Natural Language and Speech Processing

Lecture 9: Contextual Word Embeddings.
BERT, BART, GPT, and others.

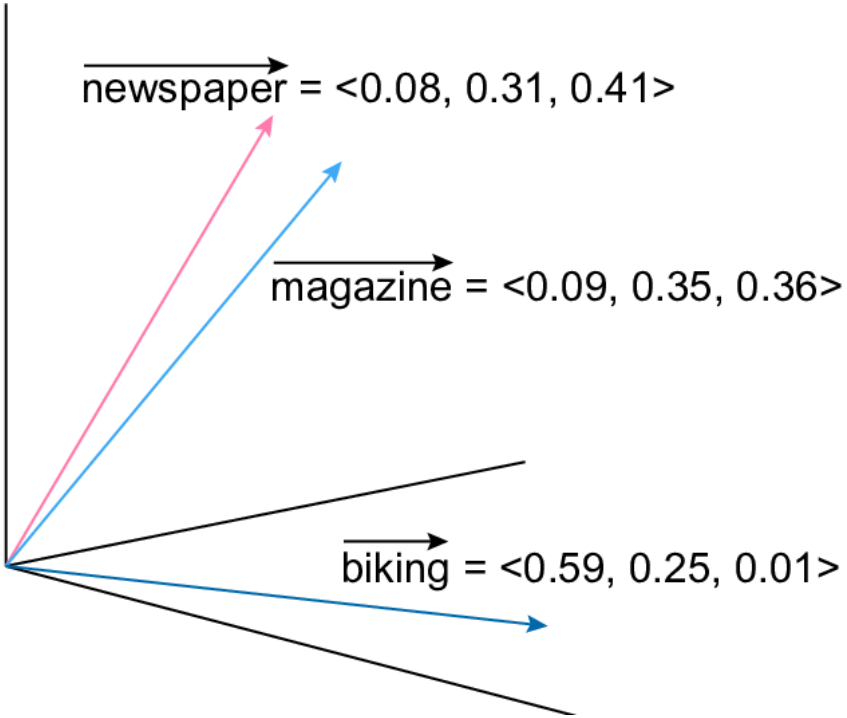
Tanel Alumäe

Contents

- Contextual word embeddings
  - ELMo
  - ULM-FiT
- Transformer model
- BERT
- GPT

Static word embeddings recap

- Word embeddings are really valuable for processing natural language with DNNs
- Word embeddings capture both semantic and syntactic information
  - Stockholm” and “Sweden” have the same relationship between them as “Cairo” and “Egypt” have between them
  - the relationship between “had” and “has” is the same as that between “was” and “is”
- Embeddings can **pre-trained** on vast amounts of text data instead of training them alongside the model on a small labeled dataset (e.g., for doing NER)



What about words with many meanings?

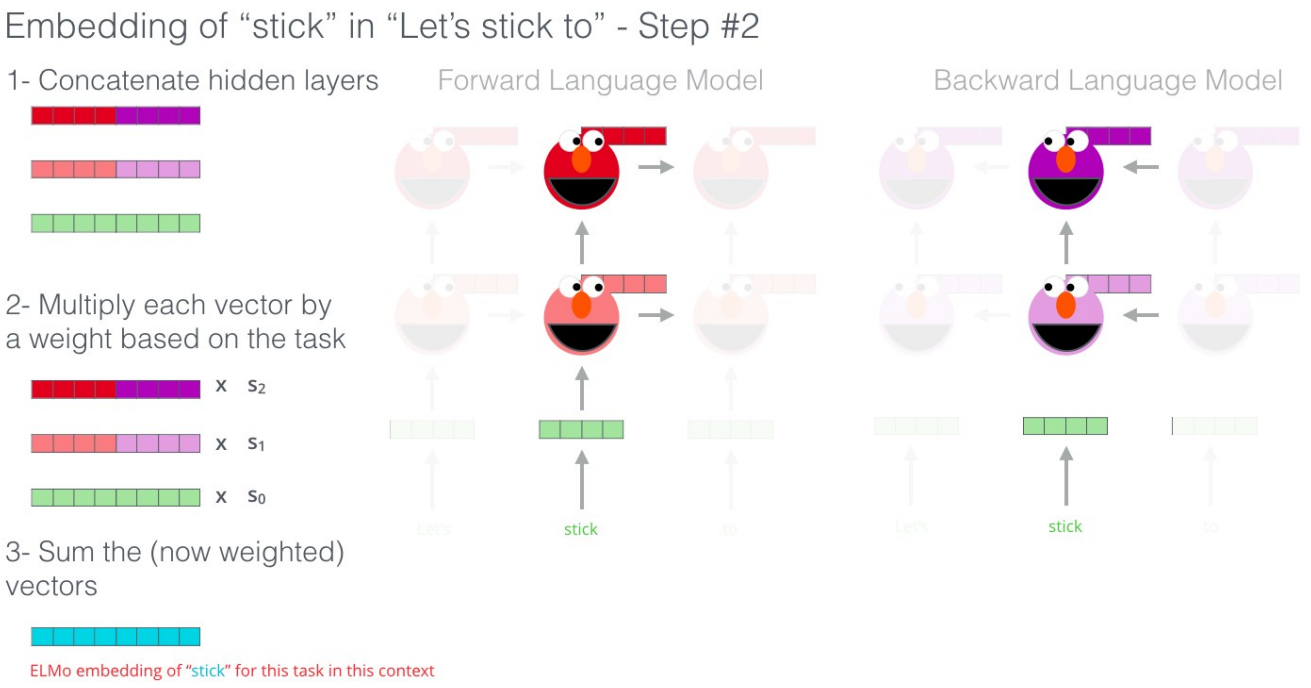
- Word embeddings map each word to a **single** static vector
- What about words with several senses?
  - bank, stick, odd, etc.
- There were some early attempts to discover words with several senses, **disambiguate** words in the training text, and train a different embedding for each sense
- E.g:
  - Preprocess:
    - „Let's **stick** to this idea" -> „Let's **stick**$_2$ to this idea"
  - After training:
    - `stick`$_1$ `-> [0.5, -1.7,  0.8,  4.9]`
    - `stick`$_2$ `-> [2.1, -0.7,  0.1, -0.9]`
- However, this is cumbersome, not very flexible and doesn't work well

ELMo: Context Matters

- Why not give it an embedding *dynamically* based on the context it's used in?
  - to both capture the word meaning in that context as well as other contextual information
- And so, contextualized word-embeddings were born
- ELMo (Peters et. al., 2018):
  - Instead of using a fixed embedding for each word, ELMo looks at the entire sentence before assigning each word in it an embedding
  - It uses a **bi-directional LSTM** to create those embeddings

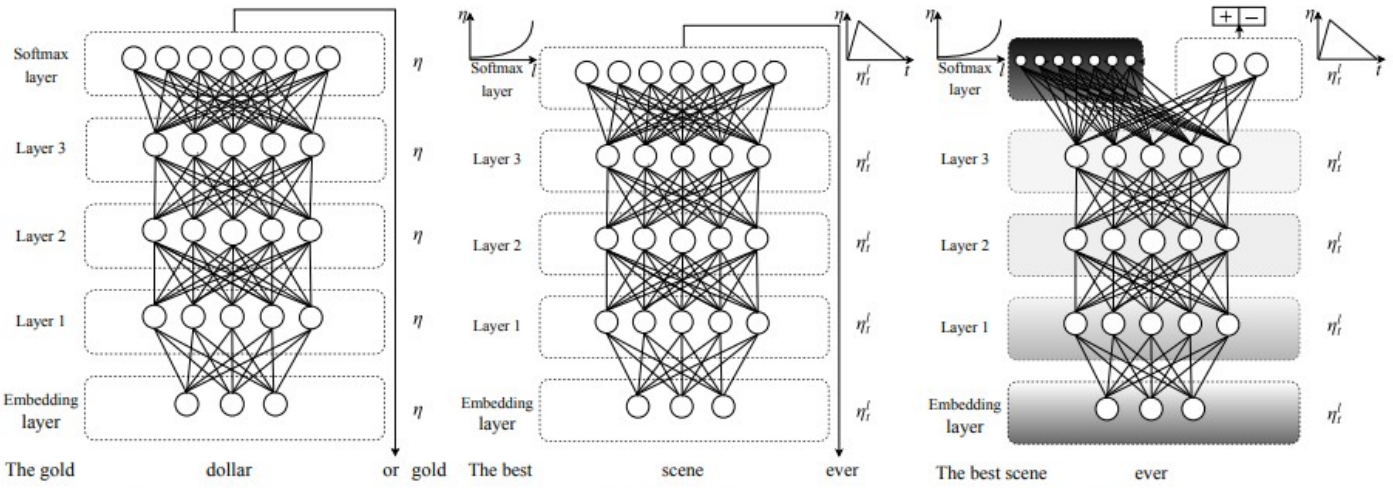
ELMo, continued

- ELMo gains its language understanding from being trained to predict the next word in a sequence of words (language modeling)
- Actually, ELMo has two LMs: forward and backward
- ELMo creates contextualized embedding through grouping together the hidden states (and initial embedding) in a certain way (concatenation followed by weighted summation)



