Foundations of Natural Language Processing

Lecture 18: Transfer Learning I (BERT)

Mirella Lapata
School of Informatics
University of Edinburgh
mlap@inf.ed.ac.uk

Recap

- We have learned about the Transformer model!
- Attention is the key component (residual connections, feed-forward layers, position embeddings, and layer normalization).
- **This lecture:** how do you train Transformer models, what can you do with them?

What is Transfer Learning?

General Definition

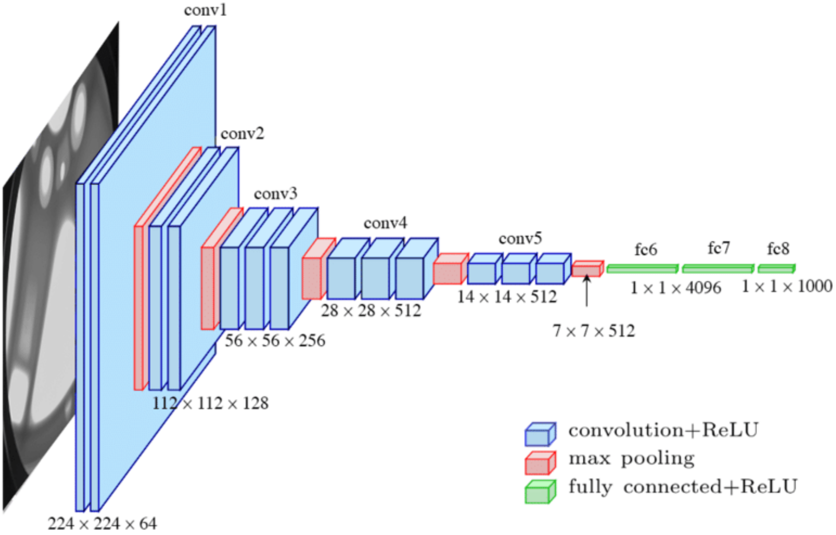
Transfer learning is a machine learning technique where a model trained on one task is re-purposed on a second related task.

Deep Learning Specific Definition

In transfer learning, we first train a base network on a base dataset and task, and then we repurpose the learned features, or transfer them, to a second target network to be trained on a target dataset and task. This process will tend to work if the features are general, meaning suitable to both base and target tasks, instead of specific to the base task.

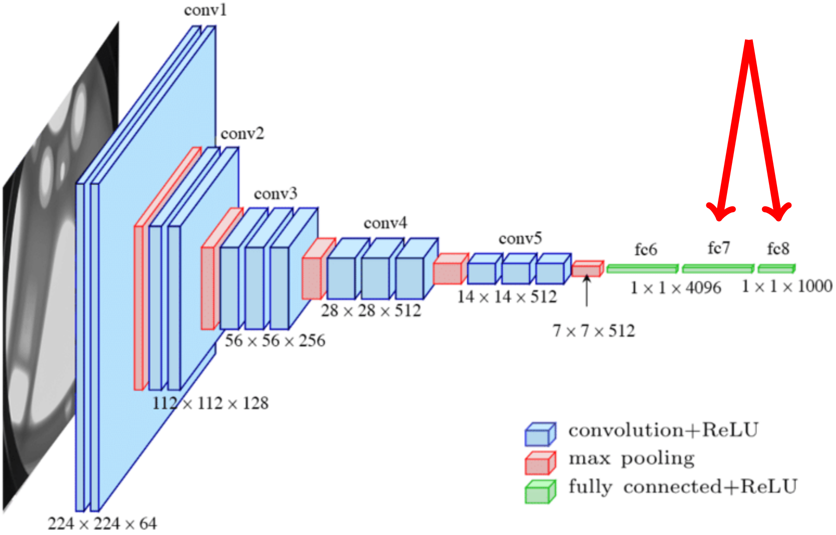
VGG-16: A Typical Object Detection Model

(Simonyan and Zisserman, 2015)



VGG-16: A Typical Object Detection Model

(Simonyan and Zisserman, 2015)



Feature Extraction

- A model like VGG-16 has a lot of layers (pretty dated!).
- Current object detection models can have hundreds of layers.
- They take forever to train and require very large training sets.
- However, computer vision researchers found that you can take the output of the last layer as a feature vector. (In this example, you would use fc7, and fc8 is just the softmax.)
- For a new task, use VGG-16 to turn an image into features, and then train a classifier for a new task on these features.

