Natural Language Understanding, Generation, and Machine Translation

Lecture 10: Pretrained Language Models

Shay Cohen
based on slides by Frank Keller

5 February 2024 (week 4)

School of Informatics
University of Edinburgh
scohen@inf.ed.ac.uk

The Story so Far

Bert Architecture

Masked Training

Pre-training and Finetuning Bert

Reading: [Devlin et al.2019].

The Story so Far

Self-attention

Self-attention is what we get when compute attention over the input sequence. Let $\mathbf{x}_1, \ldots, \mathbf{x}_t$ be the input vectors and $\mathbf{y}_1, \ldots, \mathbf{y}_t$ be the output vectors:

$$\mathbf{y}_i = \sum_j w_{ij} \mathbf{x}_j$$

We now compute the attention weight as the dot-product of each input token with every other token.

$$w'_{ij} = \mathbf{x}_i \cdot \mathbf{x}_j$$
$$\mathbf{w}_i = \text{softmax}(\mathbf{w}'_i)$$

Note that the attention weight is now called $\mathbf{w}'$ and the attention distribution $\mathbf{w}$ (rather than $\mathbf{a}$ and α).

Self-attention

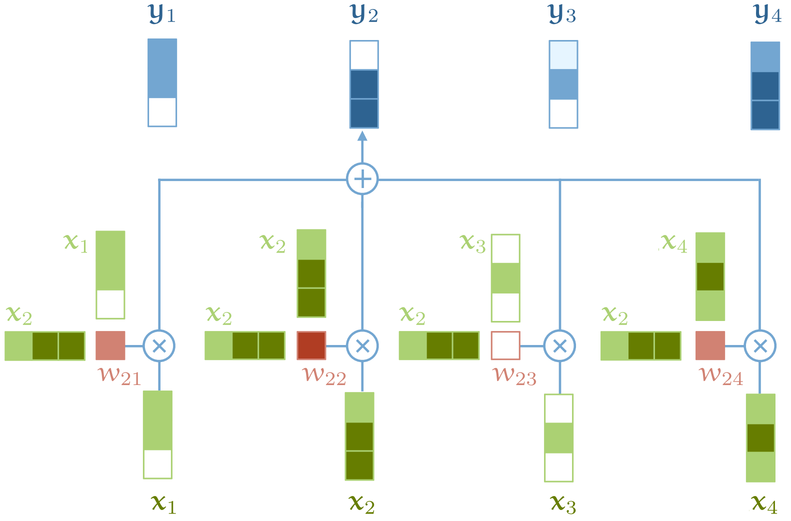
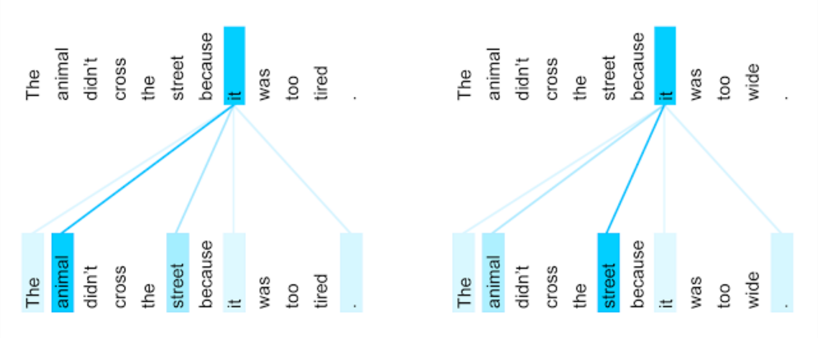


Figure from [Bloem2019].

Multi-head Attention

Multi-head attention is able to jointly attend to different parts of the input. If we just have a single head, have to average.

For example, one head could attend to the subject of a verb, another one to its object. Or different heads could attend to different referents of a pronoun.



