Deep Learning for Natural Language Processing

Eneko Agirre, Gorka Azkune
Ander Barrena, Oier Lopez de Lacalle
@eagirre @gazkune @4nderB @oierldl #dl4nlp
http://ixa2.si.ehu.eus/eneko/dl4nlp

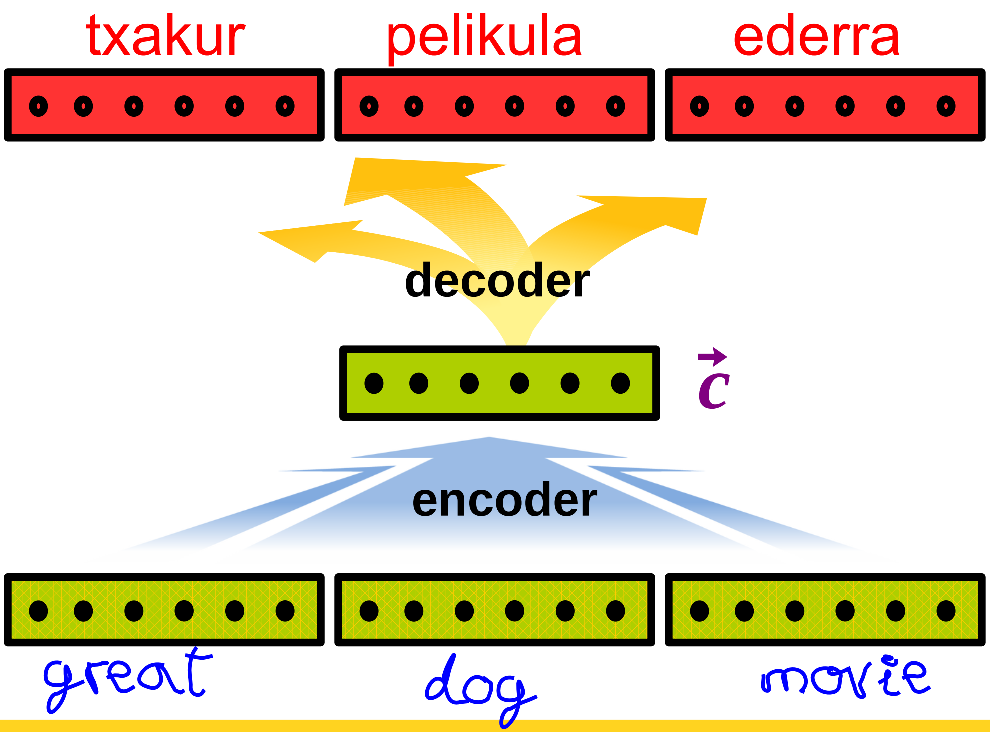
Sess. 6: Pre-trained transformers, BERT, GPT

Plan for the course

- Introduction: machine learning and NLP
- Multilayer perceptron
- Word representation and
  Recurrent neural networks (RNN)
- Sequence-to-Sequence (seq2seq) and
  Machine Translation
- Attention, transformers and
  Natural language inference
- Pre-trained transformers, BERT, GPT
- Bridging the gap between natural languages
  and the visual world

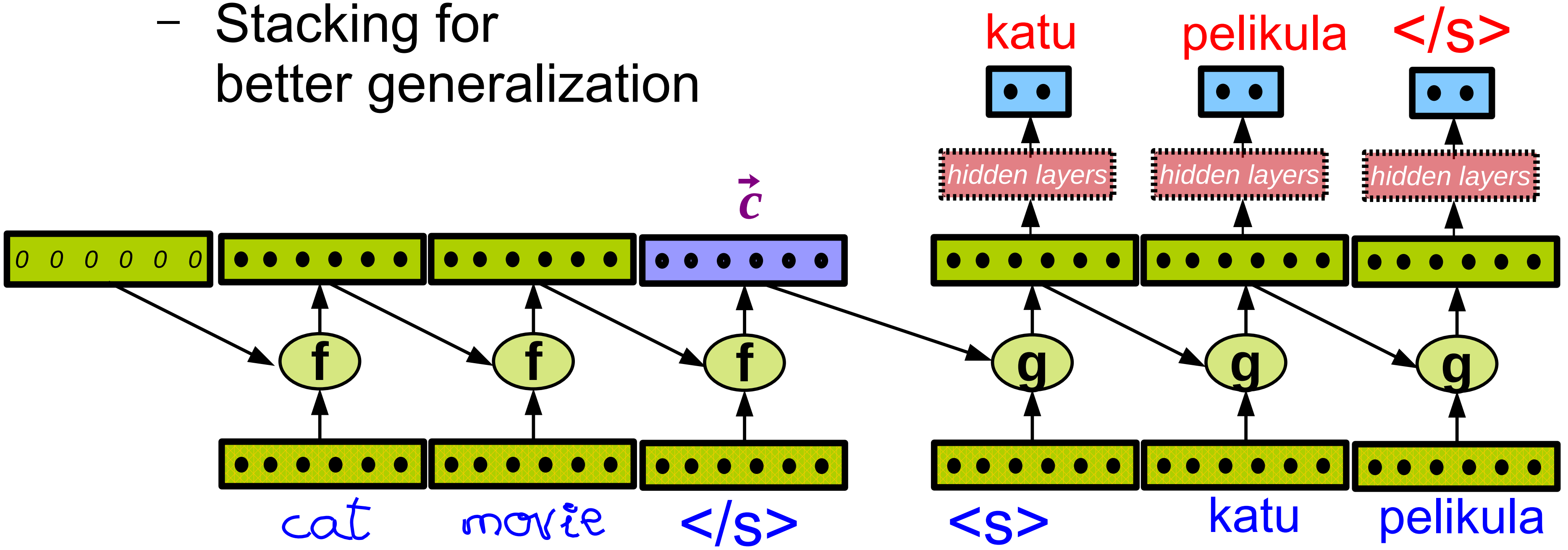
Previously...

- Learning representations
  - Word embeddings
  - Feedforward layer
  - RNN
  - Transformer



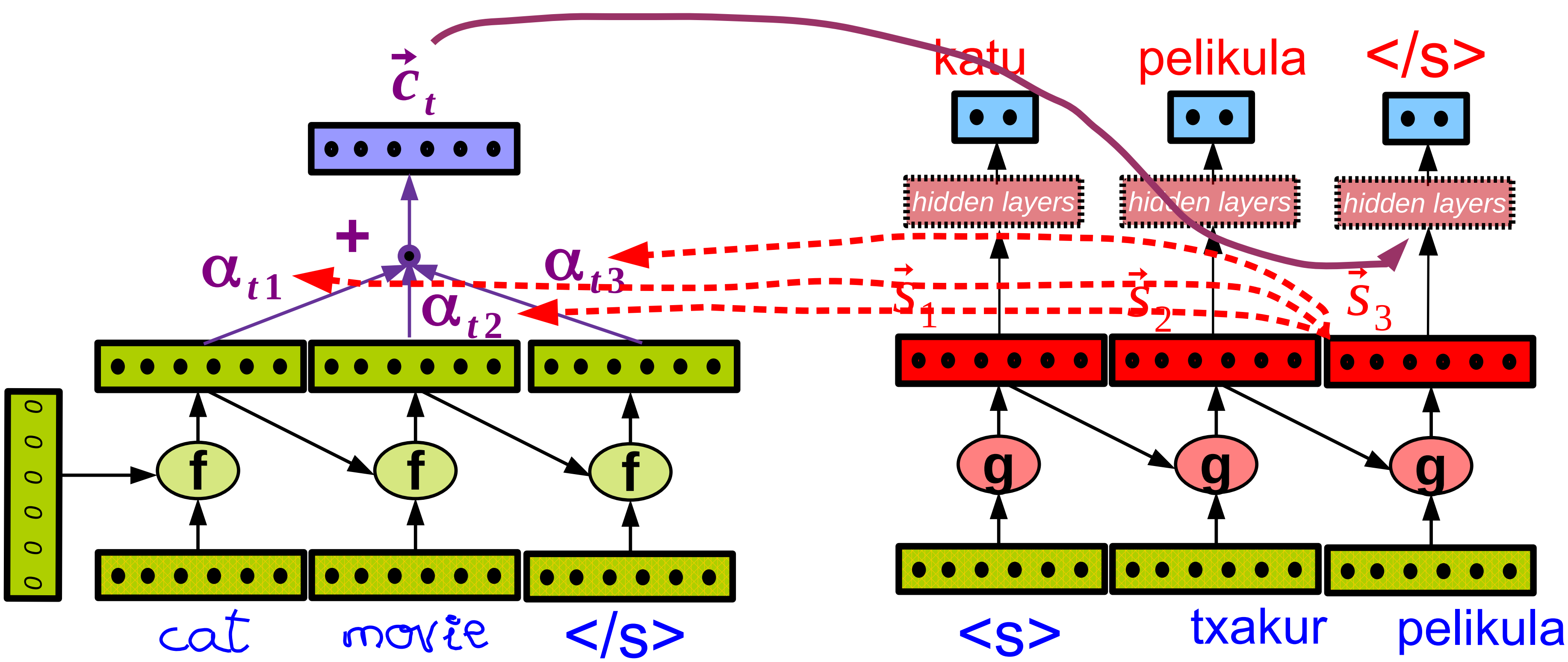
Previously...