Feature Extraction

- A model like VGG-16 has a lot of layers (pretty dated!).
- Current object detection models can have hundreds of layers.
- They take forever to train and require very large training sets.
- However, computer vision researchers found that you can take the output of the last layer as a feature vector. (In this example, you would use fc7, and fc8 is just the softmax.)
- For a new task, use VGG-16 to turn an image into features, and then train a classifier for a new task on these features.
- **Transfer learning by feature extraction!**

Pre-training and Finetuning

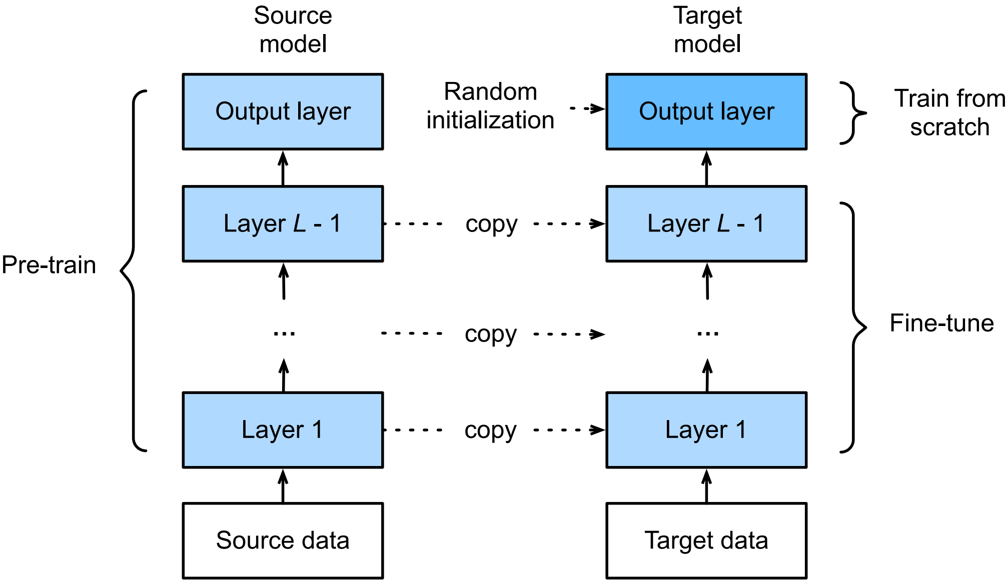
Instead of just extracting features, we can retrain the model for a new task using new data:

- *Pre-training:* train a generic **source model** (e.g., VGG-16) on a standard, large dataset (e.g., ImageNet).
- *Finetuning:* then take the resulting model, keep its parameters, and replace the output layer to suit the new task. Now train this **target model** on the dataset for the new task.

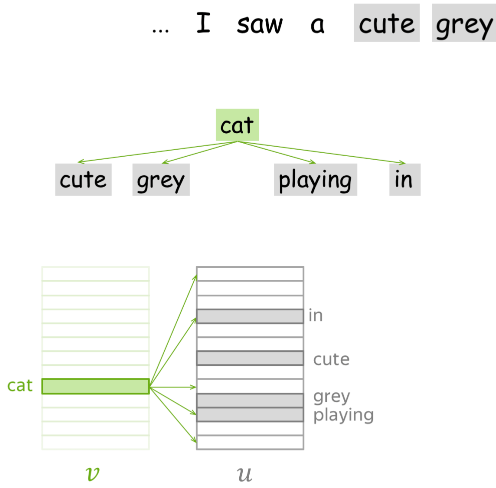
Transfer learning by finetuning.

You can think of pre-training as a way of *initializing the parameters* of your target model to good values.

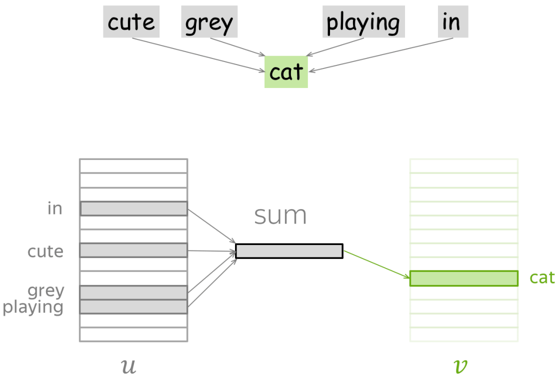
Pre-training and Finetuning



Pre-training and Finetuning in NLP: Remember word2vec?