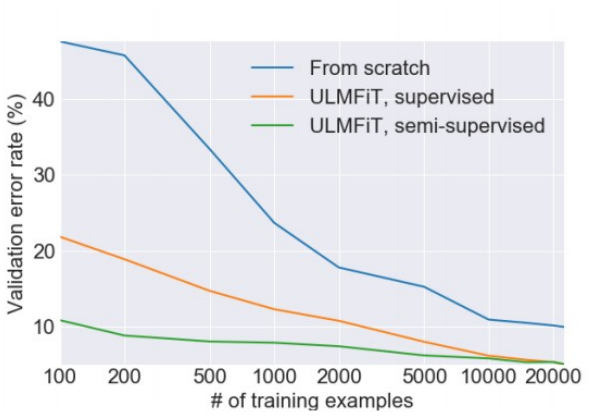
ULM-FiT

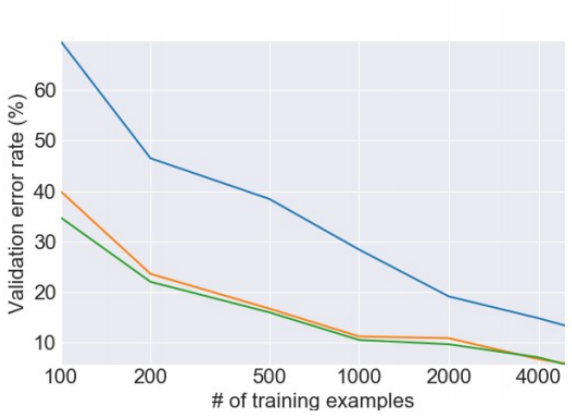
- Universal Language Model Fine-tuning (ULM-FiT, Howard & Ruder, 2018)
- Pretrained LSTM language model with many regularization tricks
- Procedure to finetune the LM on in-domain data and train the final classifier using in the fine-tuned LM

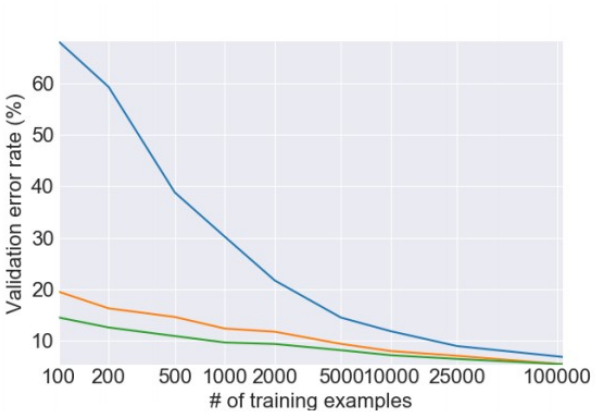


ULM-FiT

- ULM-FiT improves results a lot on typical NLP tasks (compared to training from scratch on in-domain data), especially if there is little in-domain labeled data for the task
- Below: for three different tasks:
  - **Training from scratch**
  - **Using a pre-trained ULM-FiT model „as is”**
  - **First fine-tuning a pre-trained ULM-FiT model on unlabeled in-domain data (using LM objective), and then training a classifier on labeled data**



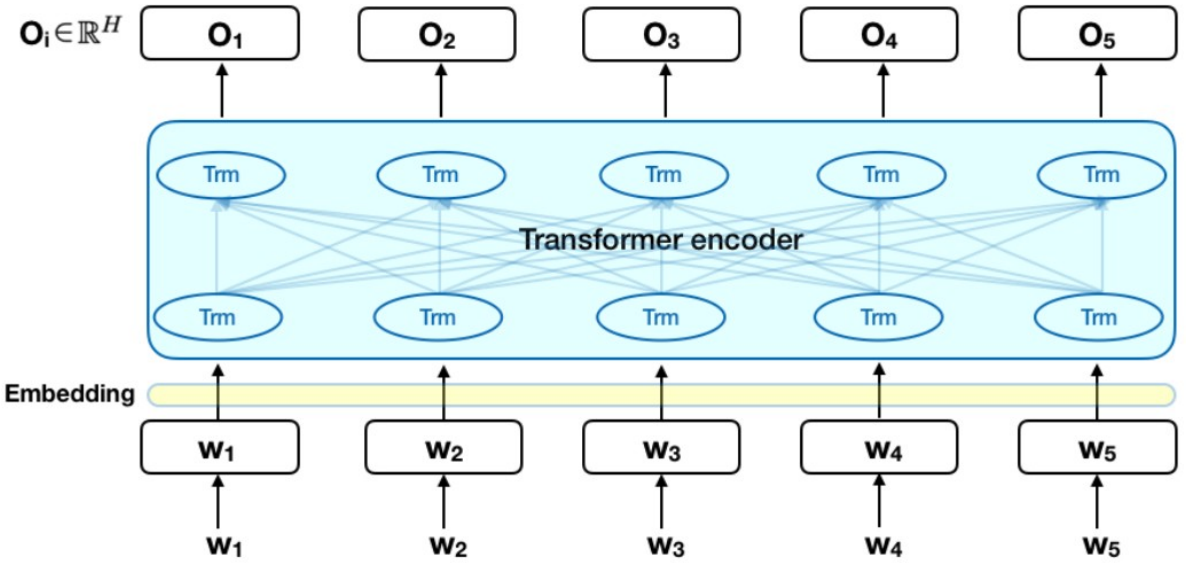




BERT

BERT

- BERT (Bidirectional Encoder Representations from Transformers, introduced in 2018) is a model that broke several records in various language-based tasks
- BERT is a „huge” pretrained model for English (a multilingual version mBERT is also available)
- BERT can be used to generate contextual word embeddings
- Or, to compute something based on a pair of texts

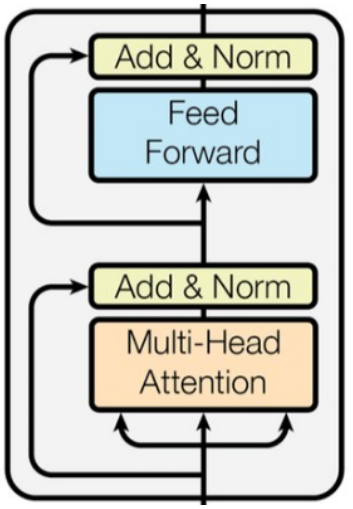


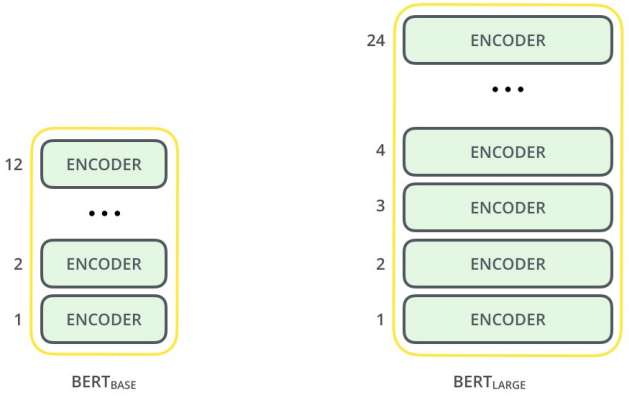
BERT applications

- Text classification
- Word classification
- Natural language understanding
  - Given: paragraph, question
  - Output: answer to the question based on the paragraph, or no answer if not available
- Almost any NLP task (in English) that doesn’t require text generation

BERT architecture

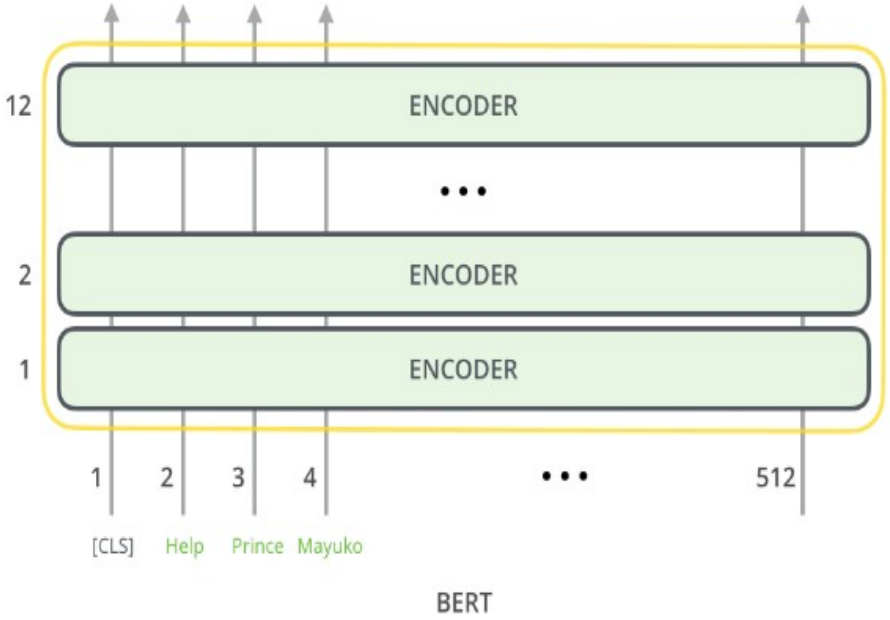
- BERT is basically a trained **Transformer Encoder** stack
- **BERT-base**: 12 layers, 768-unit hidden layers
- **BERT-large**: 20 layers, 1024-unit hidden layers
- Each layer is a self-attention layer



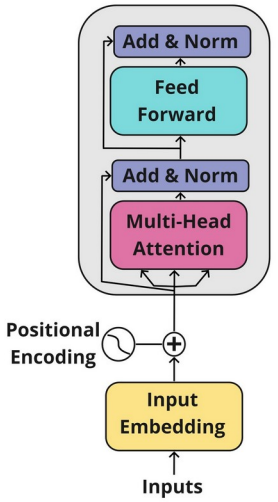


BERT model inputs

- The first input token is supplied with a special [CLS] token
- BERT takes a sequence of words as input which keep flowing up the stack
- Each layer applies self-attention, and passes its results through a feed-forward network, and then hands it off to the next encoder

