BERT

Masked Language modelling and Pre-training

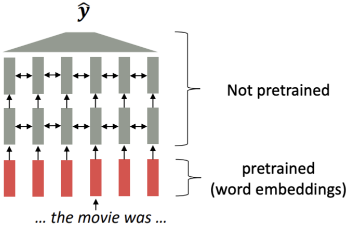
Ryan Chan

07 July 2023

Outline

Feature-based approaches

- Before 2017, the dominant paradigm was:
  1. Start with pretrained word embeddings (e.g. via `word2vec` or `GloVe`) - no context
     - "after stealing money from the bank vault, the bank robber was seen fishing on the Mississippi river bank"
     - "Bank" has the same vector representation no matter how it was used in context
  2. Learn how to incorporate context in an LSTM or Transformer while training on a task
- Using pretrained word embeddings worked much better than using embeddings learnt from scratch [Turian et al., 2010; Devlin et al., 2018]
- Key idea: include pretrained representations as additional features in task-specific architectures

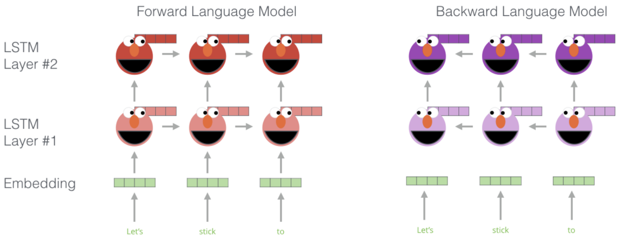


[1]

[1] Manning et al. [2017, Lecture 10]

Contextualised word-embeddings

- Question: how could we obtain an embedding for words based on the context it is used in?
  - Capture *both* the word meaning, as well as other contextual information?
- ELMo [Peters et al., 2018] generalised traditional word embeddings by extracting *context-sensitive* features:
  - Train a forward *and* backwards language model on a large number of text
  - Language modelling type tasks are useful as we can obtain vasts amount of text data that a model can learn from without needing labels



[2]

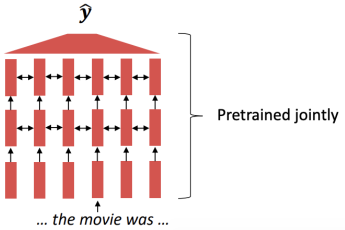
[2] Jay [2018]

Problems with feature-based approaches

- The training data for our downstream task must be sufficient to teach all contextual aspects of language
  - People were typically training large specialised architectures for specific downstream tasks - would require lots of labelled data
- Most of the parameters in our network are randomly initialised
- Although ELMo had bidirectionality, using them was still a feature-based approach to NLP tasks, and not deeply bidirectional: "ELMo used a shallow concatenation of independently trained left-to-right and right-to-left language models" [Devlin et al., 2018]

Fine-tuning approaches

- In "modern" NLP:
  1. All, or almost all, parameters in a network are initialised via pretraining
  2. Pretraining methods (the tasks that they're trained on) hide parts of the input from the model, and train the model to reconstruct those parts
- Key idea: introduce only a minimal amount of task-specific parameters, and train the model to perform downstream tasks by simply fine-tuning *all* pretrained parameters



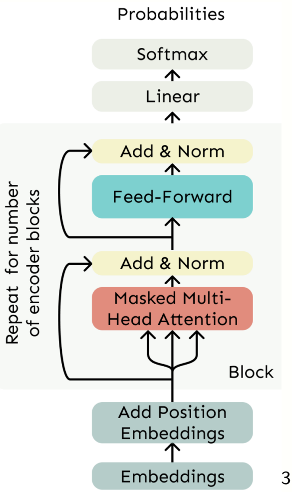
[2]Manning et al. [2017, Lecture 10]

Pretraining transformers

- The transformer encoder-decoder architecture [Vaswani et al., 2017] offered a massive improvement over the previous state-of-the-art LSTM architecture for machine translation
  - Transformers were shown to deal with long-term dependencies much better than LSTMs
  - The encoder-decoder structure of the transformer was perfect for machine translation:
    - The encoder would process the sentence in the source language (bidirectionally)
    - The decoder would process the sentence in the target language
    - We train the entire network end-to-end on the language modelling task on the output target sentence
- Question: how can we pretrain encoder-only and decoder-only models?