<u>Skip-Gram</u>: from central predict context (one at a time)



<u>CBOW</u>: from sum of context predict central

Pre-training and Finetuning in NLP: Remember word2vec?

- When we use word embeddings (e.g., skip-gram) as the input representations for a new task, then we're doing feature extraction.
- Our **source model** is the neural language model that the embeddings come from.
- The **target model** is often very different from the NLM, as our target task rarely is next word prediction.
- However, we can allow the weights of the embedding layer of the target model to be trained.
- This is a limited form of finetuning.

Pre-training and Finetuning in NLP: Remember word2vec?

- When we use word embeddings (e.g., skip-gram) as the input representations for a new task, then we're doing feature extraction.
- Our **source model** is the neural language model that the embeddings come from.
- The **target model** is often very different from the NLM, as our target task rarely is next word prediction.
- However, we can allow the weights of the embedding layer of the target model to be trained.
- This is a limited form of finetuning.

However, can we do full-scale *finetuning for NLP?* This became possible with *contextualized word embeddings.*

Static Word Embeddings

(1) The new-look **play** area is due to be completed by early spring 2010.
(2) Congressional districts favor representatives who **play** to the party base.
(3) The freshman then completed the three-point **play** for a 66-63 lead.

- Word2vec represents each word as a single vector, *independent of its context*.
- Word2vec embeddings are used for feature extraction, i.e., to initialize the embedding layer of a target model.
- They are not designed to be used for finetuning.
- They are often very efficient, so that training from scratch is possible.

Contextualized (or Dynamic) Word Embeddings

Often called *pretrained language models* or *large language models.*

- Assign a vector to a word that depends on its context, i.e., on the preceding and following words.
- They can be finetuned: we re-train some of the weights of the embedding model for the target task.
- Contextualized embeddings take a lot of memory and compute to train from scratch.
- But finetuning is an efficient way of using them.
- BERT, GPT are typical examples. Pre-trained models available for many languages and applications.

In this lecture: We introduce *BERT* as an example of contextualized word embeddings.

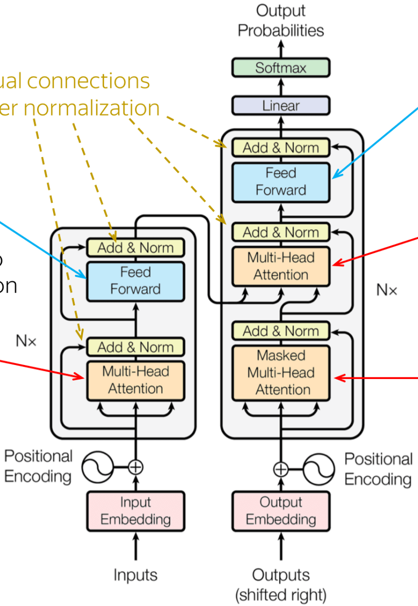
2018

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

Transformer Recap

Residual connections
and layer normalization



Feed-forward network:
after taking information from
other tokens, take a moment to
think and process this information

Encoder self-attention:
tokens look at each other

queries, keys, values
are computed from
encoder states

Feed-forward network:
after taking information from
other tokens, take a moment to
think and process this information

Decoder-encoder attention:
target token looks at the source

queries – from decoder states; keys
and values from encoder states

Decoder self-attention (masked):
tokens look at the previous tokens

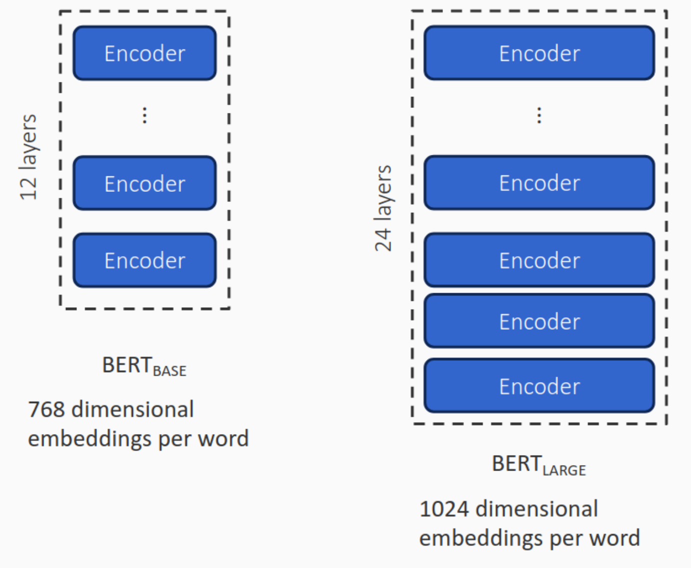
queries, keys, values are computed
from decoder states