The Transformer

The multi-head self attention mechanism needs to be integrated into a larger architecture to be useful:

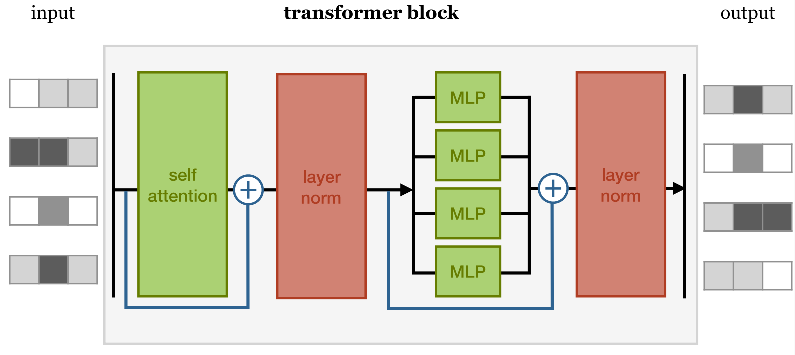


Figure from [Bloem2019].

Position Encodings

We choose a function $f : \mathbb{N} \to \mathbb{R}^k$ that maps positions to real valued vectors, and let the network learn how to interpret these.

The choice of encoding function is a hyperparameter. [Vaswani et al.2017] use:

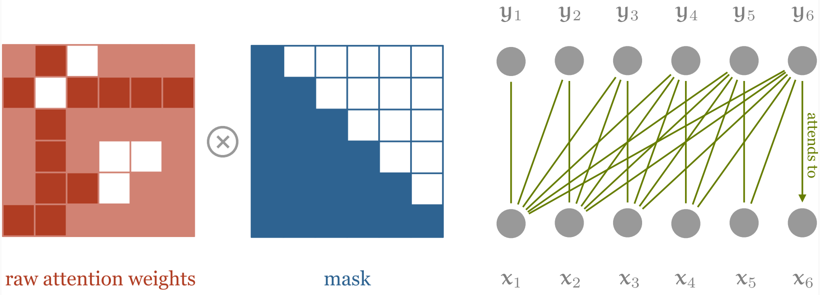
$$PE_{(pos,2i)} = \sin(pos/10000^{2i/k})$$
$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/k})$$

where *pos* is the position and i is an index over dimensions.

The positions encoding is then added to the input embedding (both are of dimensionality k).

Masking

We apply a mask to the matrix of dot products, before the softmax is applied. This disables all elements above the diagonal:



The model can no longer look forward and just copy the upcoming input; it behaves like an RNN!

Figure from [Bloem2019].

From Transformers to Embeddings

- In lecture 8, we saw how we can stack transformers and use them for *classification.* We apply mean pooling to the output and map it onto a class vector.
- We can also use transformers for *sequence prediction.* For this we need to mask some of the input.
- In lecture 9, we saw how we can derive *embeddings* from neural language models.
- Using transformers, we can build language models that represent context very well: *contextualized embeddings.*
- These language models can be used for feature extraction, but also in a *pre-training/finetuning* setup.

Bert Architecture

Bert Architecture

Bert (Bidirectional Encoder Representations from Transformers):

- designed for pre-training deep bidirectional representations from unlabeled text;
- conditions on left and right context in all layers;
- pre-trained model can be finetuned with one additional output layer for many tasks (e.g., NLI, QA, sentiment);
- for many tasks, no modifications to the Bert architecture are required;
- [Devlin et al.2019] report SotA results on 11 tasks using pre-training/finetuning approach.

Bert Architecture

Bert uses bidirectional transformer representations. This is shown to be superior to competing architectures:

- GPT [Radford et al.2018] also uses transformers, but is only trained left-to-right;
- Elmo [Peters et al.2018] uses shallow concatenation of independently trained left-to-right and right-to-left LSTMs.

Main attraction of Bert: *pre-training/finetuning* approach (though this is also present in GPT).

Bert Architecture

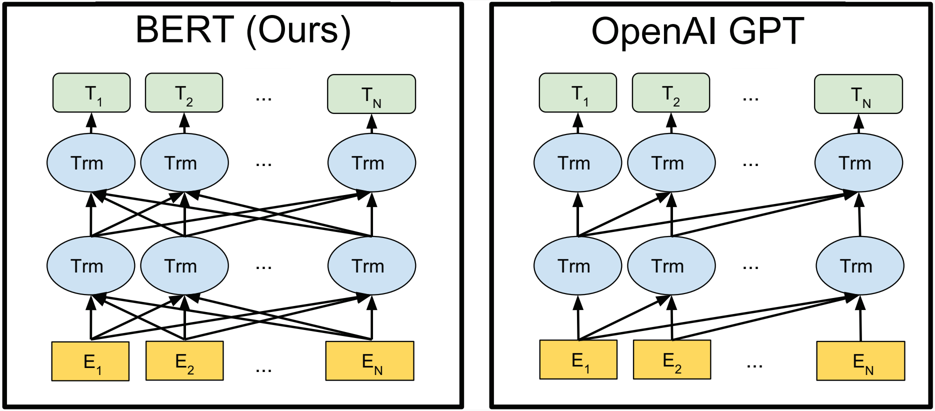


Figure from [Devlin et al.2019].