- Architectures
  - Softmax for classification
  - Encoders, decoders, siamese networks
  - Stacking for better generalization



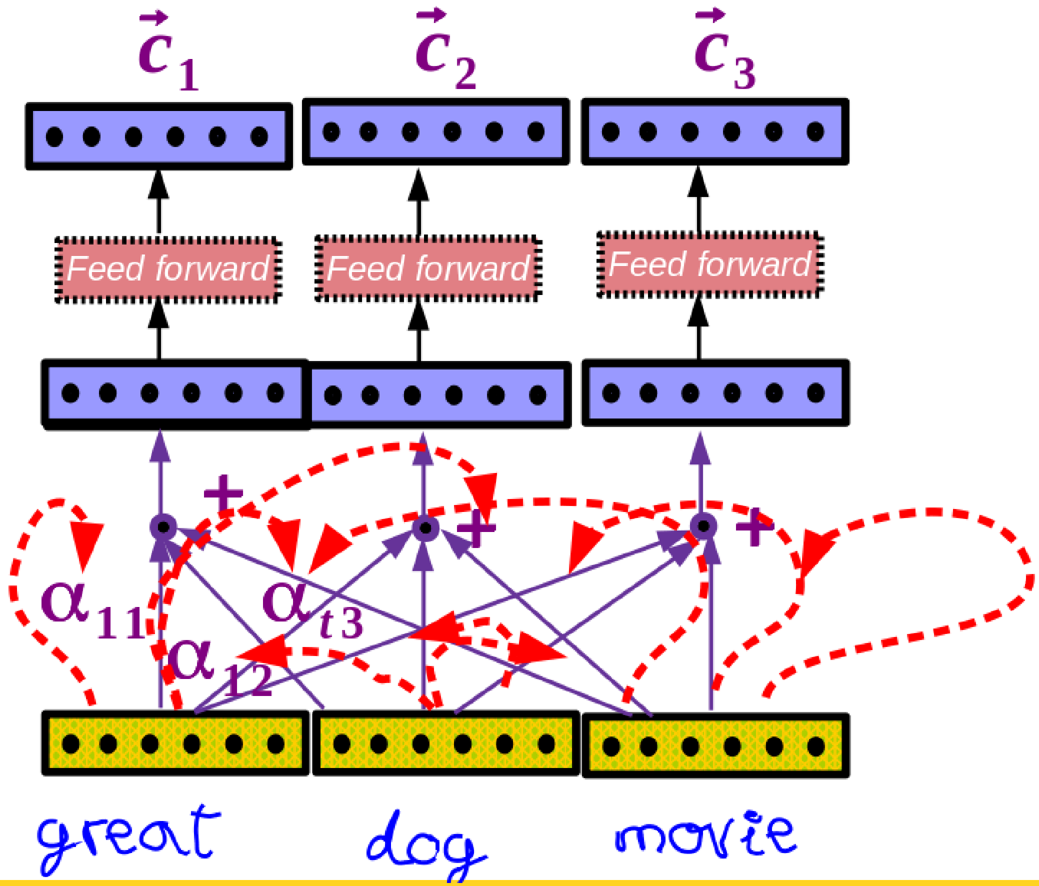
Previously...

- Architectures
  - Cross-attention



Previously...

- Architectures
  - Self-attention



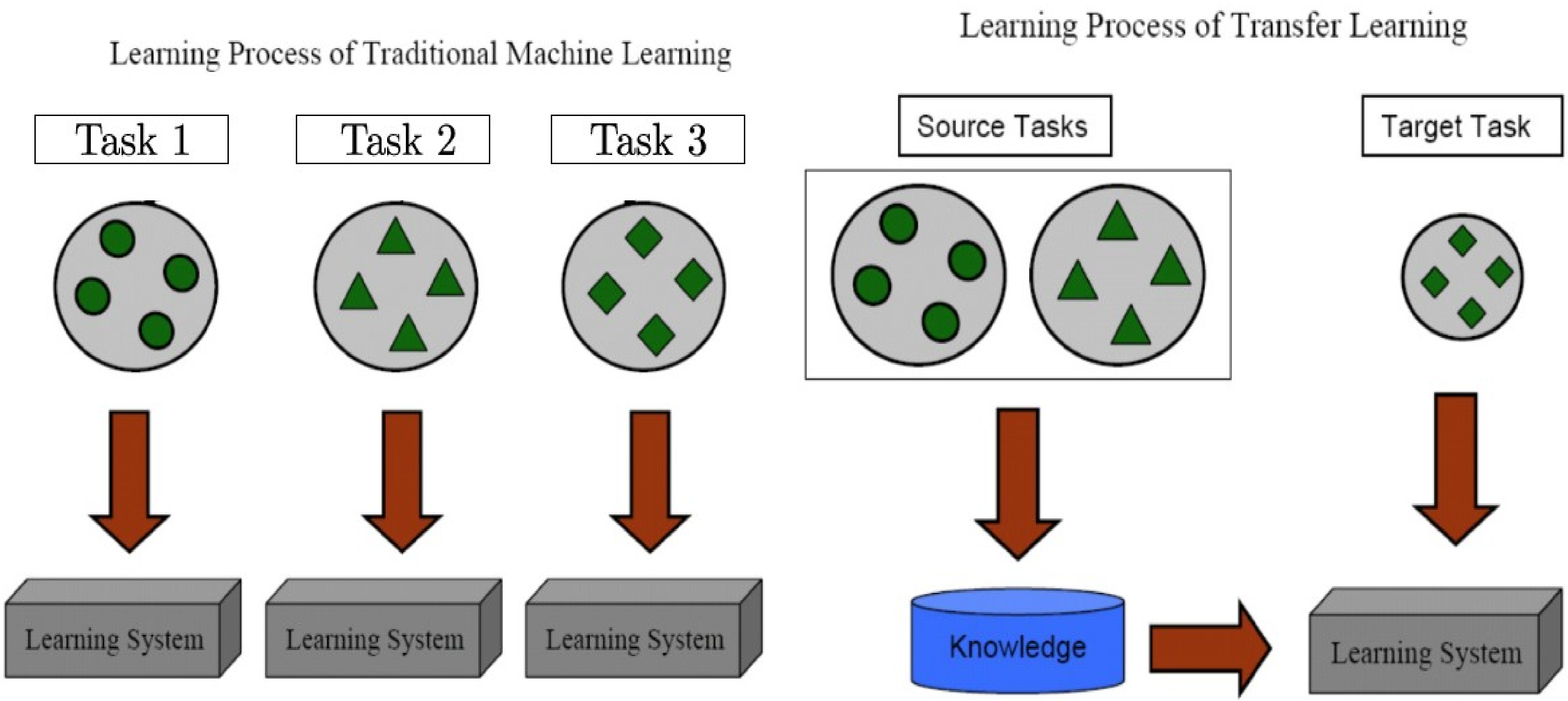
Previously...