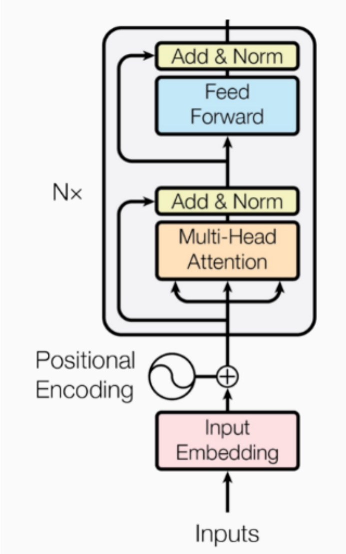
BERT Architecture

BERT (**B**idirectional **E**ncoder **R**epresentations from **T**ransformers):

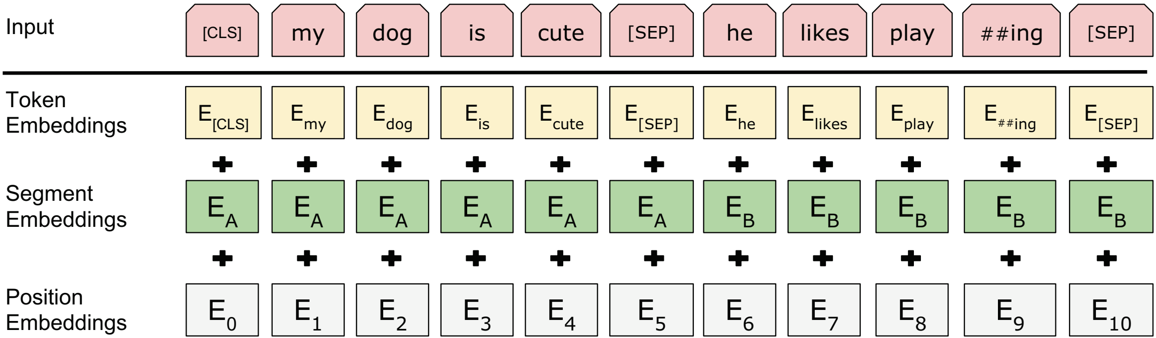
- designed for pre-training deep bidirectional representations from unlabeled text;
- conditions on left and right context in all layers;
- pre-trained model can be finetuned with one additional output layer for many tasks (e.g., NLI, QA, sentiment);
- for many tasks, no modifications to BERT architecture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

BERT: A stack of transformer encoders





BERT: Input Representation

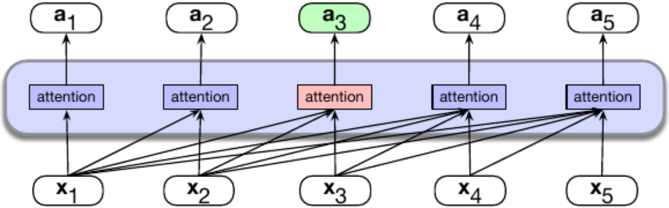


Each token is the sum of three embeddings

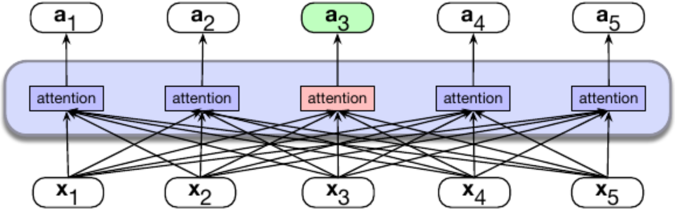
Addition to transformer encoder: sentence embeddings

Figure from Devlin *et al.* (2019).