Bert Architecture

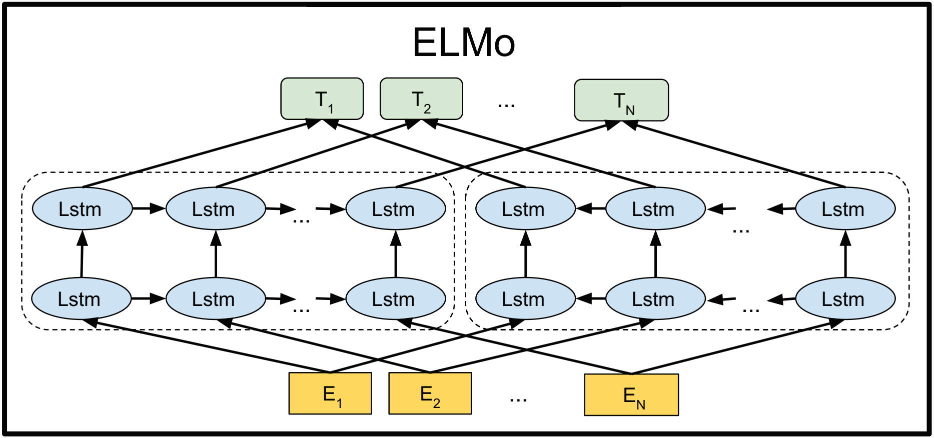


Figure from [Devlin et al.2019].

Bert Architecture

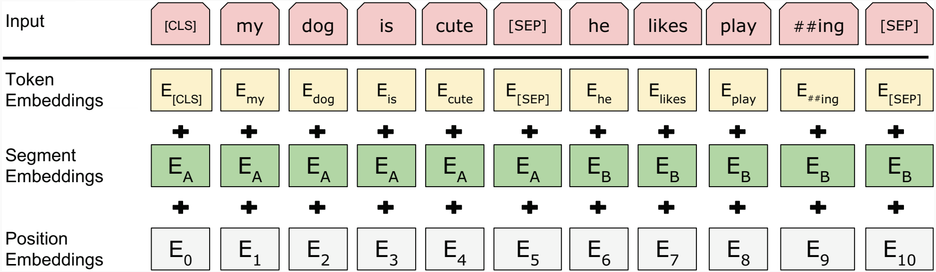
Basic Bert architecture:

- multi-layer bidirectional transformer [Vaswani et al.2017];
- L: layers (transformer blocks); H: dimensionality of hidden layer, A: number of self-attention heads;
- Bert Base: $L = 12$, $H = 768$, $A = 12$, 110M parameters;
- Bert Large: $L = 24$, $H = 1024$, $A = 16$, 340M parameters.

Input/Output Representation

- Input sequence: can be $\langle$Question, Answer$\rangle$ pair, single sentence, or any other string of tokens;
- 30,000 token vocabulary, represented as WordPiece embeddings (handles OOV words);
- first token is always `[CLS]`: aggregate sentence representation for classification tasks;
- sentence pairs separated by `[SEP]` token; and by segment embeddings;
- token position represented by position embeddings.

Bert Architecture



E: input embedding, C: final hidden vector of `[CLS]`; T_i: final hidden vector for the i-th input token.

Note: C and T_i used in subsequent slides.

Figure from [Devlin et al.2019].

Masked Training

Masked Language Model

Important departure from previous embedding models (including GPT and Elmo):

Don't train the model to predict the next word, but train it to predict the whole context.

- Problem: how can we prevent *trivial copying* via the self-attention mechanism?
- Solution: mask 15% of the tokens in the input sequence; train the model to predict these.