- SGD, backpropagation and loss functions
- Regularization, dropout and early stopping
- Residual connections, LSTM or GRUs (vanishing gradients)
- Gradient clipping (exploding gradients)
- Layer normalization

Plan for this session

- Transfer learning:
  - Pre-trained LM, BERT, GPT
  - Pre-trained multilingual LM
  - Prompting
- Deep learning frameworks
- Last words

Transfer learning

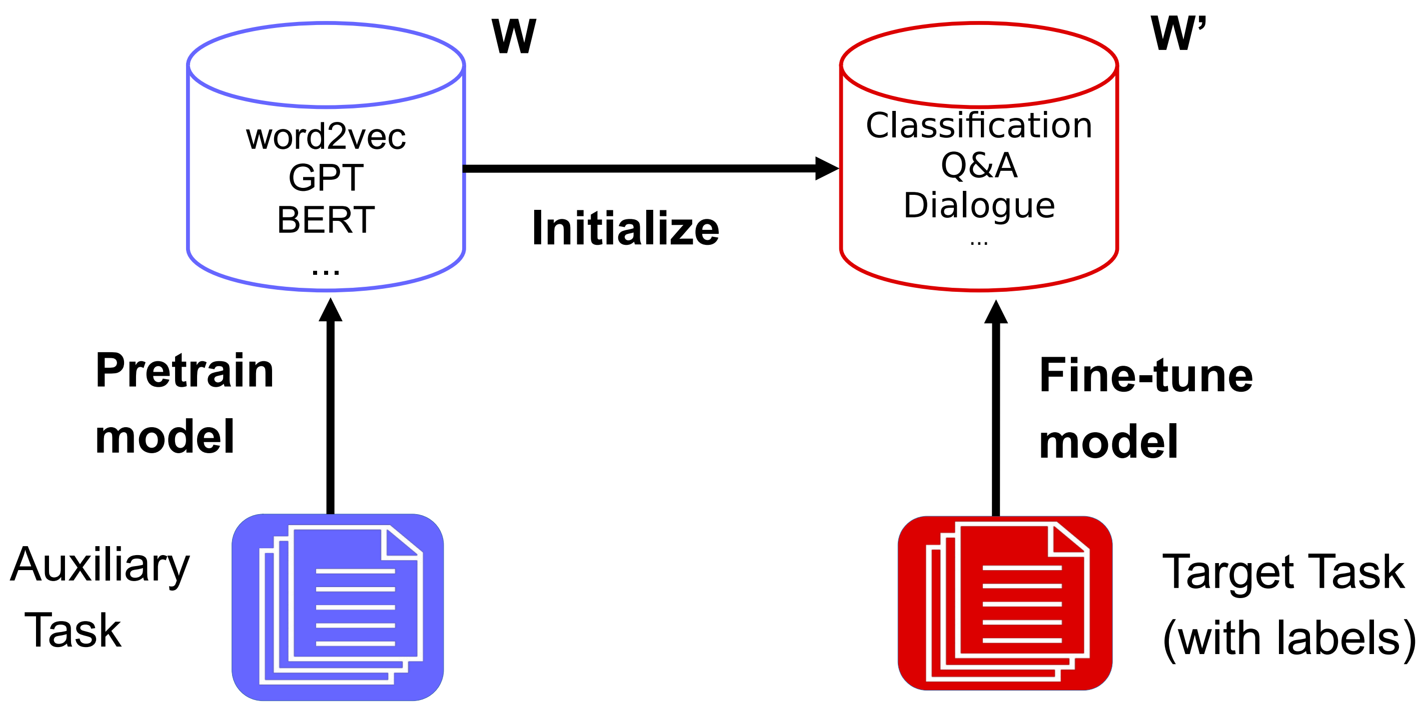


(a) Traditional Machine Learning (b) Transfer Learning

[Pan and Yang (2010)]

Transfer learning in NLP

Learn on one task (or dataset)
then transfer and adapt to target task (or dataset)



Why transfer learning in NLP?

- Many NLP tasks share common knowledge about language (e.g. linguistic representations, structural similarities)
- Tasks can inform each other
  — e.g. syntax and semantics
- Annotated data is rare, make use of as much supervision as available
- Empirically, transfer learning has resulted in SOTA for most supervised NLP tasks (classification, information extraction, Q&A, etc)

Why transfer learning in NLP?

Pre-trained transformers have revolutionized NLP, e.g. GPT-3, BERT

Support The Guardian
Available for everyone, funded by readers
Contribute → Subscribe →
Sign in
The Guardian

News | Opinion | Sport | Culture | Lifestyle

World UK Environment Science Cities Global development Football **Tech** Business More

Artificial intelligence (AI)

New AI fake text generator may be too dangerous to release, say creators

The Elon Musk-backed nonprofit company OpenAI declines to release research publicly for fear of misuse

Forbes
Billionaires Innovation Leadership Money Business Small Business Lifestyl

534 views | Nov 14, 2019, 12:56pm

Google Uses BERT Technology To Develop Its Search Results

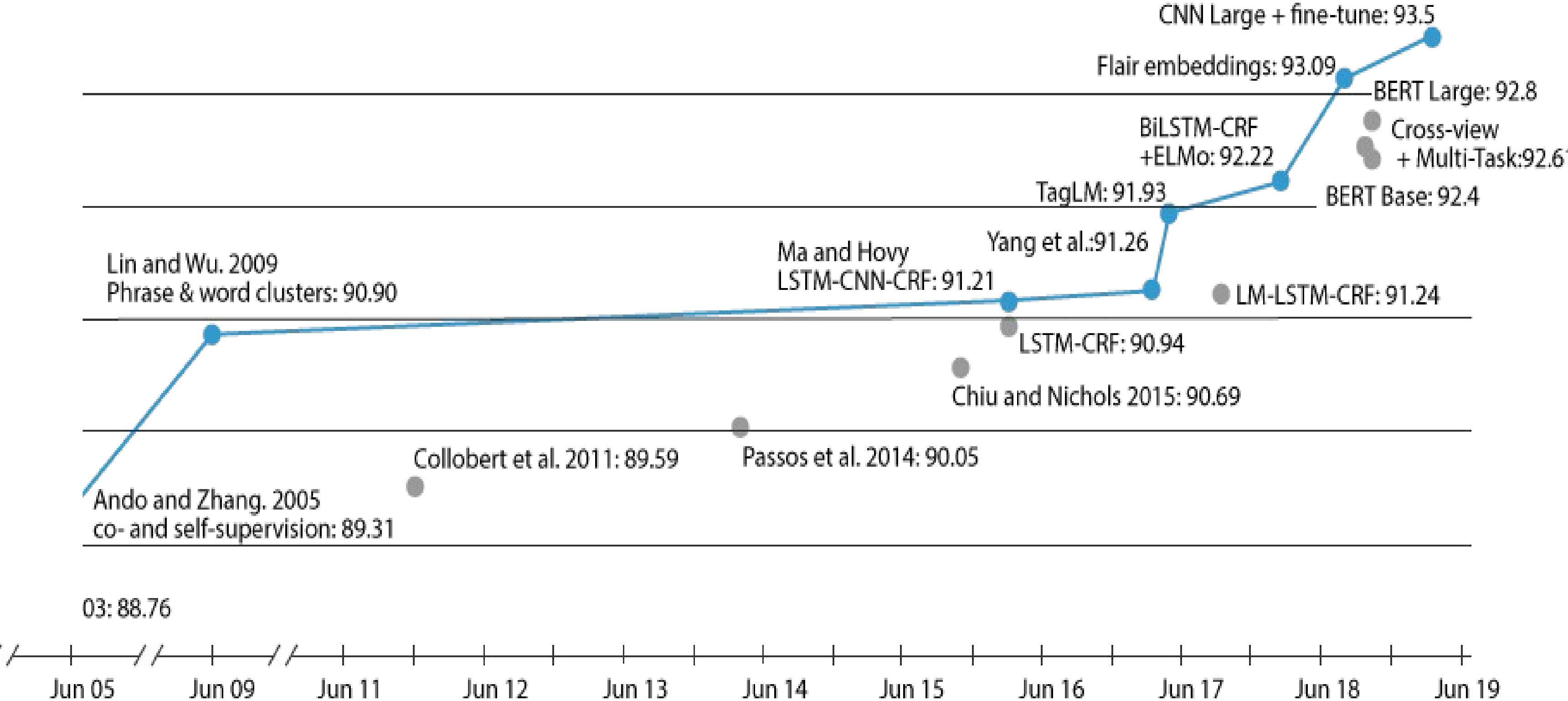
Ilker Koksal Contributor
Enterprise & Cloud

Google has recently gone live with their latest update that involves the use of BERT technology in search engine results. According to HubSpot, Google processes over 70 000 search inquiries per second. If you do the math, this number leads to 5.8 billion searches per day.

Besides making such large amount of searches per second possible, Google is also working on improving search results to provide specific and targeted content. The use of BERT and the new neural networking techniques will lead to significant changes in the search algorithm.

Why transfer learning in NLP?