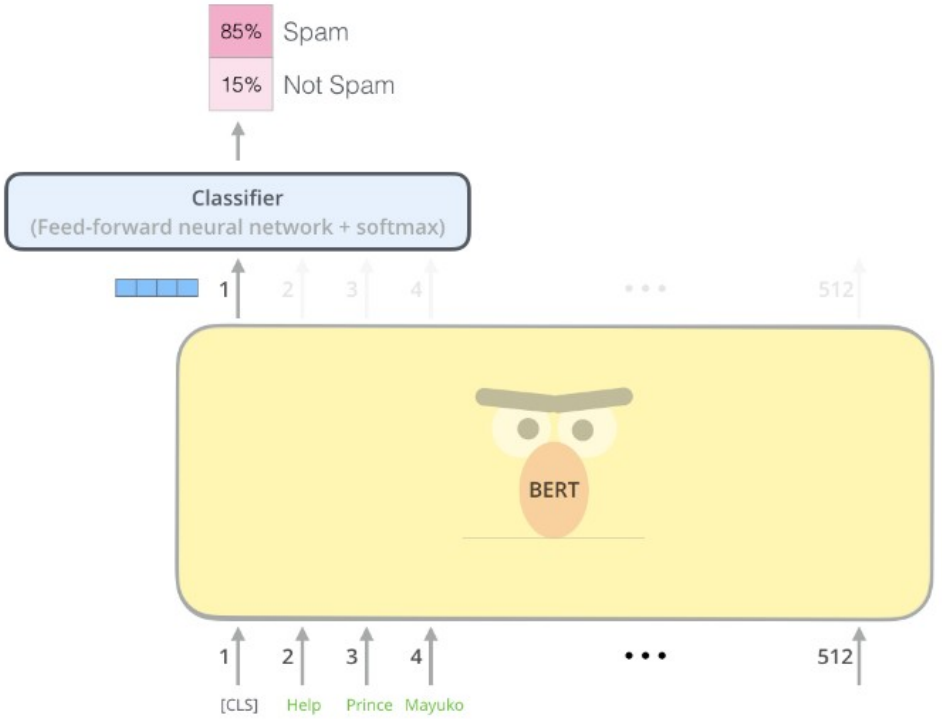


BERT



BERT model outputs

- Each position outputs a vector of size *hidden_size* (768 in BERT Base)
- For the sentence classification example we've looked at above, we focus on the output of only the first position (that we passed the special [CLS] token to)

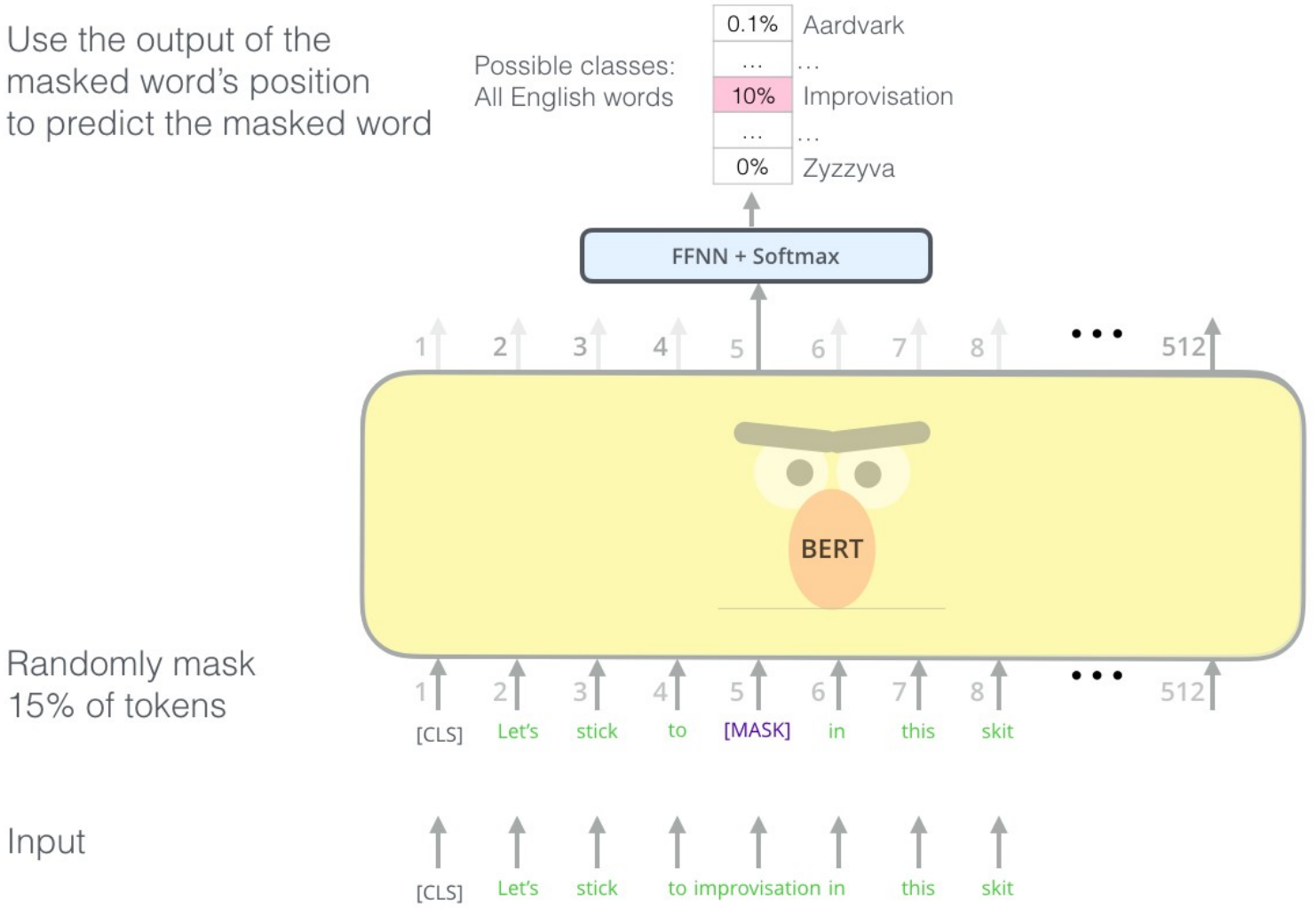


How BERT is trained

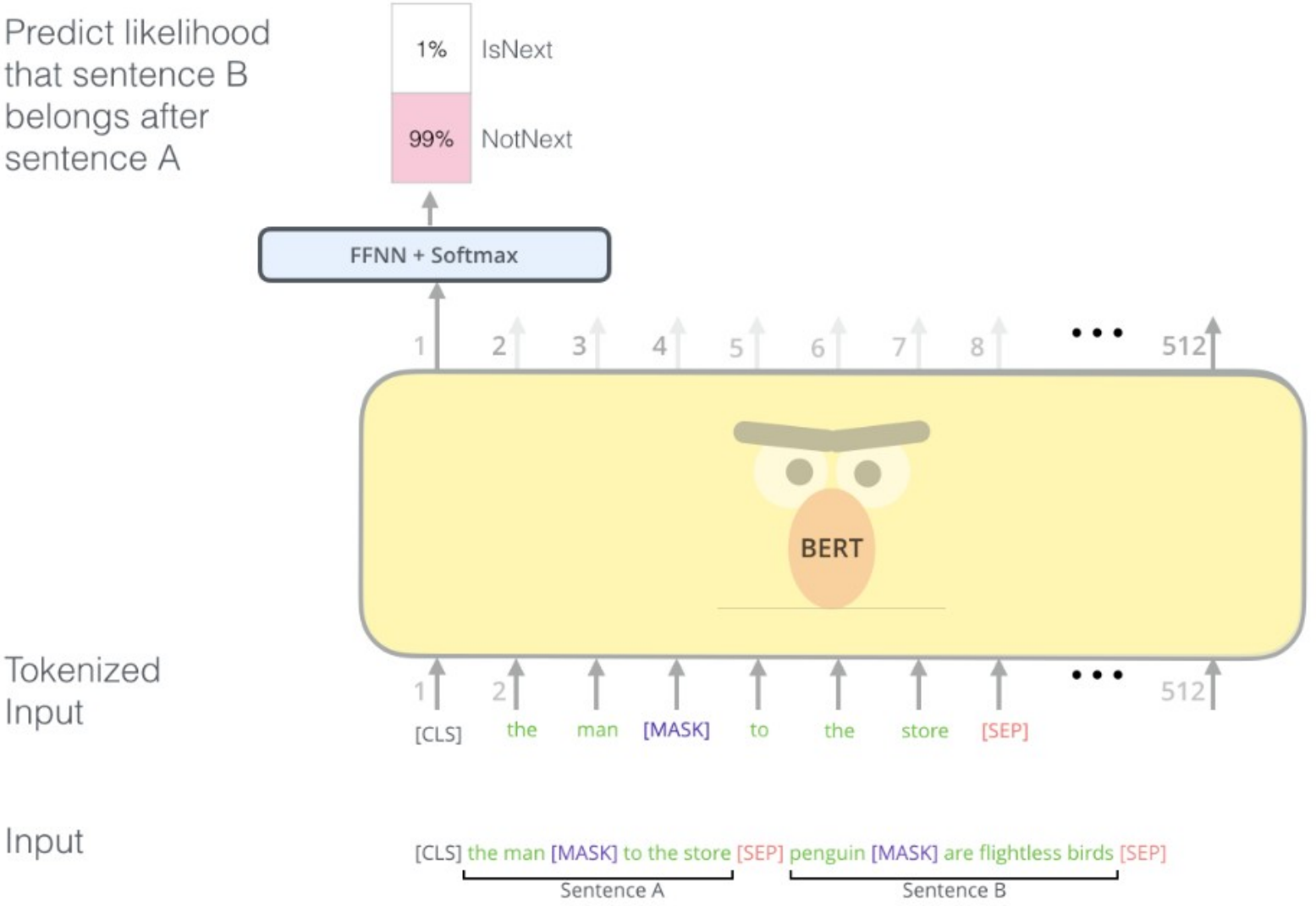
- BERT is trained using two tasks
  - i.e., the model has two outputs, it shares most of the layers for both tasks, and training is done intermixed
- One training batch otpimizes one task and the next one the other task
- Model learns to optimize itself for both tasks, and thus learns regularities that are universal for both tasks
- This is called multitask learning

BERT training: masked LM

Use the output of the masked word's position to predict the masked word



BERT training: does sentence A follow B?