Masked Language Model

Problem: masking creates mismatch between pre-training and finetuning: `[MASK]` token is not seen during fine-tuning. Solution:

- do not always replace masked words with `[MASK]`, instead choose 15% of token positions at random for prediction;
- if i-th token is chosen, we replace the i-th token with:
  1. the `[MASK]` token 80% of the time;
  2. a random token 10% of the time;
  3. the unchanged i-th token 10% of the time.
- Now use T_i to predict original token with cross entropy loss.

Masked Language Model

Why this masking scheme?

- if we always use [MASK] token, the model would not have to learn good representation for other words;
- if we only use [MASK] token or random word, model would learn that observed word is never correct;
- if we only use [MASK] token or observed word, model would just learn to trivially copy.

Secondary Task: Next Sentence Prediction

- Bert is designed to be used for tasks such a question answering (QA) and natural language inference (NLI) that require sentence pairs;
- Bert uses a special mechanism to capture this: pre-training on next sentence prediction task;
- generate training data: chose two sentences A and B, such that 50% of the time B is the actual next sentence of A, and 50% of the time a randomly selected sentence.

Pre-training and Finetuning Bert

Pre-training and Finetuning Bert

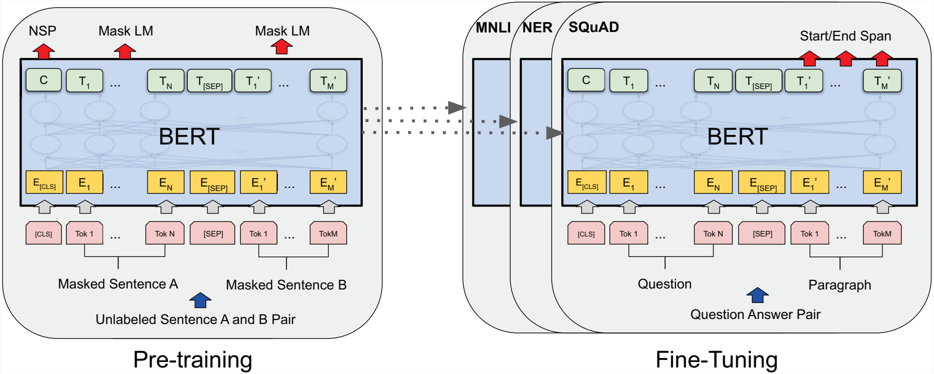


Figure from [Devlin et al.2019].

Pre-training and Finetuning Bert

- Pre-training on BooksCorpus (800M words) and English Wikipedia (2,500M words);
- to finetune for a new task, we plug task-specific inputs and outputs into Bert and re-train its parameters;
- input: designed to be two sentences:
  1. sentence pairs in paraphrasing;
  2. hypothesis-premise pairs in entailment;
  3. question-passage pairs in question answering;
  4. text-$\emptyset$ pair in text classification or sequence tagging;
- output:
  1. sequence of tokens in tasks such as QA (mark answer span);
  2. sequence of labels in tagging tasks such as NER;
  3. `[CLS]` representation is fed into an output layer for classification, such as entailment or sentiment.
- appendix of [Devlin et al.2019] gives examples.

Bert Results: Glue Benchmark

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Glue is a mixture of various natural language understanding tasks: MNLI, QNLI, WNLI: natural language inference; QQP: question equivalence; SST-2: sentiment; CoLA: linguistic acceptability; STS-B: semantic similarity; MRPC: paraphrasing; RTE: entailment.

Feature Extraction vs. Finetuning: Named Entity Recognition

System	Dev F1	Test F1
ELMo (Peters et al., 2018a)	95.7	92.2
CVT (Clark et al., 2018)	-	92.6
CSE (Akbik et al., 2018)	-	**93.1**
Fine-tuning approach		
$\text{BERT}_{\text{LARGE}}$	96.6	92.8
$\text{BERT}_{\text{BASE}}$	96.4	92.4
Feature-based approach ($\text{BERT}_{\text{BASE}}$)		
Embeddings	91.0	-
Second-to-Last Hidden	95.6	-
Last Hidden	94.9	-
Weighted Sum Last Four Hidden	95.9	-
Concat Last Four Hidden	96.1	-
Weighted Sum All 12 Layers	95.5	-