Bidirectional Transformers



a) A causal self-attention layer



b) A bidirectional self-attention layer

BERT: Architecture

Multi-layer bidirectional transformer (Vaswani *et al.*, 2023)

	L layers (transformer blocks)	H dimensionality of hidden layer	A: number of self-attention heads
BERT Base	$L = 12$	$H = 768$	$A = 12$
BERT Large	$L = 24$	$H = 1024$	$A = 16$

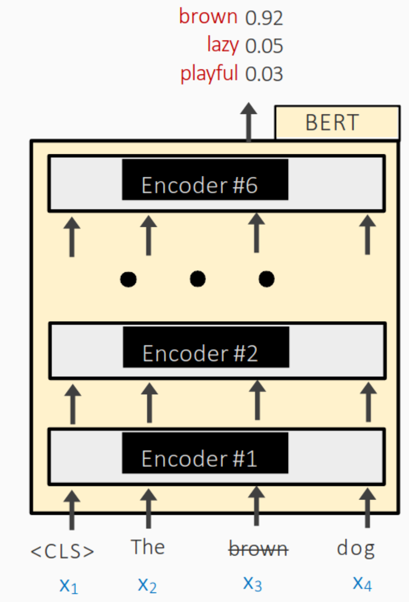
- **Input sequence:** ⟨Question, Answer⟩ pair, single sentence, or any token sequence;
- 30,000 token vocabulary, represented as WordPiece embeddings (OOV words);
- **first token is `[CLS]`:** aggregate sentence representation for classification tasks;
- sentence pairs **separated by `[SEP]` token**; and by **segment embeddings**;
- token position represented by **position embeddings.**

How is BERT pre-trained?

BERT provides embeddings for each token in its input.

- *The goal*: after pre-training, the words should have “good” contextualized embeddings for any task in the future.
- BERT uses two surrogate tasks (and corresponding objectives) for pre-training:
  (1) Masked language modeling
  (2) Next sentence prediction

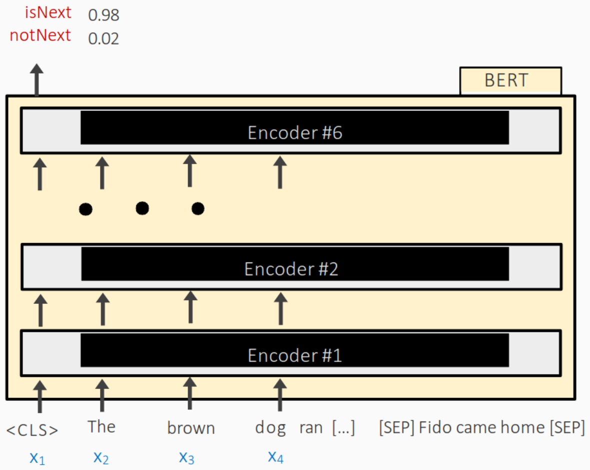
The masked language modeling objective



- Predict the Masked word (a la CBOW).
- `[MASK]` 15% of all input words are randomly masked. if i-th token is chosen, replace with:
  - a) `[MASK]` 80% of the time;
  - b) itself 10% of the time (no change)
  - c) a random token 10% of the time (corrupt it deliberately)

Can you think why this specific masking strategy was adopted?

Next sentence prediction objective



- Two sentences are fed in at a time.
- Predict the if the second sentence follows the first one or not.
- 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.

What does this objective achieve?

Pre-training and Finetuning BERT

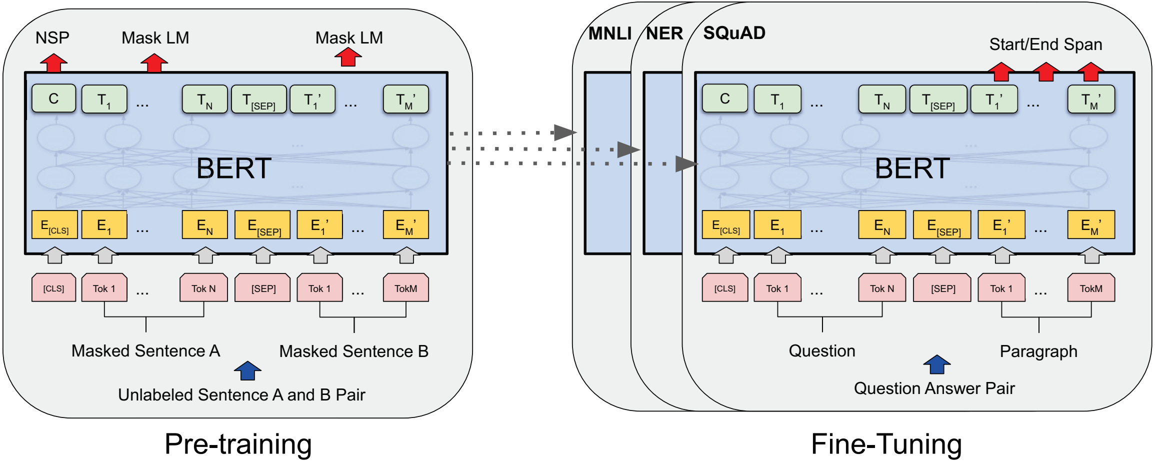


Figure from Devlin *et al.* (2019).