BERT training

- BERT is trained on Wikipedia and BooksCorpus
- BERT uses so-called WordPieces to tokenize input (to avoid having any out-of vocabulary words)
  - Common words are left as is but rare words are split into pieces
  - Original: `The chips from his wood pile refused to kindle a fire to dry his bed-clothes, and he had recourse to a more provident neighbor's to supply the deficiency.`
  - WordPieces: `['the', 'chips', 'from', 'his', 'wood', 'pile', 'refused', 'to', 'kind', '##le', 'a', 'fire', 'to', 'dry', 'his', 'bed', '-', 'clothes', ',', 'and', 'he', 'had', 'rec', '##ours', '##e', 'to', 'a', 'more', 'provide', '##nt', 'neighbor', "'", 's', 'to', 'supply', 'the', 'deficiency', '.']`
-

BERT: contextual embeddings

- BERT „transforms” a static word embedding into a contextual word embedding

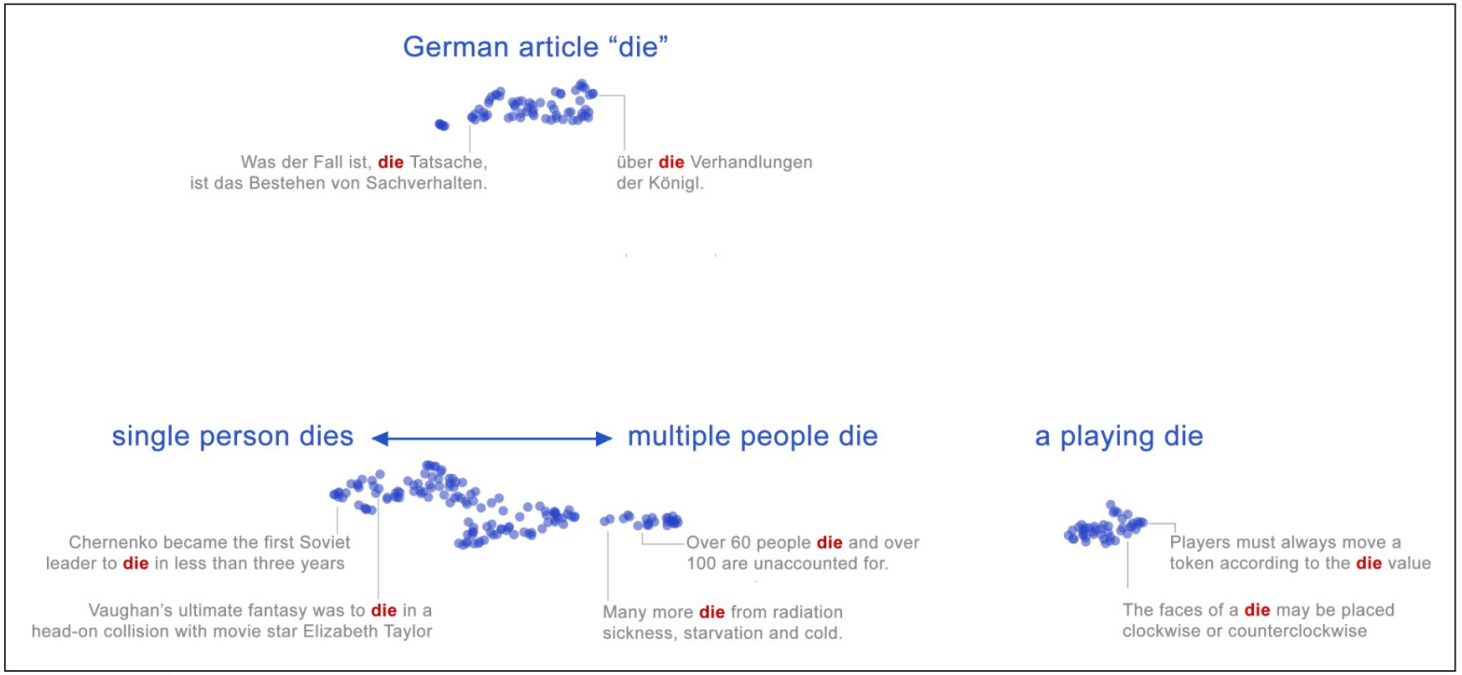
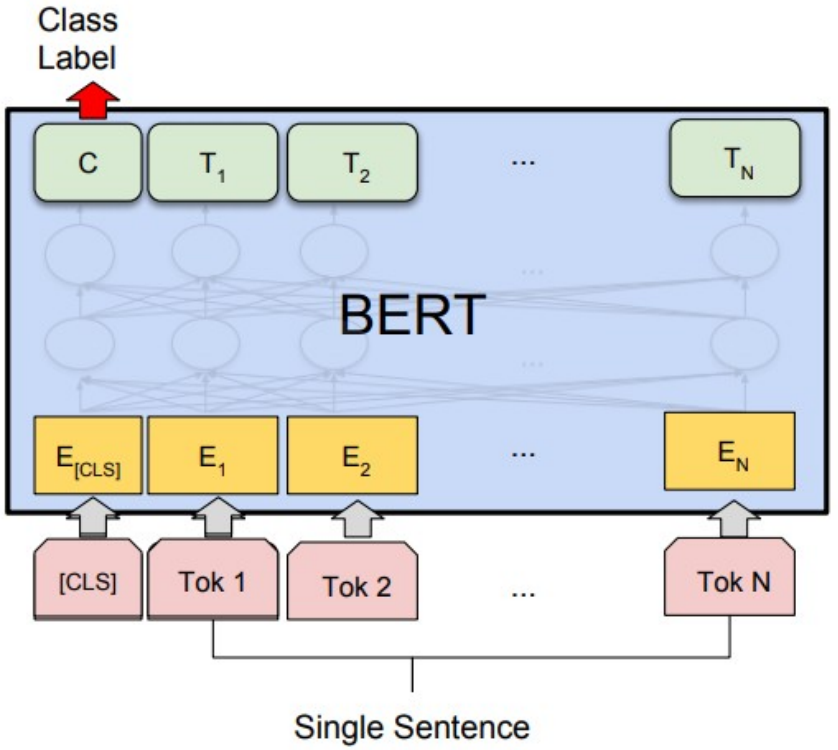


Figure 11.7 Each blue dot shows a BERT contextual embedding for the word *die* from different sentences in English and German, projected into two dimensions with the UMAP algorithm. The German and English meanings and the different English senses fall into different clusters. Some sample points are shown with the contextual sentence they came from. Figure from Coenen et al. (2019).

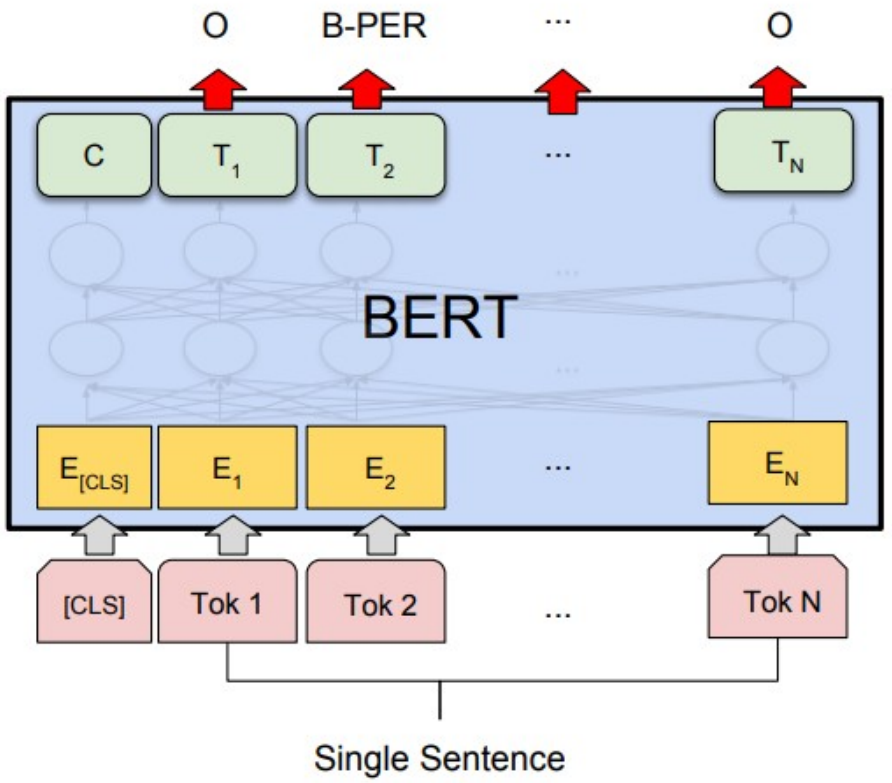
How to use BERT: text classification

- Download a pretrained BERT
- Use „as is” or finetune on (possibly unlabeled) data from your domain
- BERT gives 768 (or 1024 for the large model) dim vector for each input word
  - For text classification, take the 768-dim vector corresponding to [CLS] pseudo-word
  - Add a BERT-to-softmax layer on top
  - Train on your (limited) training data
  - You can choose whether to freeze BERT during training or finetune it together with the classification task