Performance on Named Entity Recognition (NER) on CoNLL-2003 (English) over time



Pre-training tasks (datasets)

OPTION 1) Supervised pre-training

- Very common in vision (ImageNet),
  less in NLP due to lack of large labeled datasets
- Machine translation
- NLI for sentence representations
- Task-specific — transfer from one Q&A dataset to another

Pre-training tasks and datasets

OPTION 1) Supervised pre-training

- Very common in vision (ImageNet),
  less in NLP due to lack of large labeled datasets
- Machine translation
- NLI for sentence representations
- Task-specific — transfer from one Q&A dataset to another

OPTION 2) Self-supervised (LM) pre-training

- Unlabeled data (~unsupervised, self-learning)
- Easy to gather HUGE corpora:
  Wikipedia, news, web crawl, social media, etc.
- Train some variant of a Language Model

Self-supervised LM pre-training

- Most successful pre-training
- Informally, a LM learns some variant of
  $P_{\theta}(text)$ or $P_{\theta}(text \mid some\ other\ text)$

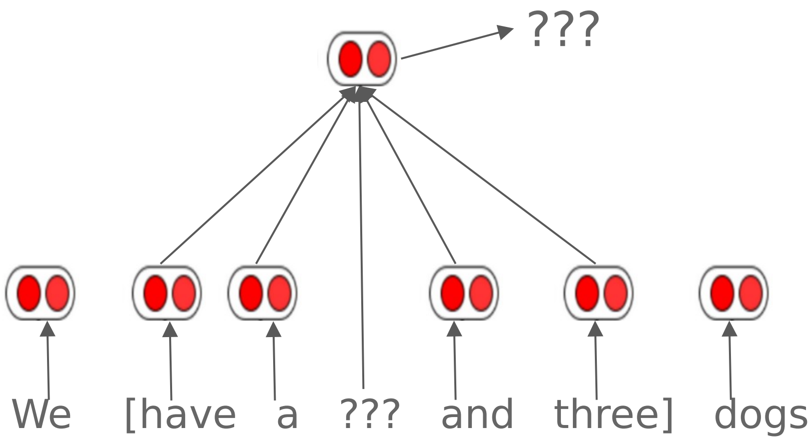
- In order to do transfer learning:

  1) Pre-train a (transformer) model as a language model
  2) Fine-tune in target task

     OR extract contextual embeddings

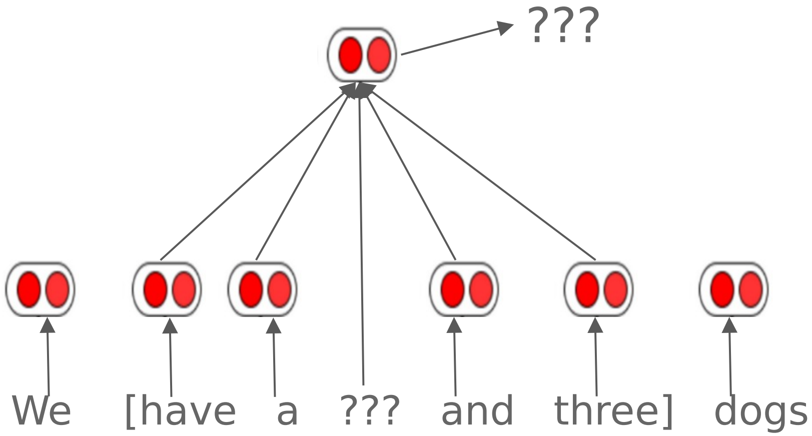
LM variants

word2vec, Mikolov et al (2013)

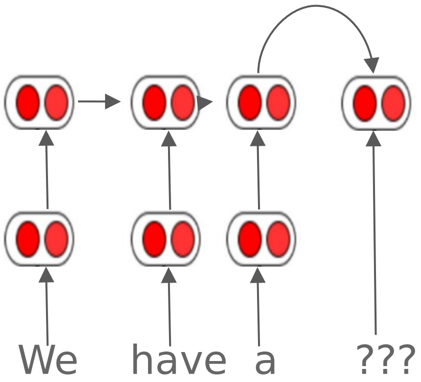


LM variants

word2vec, Mikolov et al (2013)

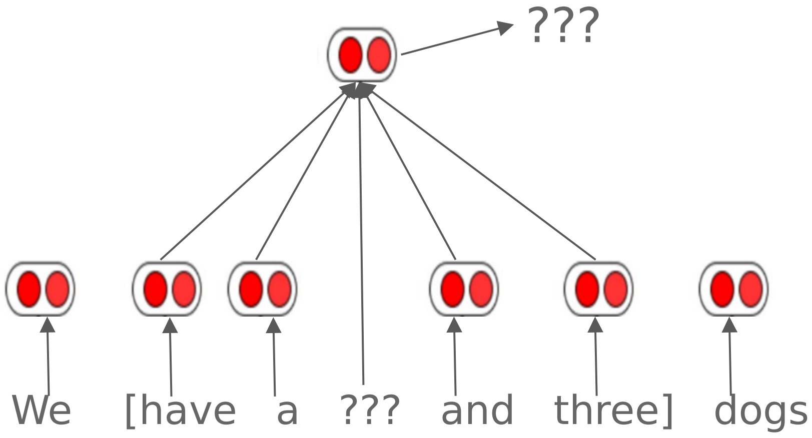


GPT (Radford et al. 2018)

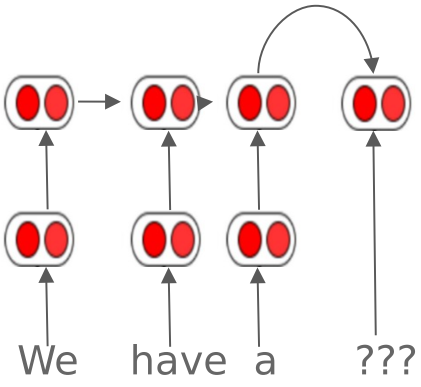


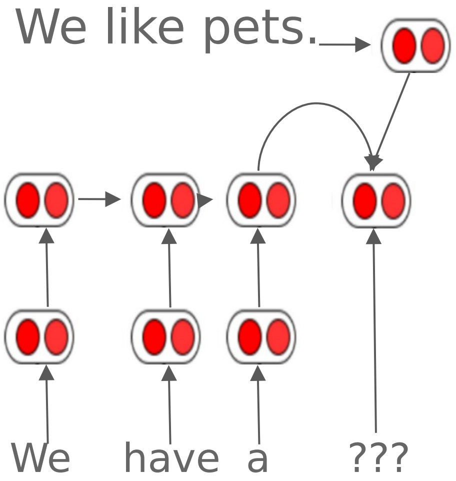
LM variants

word2vec, Mikolov et al (2013)



GPT (Radford et al. 2018)



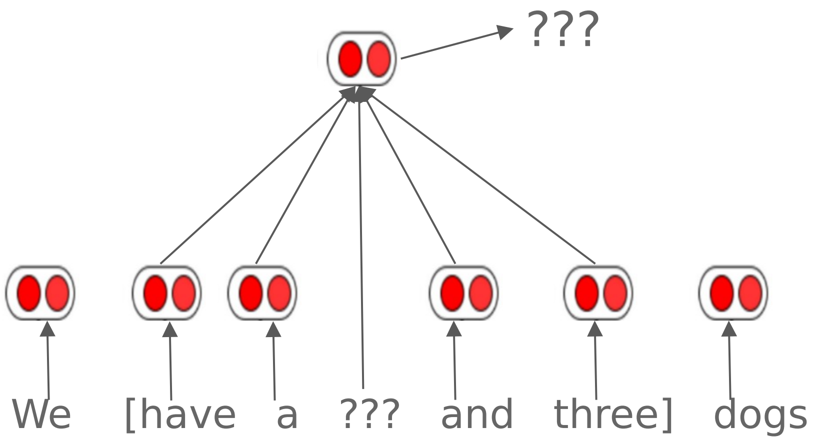


Skip-Thought

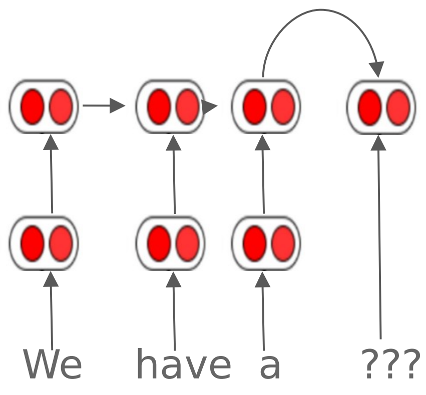
(Kiros et al., 2015)