Pretraining decoders (GPT)

- OpenAI GPT [Radford et al., 2018] pretrained a decoder model on the standard causal language modelling task: given some tokens, can we predict the next token?
- Radford et al. [2018] obtained (previously) state-of-the-art results on many sentence-level and token-level tasks by pretraining and fine-tuning a unidirectional, left-to-right archiecture:
  - Every token can only attend to previous tokens in the self-attention layers
- Decoders are nice to generate from, but cannot condition on future words

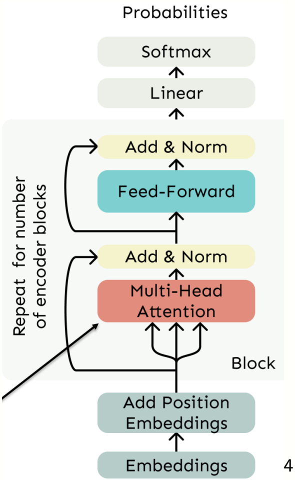


3

3[Manning et al., 2017, Lecture 8]

BERT: Bidirectional Encoder Representations from Transformers

- Devlin et al. [2018] argue that pretraining decoders is sub-optimal: for many sentence-level taks, it is crucial to incorporate context from both directions
- Encoders allow you to build representations using future words
  - Using the transformer architecture will also improve upon the “bidirectional” ELMo approach which used a concatenation of independently trained left-to-right and right-to-left LSTM language models
- Question: how do we pretrain transformer encoder?

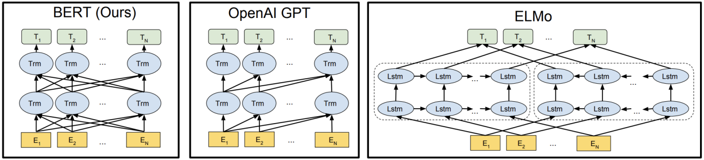


4

[4] [Manning et al., 2017, Lecture 8]

BERT

- There are two steps in the BERT framework:
  1. Pretraining
     - Transformer *encoder* model is trained on unlabelled data
  2. Fine-tuning
     - The BERT model is initialised with the pretrained parameters, and *all* of the parameters are fine-tuned using labelled data from some downstream task
- We still have different (fine-tuned) models for different downstream tasks, but they all share the core BERT architecture and initialised with same pretrained parameters



[5]

[5] Devlin et al. [2018, Figure 3]

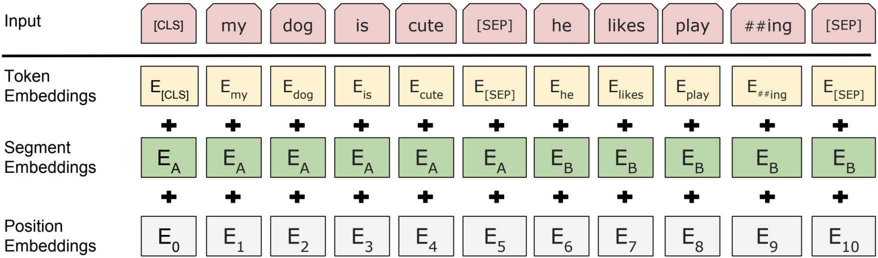
Pretraining BERT

- Unlike ELMo [Peters et al., 2018] and OpenAI GPT [Radford et al., 2018], BERT [Devlin et al., 2018] does not use traditional left-to-right or right-to-left language models to pretrain, but use two unsupervised learning tasks:
  1. Masked Language Modelling (MLM)
  2. Next Sentence Prediction (NSP)

Input representations in BERT

- We want to build an architecture which can be easily fine-tuned to a variety of downstream tasks
  - to excel at both token-level and sentence-level
  - to work with single sentences and pairs of sentences
- To enable this, BERT introduces some new special tokens:
  - The first token of every sequence is always a special classification token: `[CLS]`
    - The final hidden state corresponding to this token is always used as the sequence representation for classification tasks
  - Sentences are differentiated in two ways:
    1. Pairs of sentences are separated with a special separation token: `[SEP]`
    2. Introduce learned embeddings for indicating whether or not a token belongs to sentence `A` or sentence `B`

Input representations in BERT



[6]

[6] Devlin et al. [2018, Figure 2]