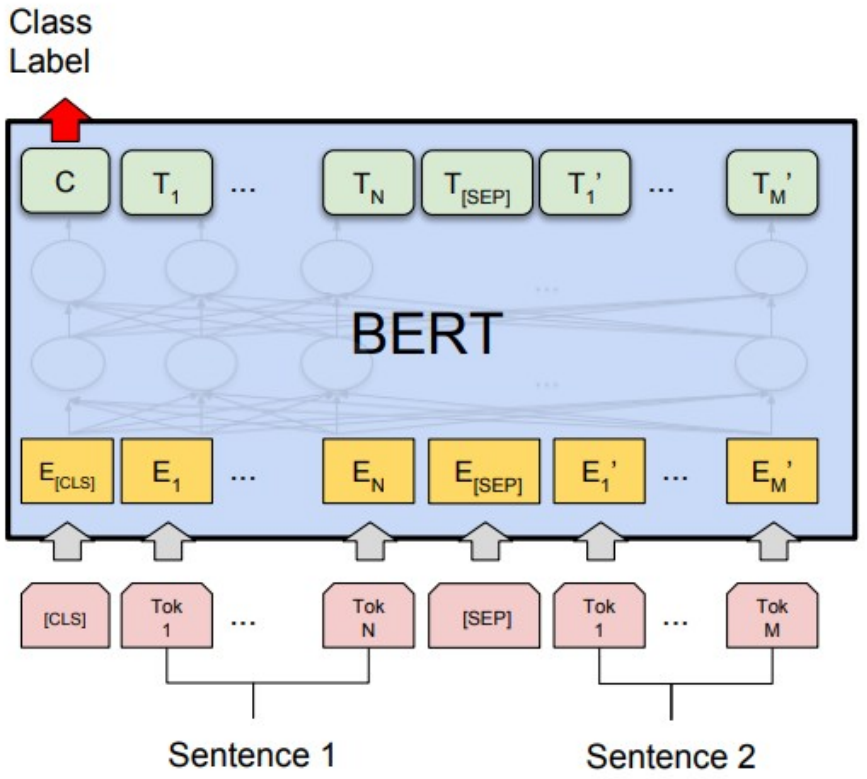
How to use BERT: word classification

- For word classifiifcation (e.g., NER), use outputs of BERT as word 768-dim embeddings, and add some layer(s) on top, e.g.
  - Optional recurrent layer
  - And a final softmax layer



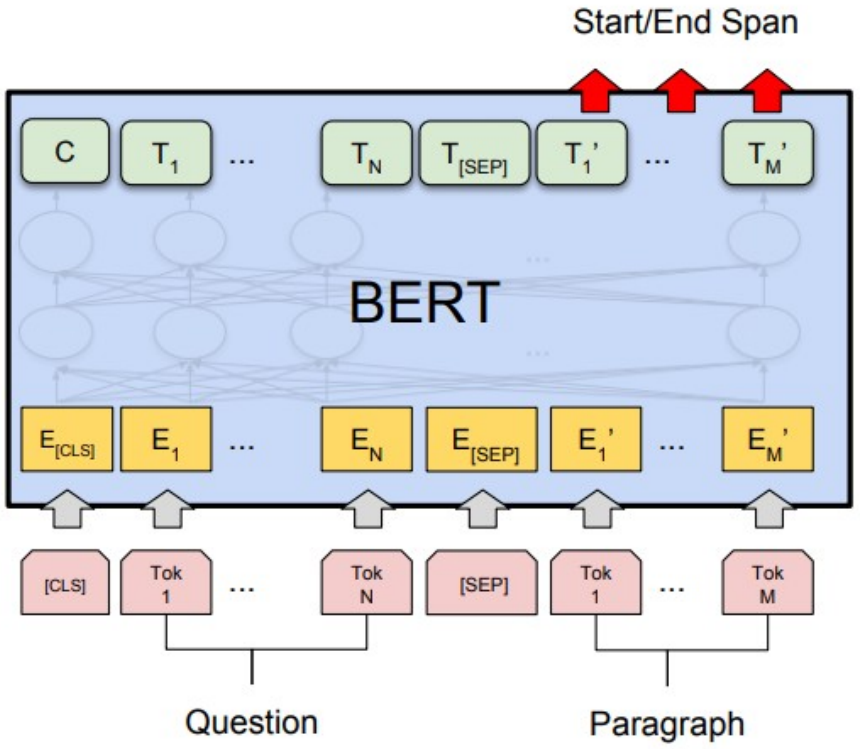
How to use BERT: sentence pair classification

- Sentence pair classification, e.g.:
  - Sentence similiarity detection
  - Textual entailment (does the truth of sentence 2 follow from sentence 1)
- Feed both sentence 1 and sentence 2 to BERT, seperated by the [SEP] token
- Read output corresponding to [CLS] pseudo-word
- Add a softmax layer



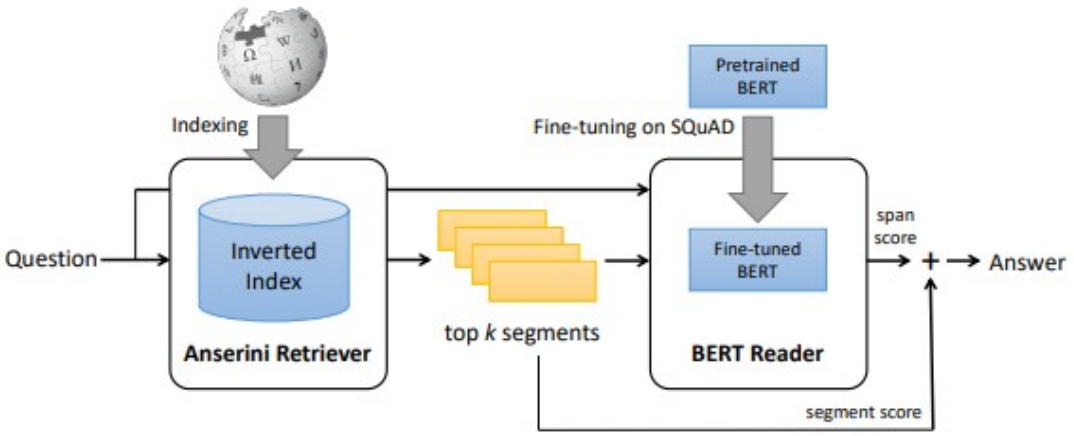
How to use BERT: question answering

- Feed question and the text that contains the answer (paragraph) to BERT, seperated with the [SEP] tag
- Use BERT outputs of the paragraph to find most likely start and end spans that contain the answer
  - Just a softmax over the words of the paragraph for span start
  - Also a softmax (with end>start constraint) for span end



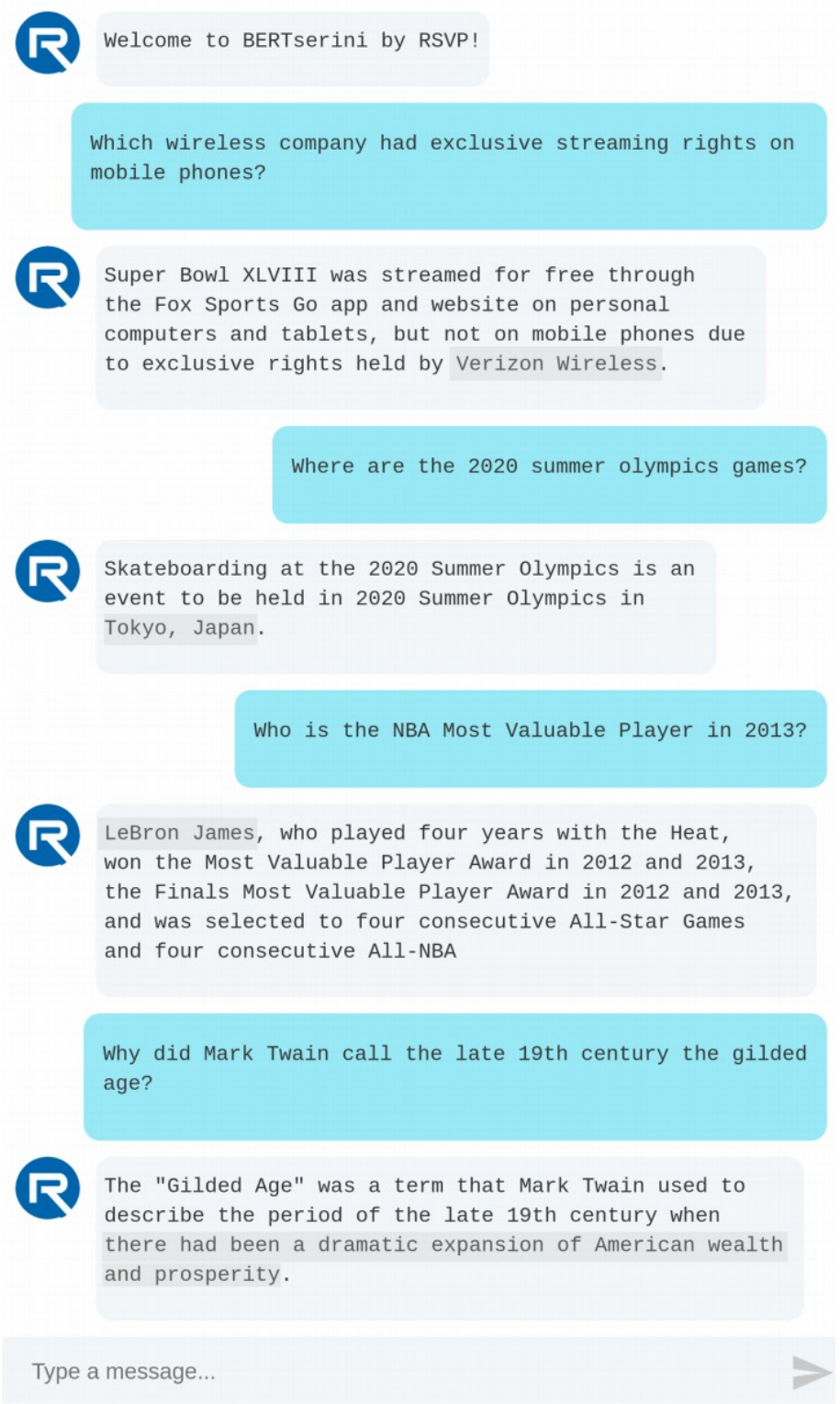
Open-domain question answering using BERT