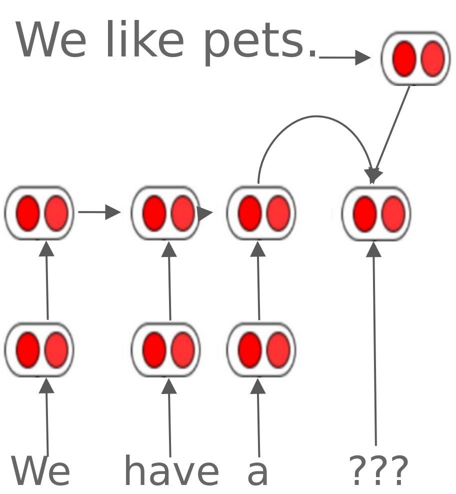
LM variants

word2vec, Mikolov et al (2013)



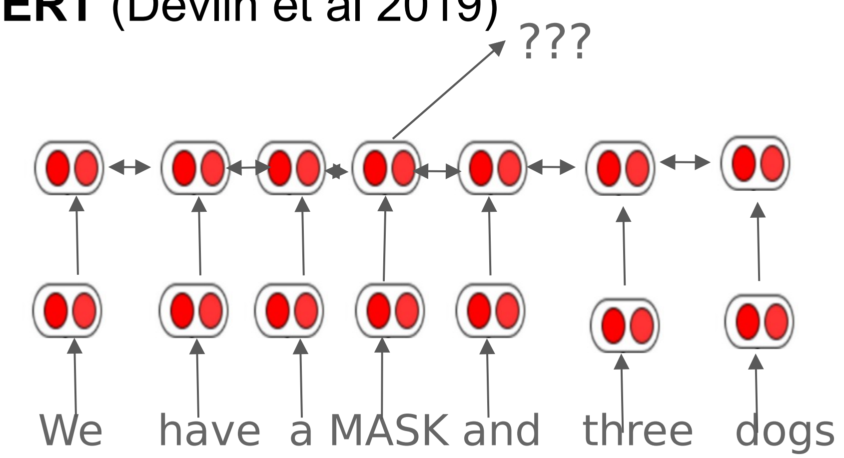
GPT (Radford et al. 2018)





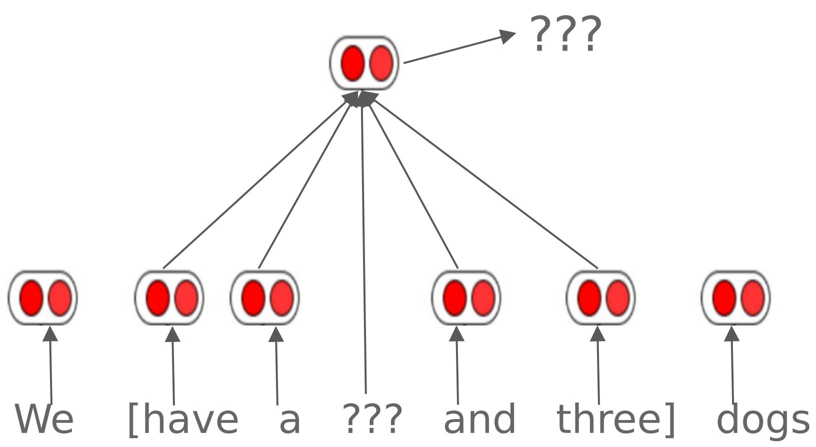
Skip-Thought
(Kiros et al., 2015)

BERT (Devlin et al 2019)



word2vec

Efficient algorithm + large scale training
→ high quality word vectors



Mikolov et al., 2013 See also:

Pennington et al. (2014): GloVe
Bojanowski et al. (2017): fastText

word2vec

Word embedding methods (e.g. word2vec) learn one vector per word:

cat = [0.1, -0.2, 0.4, …]

dog = [0.2, -0.1, 0.7, …]

word2vec

Word embedding methods (e.g. word2vec) learn one vector per word:

cat = [0.1, -0.2, 0.4, …]

dog = [0.2, -0.1, 0.7, …]

PRP	VBP	PRP	NN	CC	NN	.
I	love	my	cat	and	dog	.

I love my cat and dog . => “positive"

Contextual word vectors Motivation

Word vectors compress all contexts into a *single vector*

Nearest neighbor vectors to “**play**”

game
player
score
compete
playmaking
performances
rehearse
theater

Contextual word vectors Key Idea

Instead of learning one **static** vector per word,
learn a vector that depends on **context**

vector(**play** | *The kids **play** football in the park.*)

!=

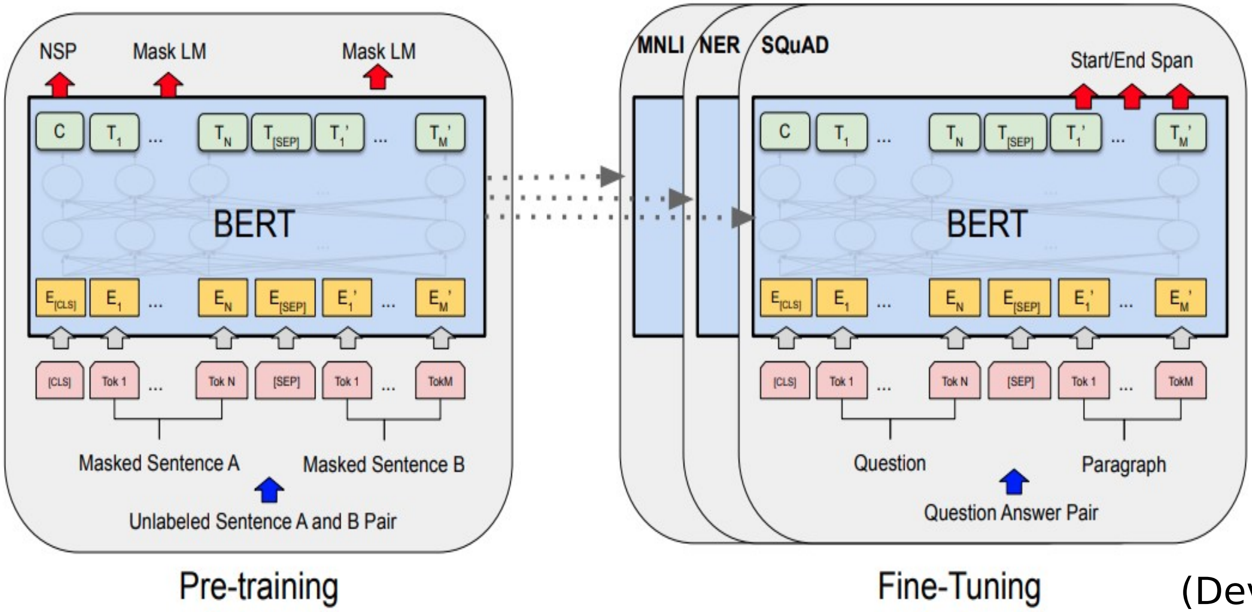
vector(**play** | *The Broadway **play** premiered yesterday.*)

Learn **contextual word embeddings**!

Requires **pre-trained language models**
i.e. requires to run the model, not a static embedding file

BERT

- Pretrains sentence and contextual word representations
- Architecture: Transformer-based encoder
  BERT-large: 24 layers, 16 attention heads, 1024 dim. (340M par.)
- Losses: masked LM and next sentence prediction