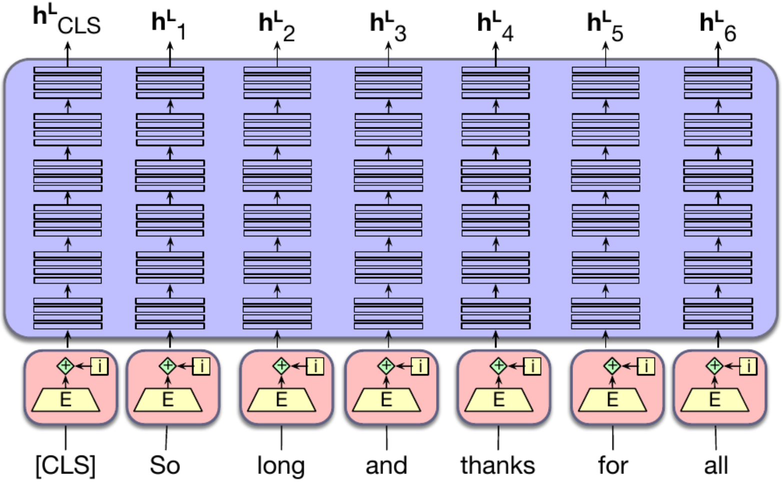
Pre-training and Finetuning BERT

Training details

- The model is trained on unlabeled data (self-supervised learning)
- **Data:** Wikipedia (2.5B words) + BookCorpus (800M words)
- **Training Time:** 1M steps (40 epochs), 4 TPU days
- **Optimizer:** AdamW, 1e-4 learning rate, linear decay layer

- 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 (see appendix of Devlin *et al.* (2019) for examples):
  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.

Contextual Embeddings



Whereas word2vec had a single vector for each word type, contextual embeddings provide a single vector for each instance of that word type in its sentential context.

The senses of the word “die”

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		
BERT$_{\text{LARGE}}$	96.6	92.8
BERT$_{\text{BASE}}$	96.4	92.4
Feature-based approach (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	-

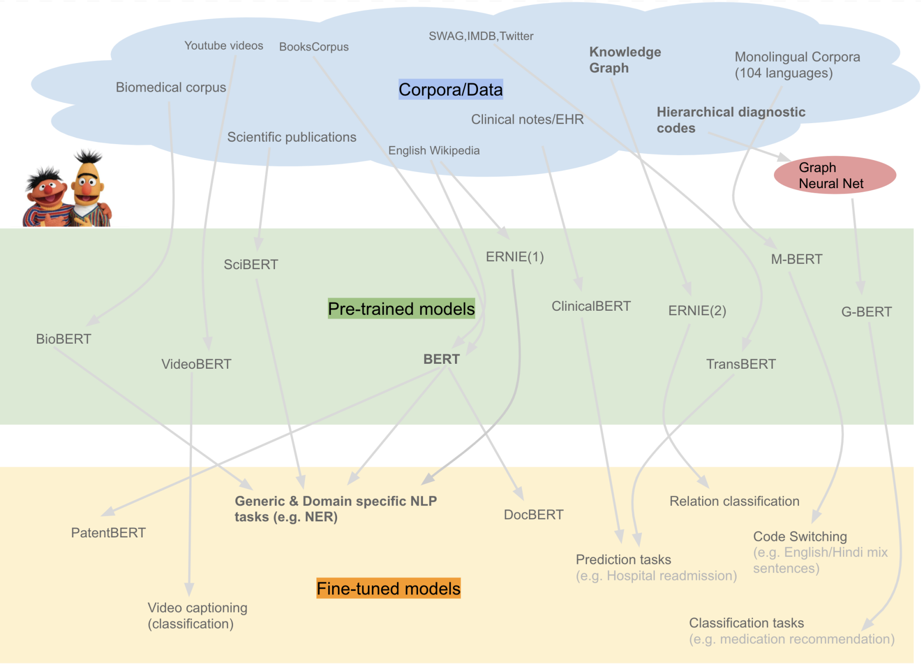
BERT and the State of the Art

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 68 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

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, and T. Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2023). Attention is all you need.