- Yang et al, *End-to-End Open-Domain Question Answering with BERTserini*, 2019



- Method:
  - Retrieve top *k* paragraphs from Wikipedia using some simple method (e.g., bag-of-words based)
  - Use BERT (fine-tuned on a Question Answering corpus) to find most likely answer (or no answer)

- BERTserini in action



Use BERT

- Pretrained BERT is available for download, and it is relatively easy to use it for almost any NLP task
- If your data is in English, you should consider using it
  - However, it's slow
- A number of more advanced models have been proposed, most with funny names: **RoBERTa**, CamemBERT (BERT for French), XLM, XLNet, T5, etc.
  - EstBERT also available from 2020
- Use the ***transformers*** Python library to use them
- A takeaway from the RoBERTa paper: more compute, more data can improve pretraining even when not changing the underlying Transformer encoder

Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7

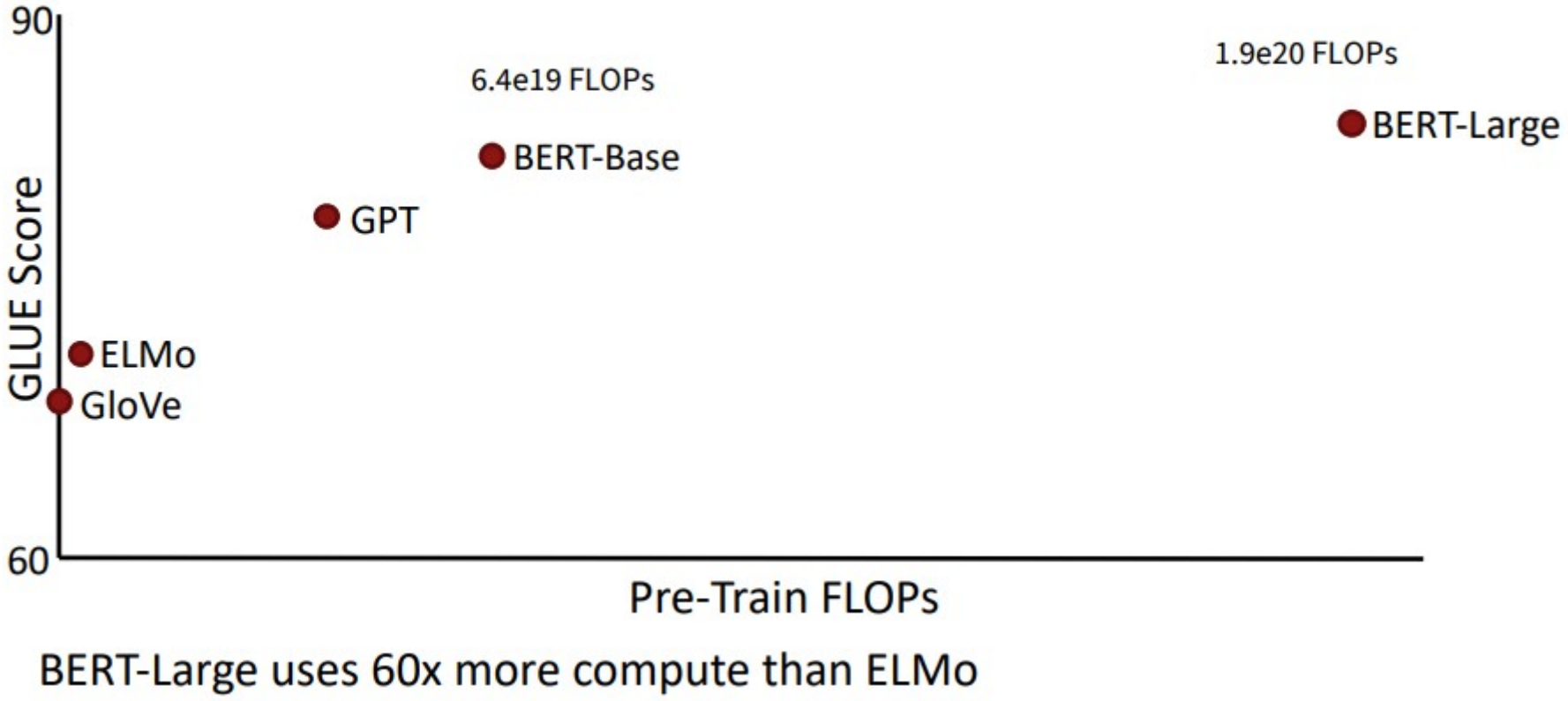
Multilingual pretraining

- BERT is trained only on English text
- What happens if we train on mixed multilingual data?
  - **Multilingual representation emerge!**
  - Why is this good?
    - Can be used in very low-resource scenarios
    - E.g.: you want to do named entity recognition, but have only English training data, and none from Estonian
    - Solution: finetune the pretrained model on English, and use on Estonian
    - It actually works (of course, not as well as with native training data)
- Examples of multilingual pretrained models (all available via *transformers* library):
  - mBERT
  - XLM-Roberta

Multililingual models

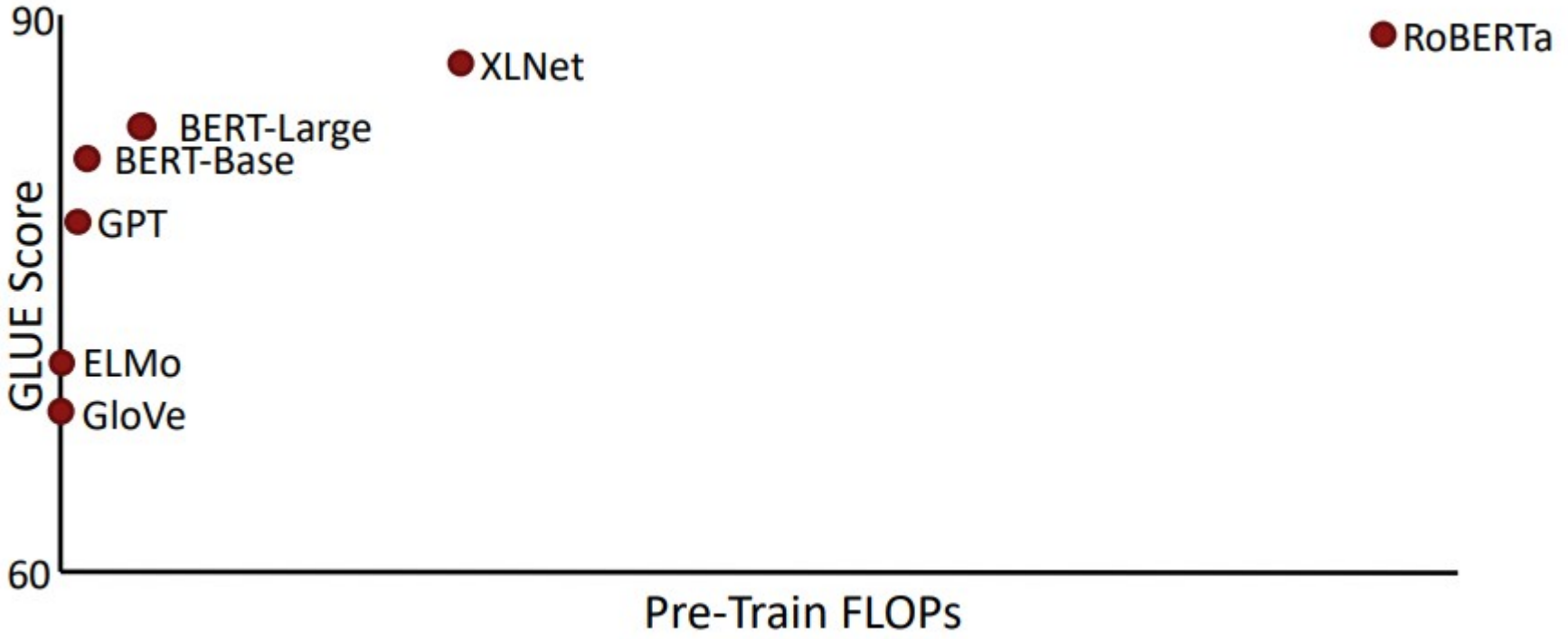
- Multilingual models (such as XLM-RoBERTa) allow to do „zero-shot” cross-lingual training:
  - Train your model on an English dataset, apply it for Estonian!
  - Works surprisingly well for simpler tasks
  - Finetuning with a small amount of language-specific data helps
- Results on Estonian Question Answering tasks (Anu Käver’s Ms thesis, 2021):
  - Model: XLM-RoBERTa
  - Data: Squad (English, ~100K training samples), EstQA (~800 training samples). Evaluation: EstQA test set
  - Results:
    - Finetune on Squad: accuracy 73.4%
    - Finetune on EstQA: 46.1%
    - **Finetune on Squad, then on EstQA: 77.8%**