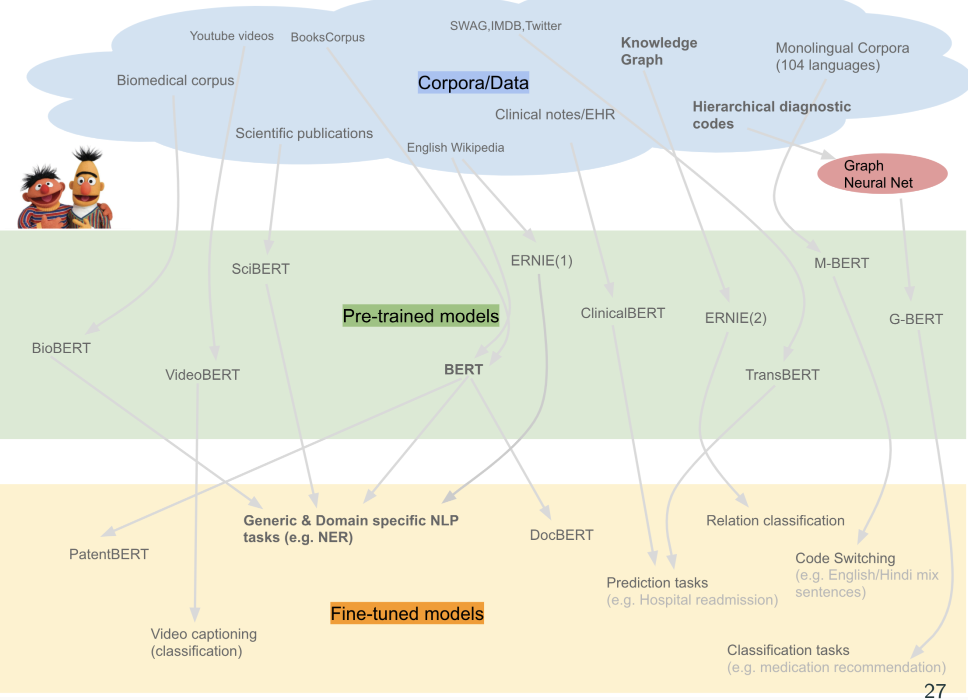
Bert and the State of the Art

The current state of the art in NLP owes a lot to Bert (and many subsequent large language models):

- pre-training a large language model and then fine-tuning or prompting is state of the art for many NLP tasks
- pre-training requires lots of resources! Estimate for Bert (Tim Dettmers): 4 GPUs (RTX 2080Ti) for 100 days
- fine-tuning existing model is relatively quick, but fitting model in GPU memory can be a challenge
- getting new state of the art results with ever-larger models and data is a game that will continue, but only few can play
- positive (for everybody else): we also need smart ideas for architectures, objective functions, evaluation, etc.

Summary

- Bert is a contextualized language model that uses a deep, bidirectional transformer architecture;
- it is pre-trained on unlabeled text using masking and next sentence prediction;
- it is designed for finetuning with minimal architectural modifications;
- the input uses sentence pairs; the output can be sentences, labels, classification decisions, depending on task;
- Bert-based models are state of the art on many NLP tasks.
- There are many variants of Bert ...


Corpora/Data
Youtube videos
BooksCorpus
SWAG,IMDB,Twitter
Knowledge Graph
Monolingual Corpora (104 languages)
Biomedical corpus
Clinical notes/EHR
Hierarchical diagnostic codes
Scientific publications
English Wikipedia
Graph Neural Net
Pre-trained models
SciBERT
ERNIE(1)
M-BERT
ClinicalBERT
ERNIE(2)
G-BERT
BioBERT
VideoBERT
BERT
TransBERT
Fine-tuned models
Generic & Domain specific NLP tasks (e.g. NER)
DocBERT
Relation classification
PatentBERT
Code Switching
(e.g. English/Hindi mix sentences)
Prediction tasks
(e.g. Hospital readmission)
Video captioning (classification)
Classification tasks
(e.g. medication recommendation)

References

Peter Bloem.
2019.
Transformers from scratch.
Blog, http://www.peterbloem.nl/blog/transformers.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova.
2019.
BERT: Pre-training of deep bidirectional transformers for language understanding.
In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 4171–4186, Minneapolis, MN.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer.
2018.
Deep contextualized word representations.
In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 2227–2237, New Orleans, LA.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever.
2018.
Improving language understanding with unsupervised learning.
Technical report, OpenAI.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin.
2017.
Attention is all you need.
In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008, Red Hook, NY. Curran Associates.