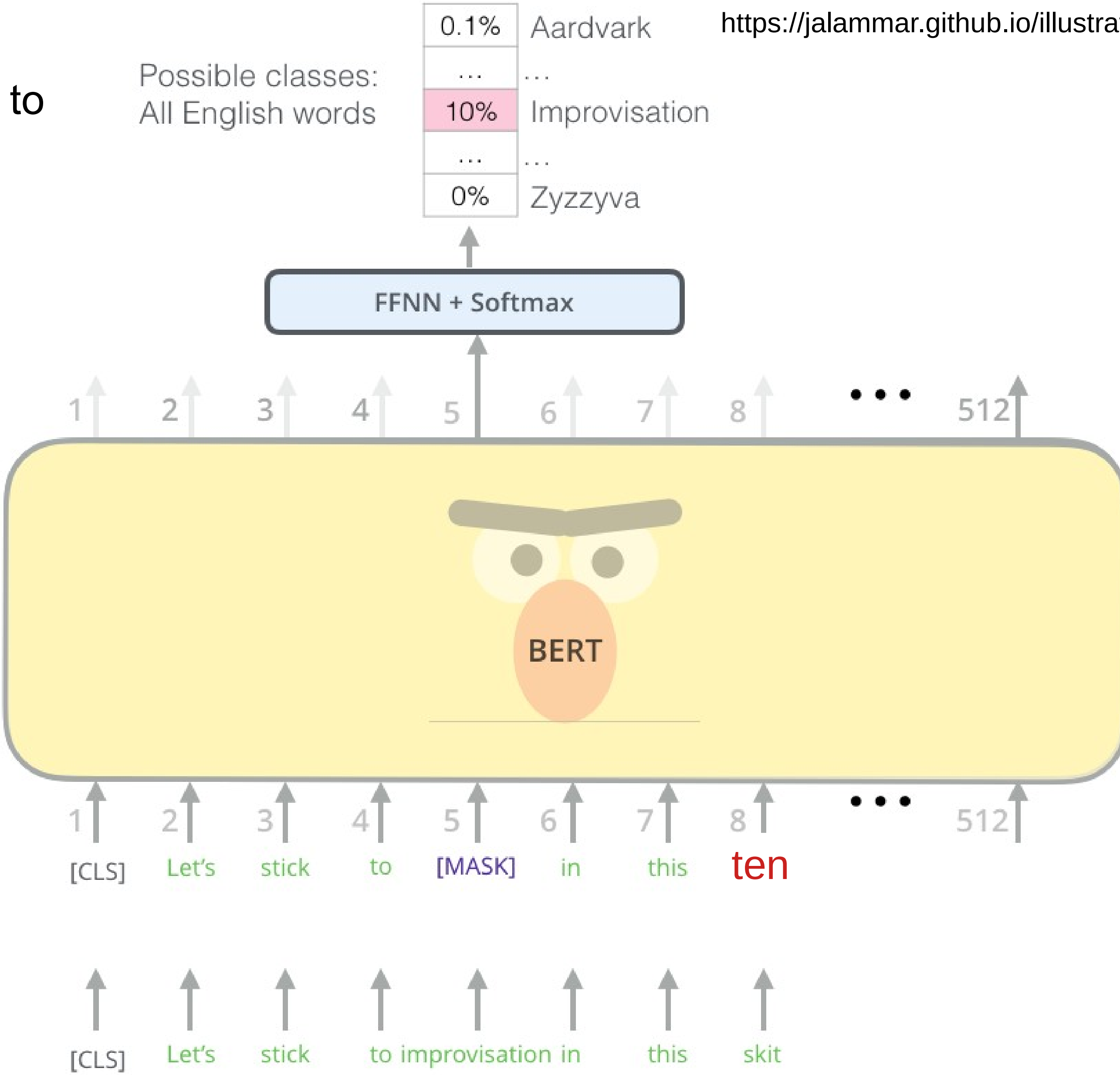


(Devlin et al. 2019)

BERT pre-training (MLM loss)

Use the output of the wrong word's position to predict the hidden or correct word

https://jalammar.github.io/illustrated-bert/

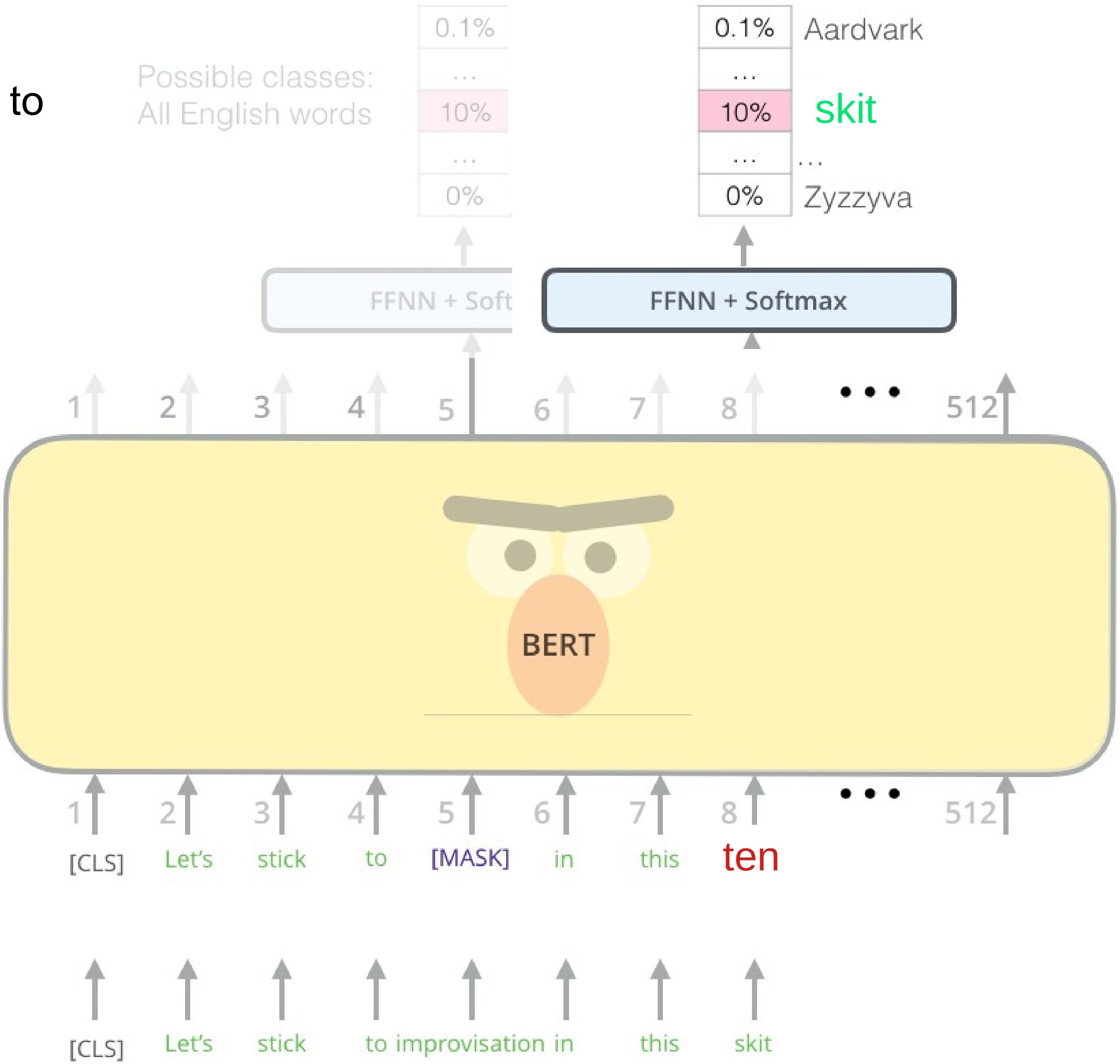


Choose 15% of tokens and mask them (80%) or replace them (10%)

Input

BERT pre-training (MLM loss)

Use the output of the wrong word's position to predict the hidden or correct word