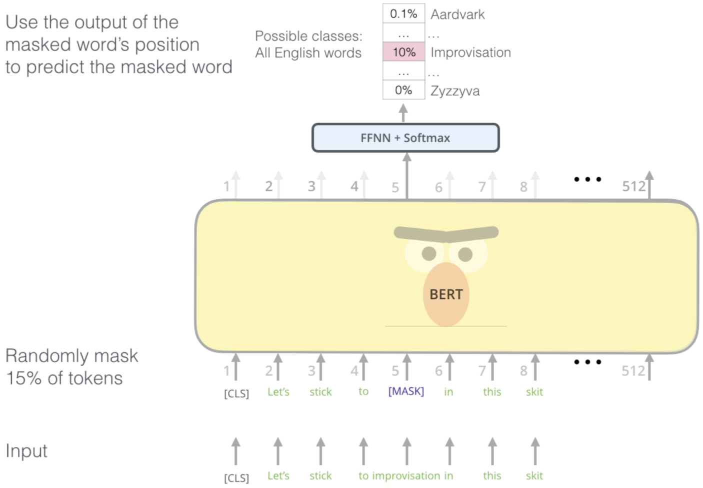
Masked Language Modelling (MLM)

- Cannot use standard causal language modelling task
  - bidirectional conditioning in the encoder would allow each word to see all other tokens in the sequence
  - the model could trivially predict the next target word in a multi-layered network
- Instead, we can pretrain transformer encoders using the masked language modelling task (also known as the Cloze task [Tay, 1953])
  - Simply mask a percentage of tokens at random, and predict those masked tokens
  - The final hidden vectors corresponding to the mask tokens are fed into an output softmax over the vocabulary (like we do in standard language modelling)
- "I went to the store" $\rightarrow$ "I [MASK] to the [MASK]"

MLM in BERT

- Mask 15% of the tokens in the training set at random
  - But we *do not* replace all of these with the special `[MASK]` token
  - Doing this alone creates a problem in that we create a mismatch between pre-training and fine-tuning as '[MASK]' does not appear in fine-tuning
- In BERT:
  - - 15% of tokens are sampled at random for prediction
  - 80% of these tokens are replaced with `[MASK]`
  - 10% are replaced with a random token in the vocabulary
  - 10% are unchanged

MLM in BERT



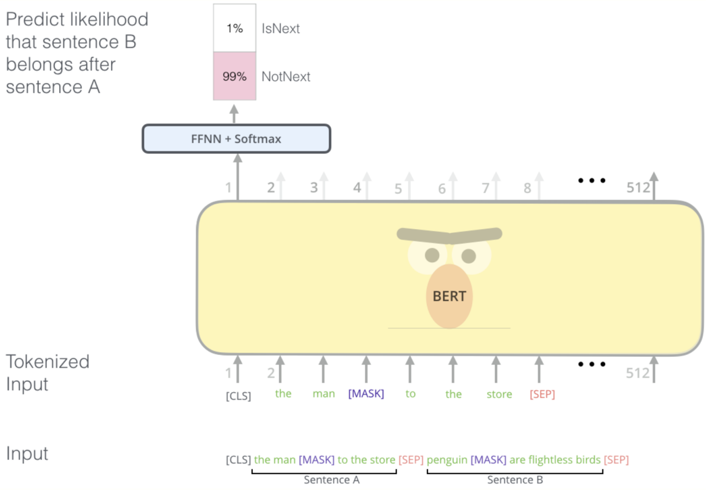
7

[7] Jay [2018]

Next Sentence Prediction (NSP)

- Many downstream tasks, like question answering, are based on understanding the *relationship* between two sentences
- Devlin et al. [2018] argued this is not directly captured by language modelling
- To train BERT to understand sentence relationships, it is also pretrained to the Next Sentence Prediction (NSP) task:
  - When choosing sentences `A` and `B`:
    - 50% of the time, `B` actually is the next sentence that follows `A`: given `IsNext` label
    - 50% of the time, `B` is a random sentence from the corpus: given `NotNext` label
- The final hidden state representation for the `[CLS]` token passed into a two-class output softmax

Next Sentence Prediction (NSP)



BERT architecture and pretraining set up

- Some details on the BERT architecture and data used to pretrain BERT in the paper

Fine-tuning BERT

- Discuss how we can fine-tune BERT to different tasks

Summary of results in BERT

-

Limitations of pretrained encoders

-

References

(1953). Cloze procedure: A new tool for measuring readability, author=Taylor, Wilson L. *Journalism Bulletin*, 30(4):415–433.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*.

Jay, A. (2018). The Illustrated BERT.

Manning, C., Socher, R., Fang, G. G., and Mundra, R. (2017). CS224n: Natural Language Processing with Deep Learning.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). Improving Language Understanding by Generative Pre-Training.

Turian, J., Ratinov, L., and Bengio, Y. (2010). Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual meeting of the Association for Computational Linguistics*, pages 384–394.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention Is All You Need. *Advances in Neural Information Processing Systems*, 30.