Sentiment classification
  - Linguistic acceptability
  - Text categorization

[figure from Devlin et al. 2018]
Sentence-pair Classification with BERT

- Feed both sentences, and CLS token used for classification

- Example tasks:
  - Textual entailment
  - Question paraphrase detection
  - Question-answering pair classification
  - Semantic textual similarity
  - Multiple choice question answering

[figure from Devlin et al. 2018]
Tagging with BERT

- Can do for a single sentence or a pair
- Tag each word piece
- Example tasks: span-based question answering, name-entity recognition, POS tagging
Results

- Fine-tuned BERT outperformed previous state of the art on 11 NLP tasks

- Since then was applied to many more tasks with similar results

- The larger models perform better, but even the small BERT performs better than prior methods

- Variants quickly outperformed human performance on several tasks, including span-based question answering — but what does this mean is less clear

- Started an arms race (between industry labs) on bigger and bigger models
Hard to do with BERT

- BERT cannot generate text (at least not in an obvious way)

- Not an autoregressive model, can do weird things like stick a [MASK] at the end of a string, fill in the mask, and repeat

- Masked language models are intended to be used primarily for “analysis” tasks
What does BERT Learn?

- A lot of recent work studying this problem
- Some very interesting results
- But, it’s not completely clear how to interpret them
What does BERT Learn?

• Try to solve different linguistic tasks given each level

• Goal: see what information each new level adds

• Method: try to solve different tasks using mixing weights on levels

• Each task classifier takes a single mixed hidden representation $h_{i,\tau}$ or a pair of representations

\[
i : \text{token index} \\
K : \text{number of levels} \\
\tau : \text{task} \\
\gamma_\tau : \text{task parameter} \\
a_{\tau} : \text{mixing parameters} \\
s_{\tau} = \text{softmax}(a_{\tau}) \\
\]

\[
h_{i,\tau} = \gamma_\tau \sum_{k=0}^{K} s_{\tau}^k h_{i}^k \\
\]

[Tenney et al. (2019)]
What does BERT Learn?

- Each plot shows a task
- Plots show $s^k_T$ weights magnitude in blue, and the number of self-attention levels
- The performance delta when adding this layer is in purple
- Largely: higher level semantic tasks happen in later levels

[figure from Tenney et al. (2019)]
Where to get BERT?

- The Transformers library: https://github.com/huggingface/transformers
- Provides state-of-the-art implementation of many models, including BERT and RoBERTa
- Including pre-trained models
Vision-language Reasoning

- Goal: pre-trained representations for language and vision, where the input is a sentence and image
- Self-attention in BERT allows attending between two sentences
- How can we extend that to a sentence paired with an image?
Vision-language Reasoning

- Solution: pre-process the image to extract bounding boxes around objects
- Now the image is an unordered list of discrete objects
- Objectives: masked language model + masked region modeling + image-text matching

[Li et al. 2019; Tan and Bansal 2019; Chen et al. 2019; and several other simultaneous papers]
Vision-language Reasoning

[figure from Chen et al. 2019]
Results

- Similar trend to what we observe with BERT
- State of the art on 13 vision+language benchmarks
- Similar to BERT, there larger is better