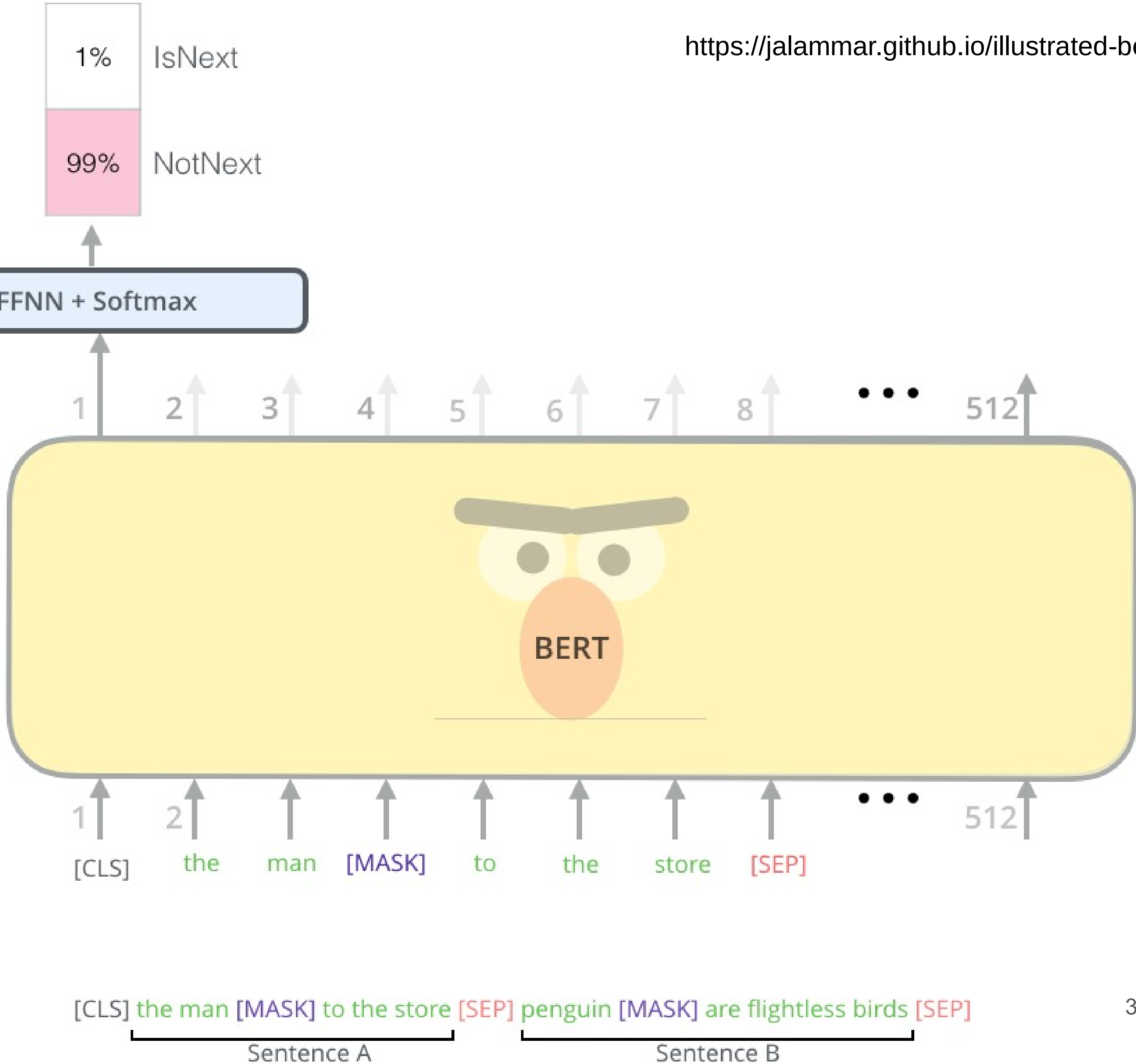
Choose 15% of tokens and mask them (80%) or replace them (10%)

Input

BERT pre-training (NSP loss)

https://jalammar.github.io/illustrated-bert/

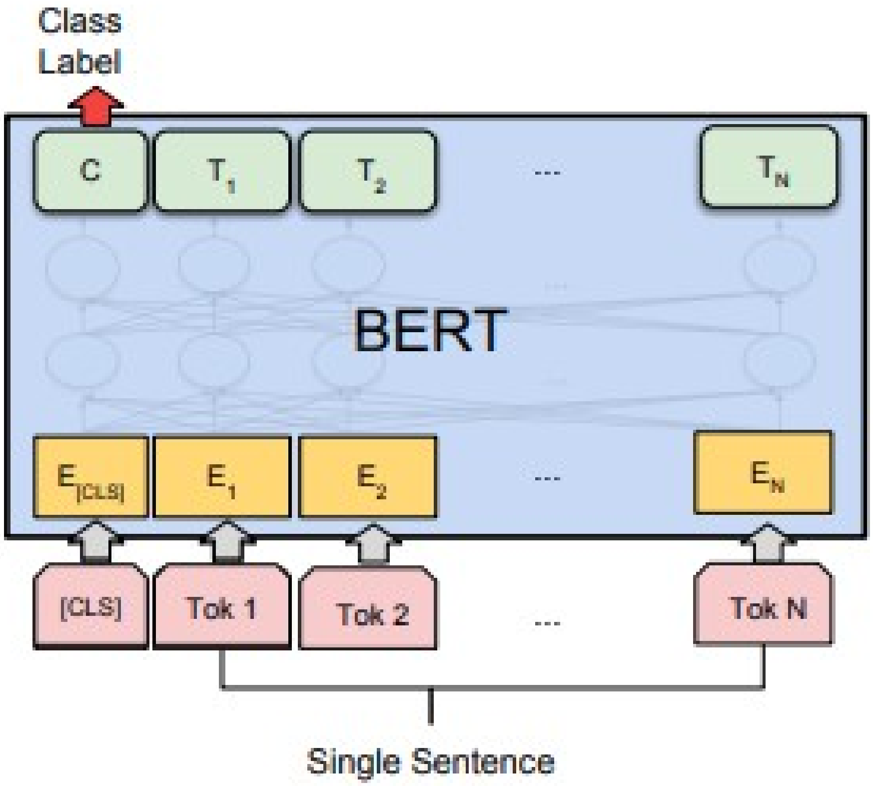
Predict whether sentence B belongs after sentence A



Tokenized input

Input

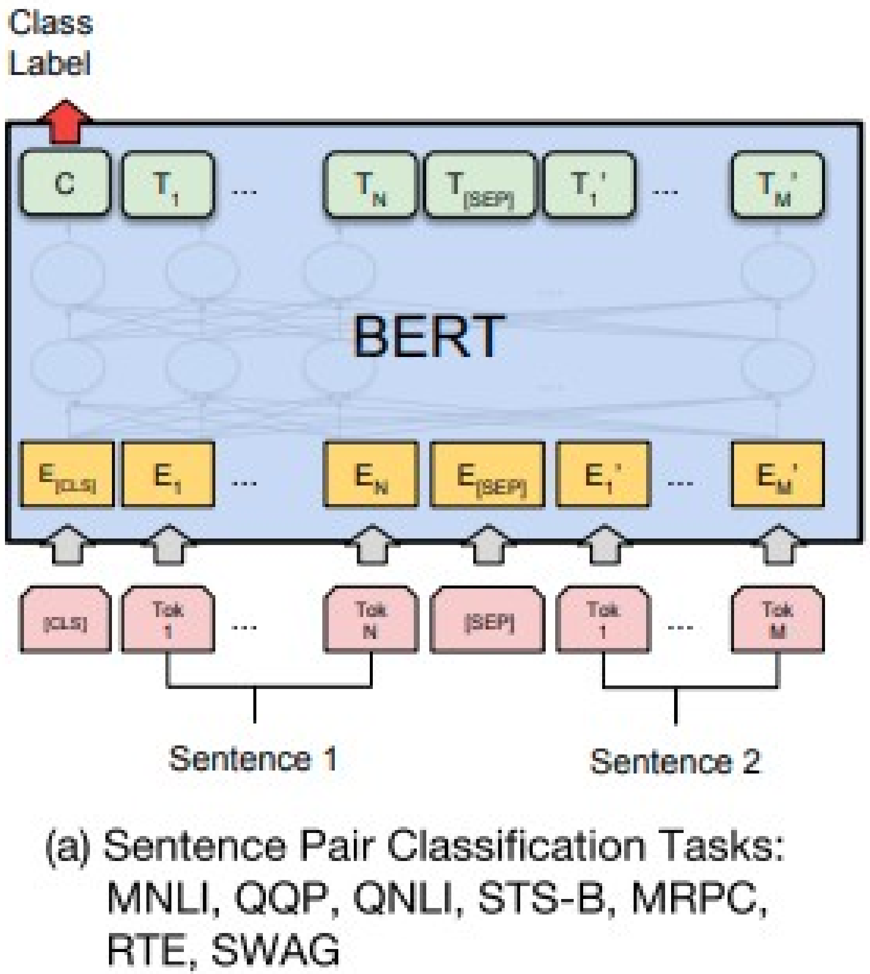
BERT (fine tuning on tasks)