Pretraining is computationally heavy



BERT-Large uses 60x more compute than ELMo

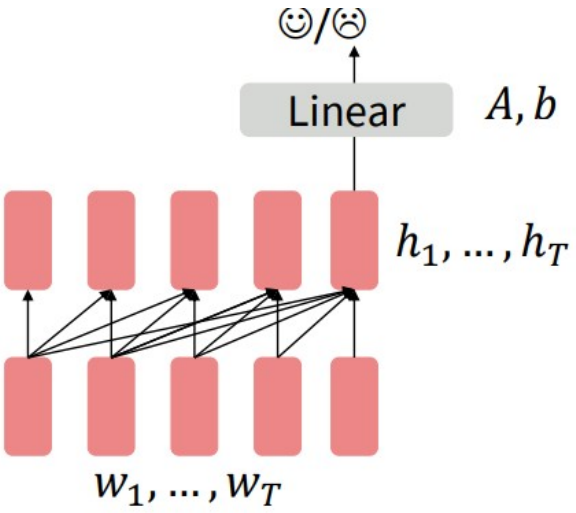
And it gets worse



RoBERTa uses 16x more compute than BERT-Large

GPT

- GPT (Generative Pretarined Transformer)
  - Pretrain using next token prediction (language modeling task!)
  - Nice to generate from; can’t condition on future words
  - Use for downstream task (word or text classification) as any LSTM model
    - E.g. using the hidden state of the last word



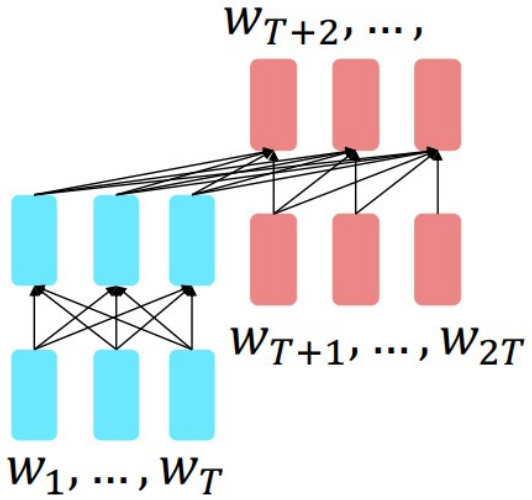
GPT2

- GPT-2, a larger version (1.5B) of GPT (117M params) trained on more data, was shown to produce relatively convincing samples of natural language
- OpenAI even didn’t want to release the model to the public, claiming that it is dangerous to the society

Context (human-written): In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.
GPT-2: The scientist named the population, after their distinctive horn, Ovid’s Unicorn. These four-horned, silver-white unicorns were previously unknown to science. Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved. Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

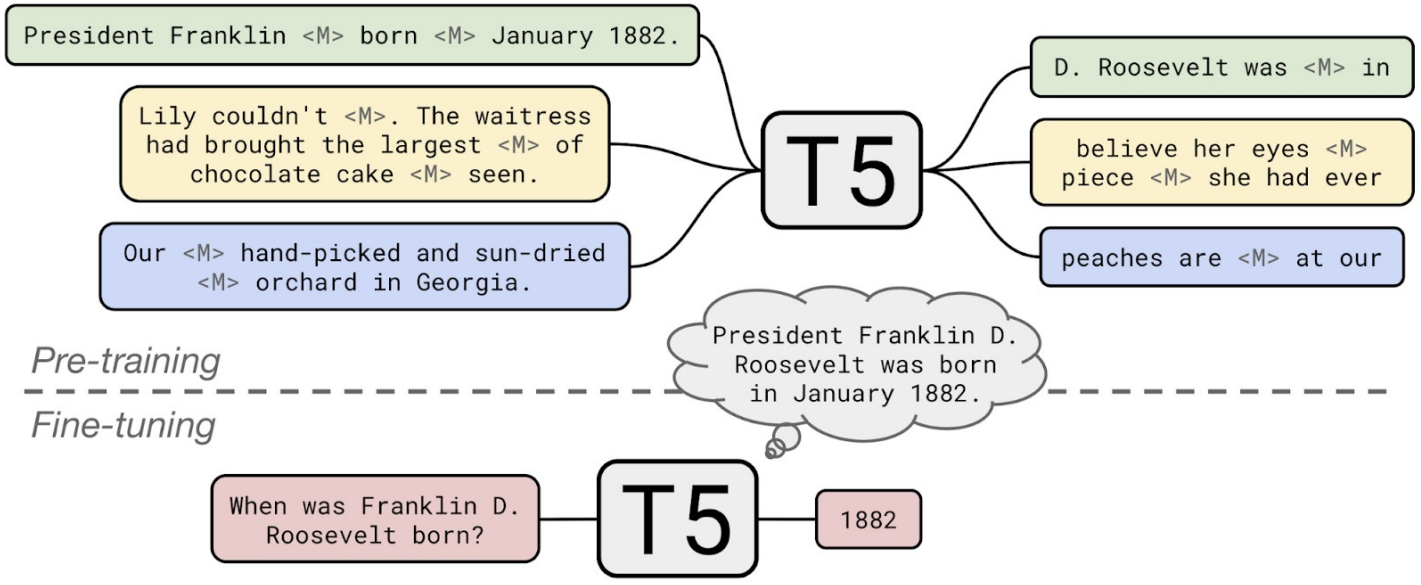
Pretrained encoder-decoders

- For encoder-decoders, we could do something like language modeling, but where a prefix of every input is provided to the encoder and is not predicted
- The encoder portion benefits from bidirectional context; the decoder portion is used to train the whole model through language modeling
- Useful for finetuning for downstream tasks that naturally contain a fixed input sequence and a (generated) output sequence, e.g.:
  - Abstractive summarization
  - Abstractive keyphrase extraction
  - Machine translation
  - Question answering
  - Chatbot



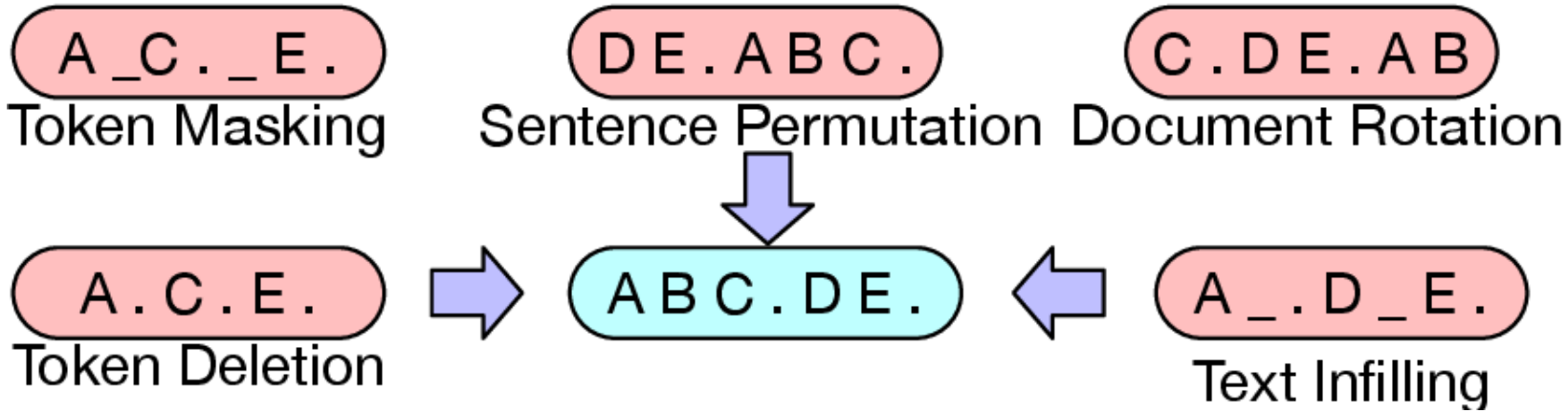
Pretraining encoder-decoders: what pretraining objective to use?

- Google's T5 uses span corruption as a pretraining task
  - Replace different-length spans from the input with unique placeholders;
  - Decode out the spans that were removed!



Pretraining encoder-decoders: what pretraining objective to use?

- Facebook's BART uses donoising as pretraining task
  - Original sentence is noised (partially masked or permuted or „rotated" or partially deleted)
  - Target sentence is the clean sentence
  - Model learns to „denoise" the corrupted input



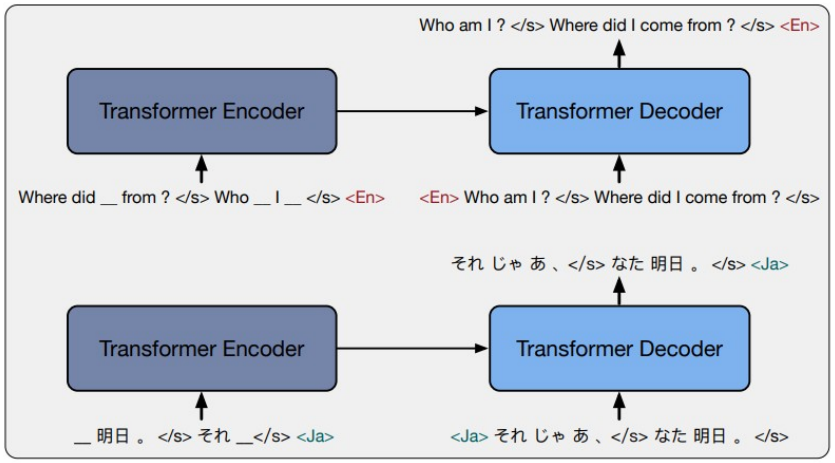
Pretrained seq2seq models for summarization

- Pretrained seq2seq models are very good in summarization
- Without pretraining, summarization models need a lot of training data (millions of article-summary pairs) and even then produce often totally unusable output
- Finetuned BART and T5 can achieve good results even with few summaries to finetune on

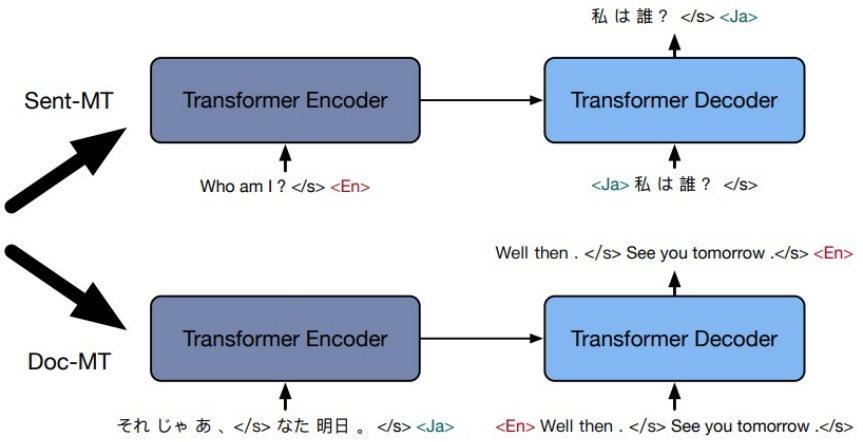
This is the first time anyone has been recorded to run a full marathon of 42.195 kilometers (approximately 26 miles) under this pursued landmark time. It was not, however, an officially sanctioned world record, as it was not an "open race" of the IAAF. His time was 1 hour 59 minutes 40.2 seconds. Kipchoge ran in Vienna, Austria. It was an event specifically designed to help Kipchoge break the two hour barrier.	Kenyan runner Eliud Kipchoge has run a marathon in less than two hours.
PG&E stated it scheduled the blackouts in response to forecasts for high winds amid dry conditions. The aim is to reduce the risk of wildfires. Nearly 800 thousand customers were scheduled to be affected by the shutoffs which were expected to last through at least midday tomorrow.	Power has been turned off to millions of customers in California as part of a power shutoff plan.

mBART -- multilingual

- Pretrained on multilingual data
- Inputs are suffixed and outputs are prefixed using language tags
- Can be finetuned for machine translation with excellent results
  - Especially for low-resource MT
- Also, allows „zero shot cross-lingual transfer”
  - E.g., finetune model on English summarization data, apply for Estonian (without any Estonian data)
  - Not perfect – sometimes „accidental translations” occur



Multilingual Denoising Pre-Training (mBART)



Fine-tuning on Machine Translation

mBART: Results

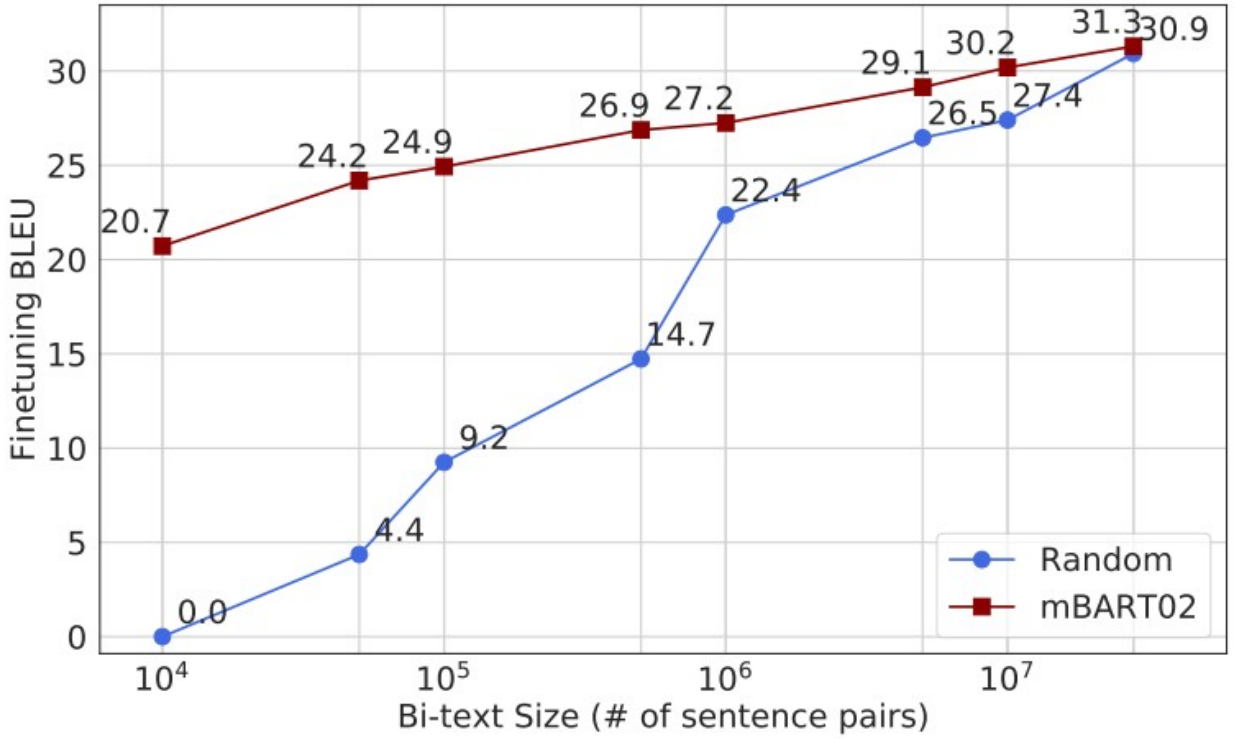
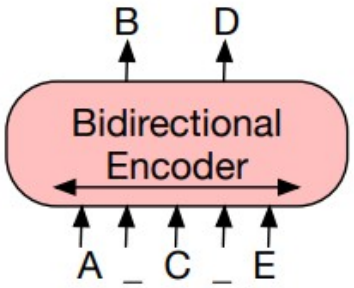
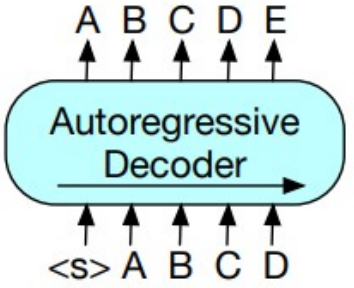


Figure 4: **Fine-tuning curves for En-De along with size of bitext.** The x-axis is on a log scale.

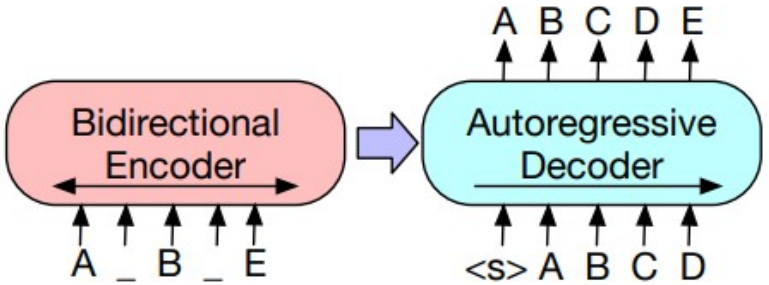
Summary of different types of pretraned NLP models



(a) BERT: Random tokens are replaced with masks, and the document is encoded bidirectionally. Missing tokens are predicted independently, so BERT cannot easily be used for generation.



(b) GPT: Tokens are predicted auto-regressively, meaning GPT can be used for generation. However words can only condition on leftward context, so it cannot learn bidirectional interactions.



(c) BART: Inputs to the encoder need not be aligned with decoder outputs, allowing arbitary noise transformations. Here, a document has been corrupted by replacing spans of text with mask symbols. The corrupted document (left) is encoded with a bidirectional model, and then the likelihood of the original document (right) is calculated with an autoregressive decoder. For fine-tuning, an uncorrupted document is input to both the encoder and decoder, and we use representations from the final hidden state of the decoder.

Questions?