(b) Single Sentence Classification Tasks: SST-2, CoLA

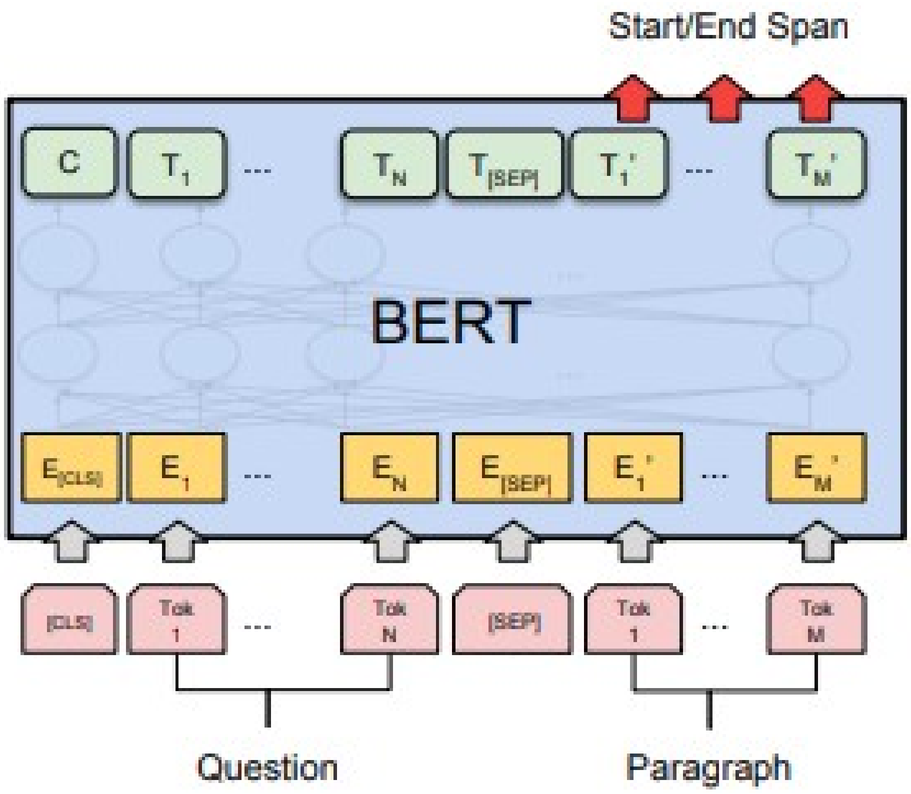
(Devlin et al. 2019)

BERT (fine tuning on tasks)



(Devlin et al. 2019)

BERT (fine tuning on tasks)



(c) Question Answering Tasks:
SQuAD v1.1

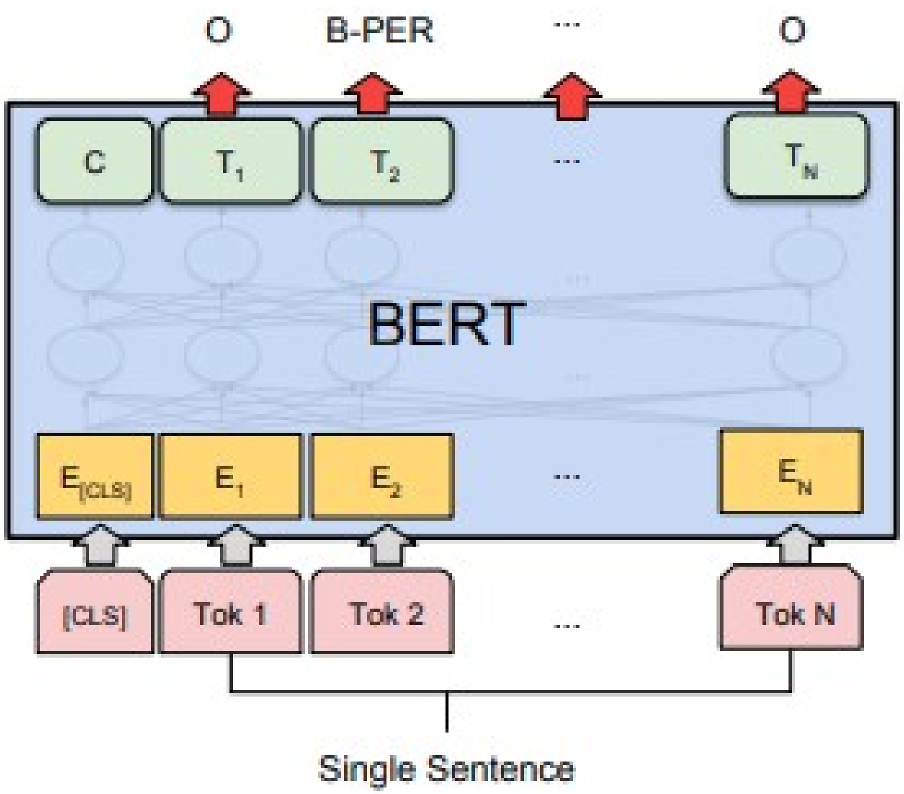
Query: Where is Obama from?

Document: Obama was born in

Hawaii (US).

(Devlin et al. 2019)

BERT (fine tuning on tasks)



(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

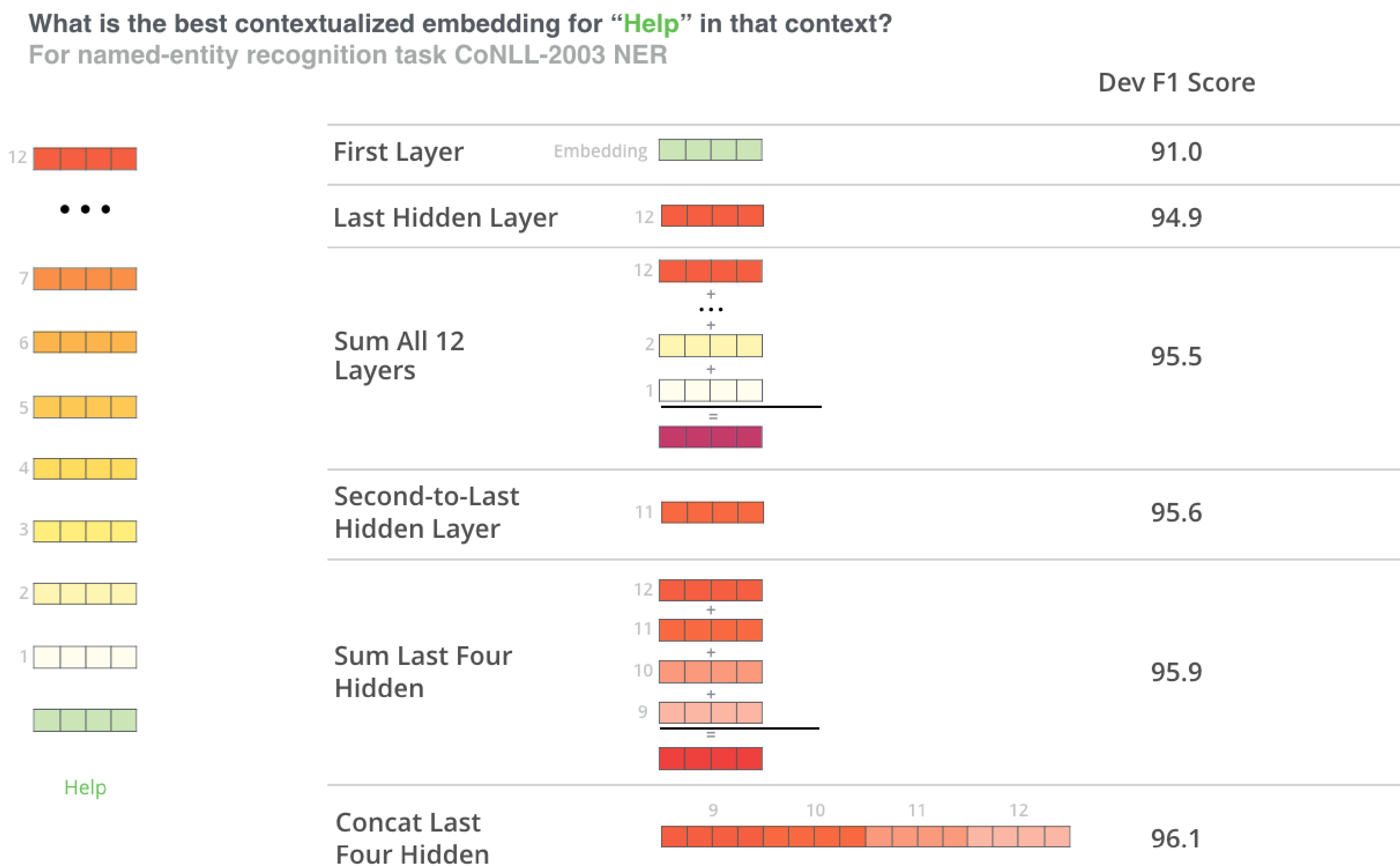
Input: John loves Mary

Output: B-PER O B-PER

Two named-entities of category person

(Devlin et al. 2019)

BERT (feature extraction)



Self-supervised LM pre-training

- Informally, a LM learns some variant of
  $P_{\Theta}(text)$ or $P_{\Theta}(text \mid some\ other\ text)$

- In order to do transfer learning:

  1) Pre-train a (transformer) model as a language model
  2) Fine-tune in target task

     OR extract contextual embeddings

BERT (fine-tuning)

SOTA GLUE benchmark results (sentence pair classification).

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**

(Devlin et al. 2019)

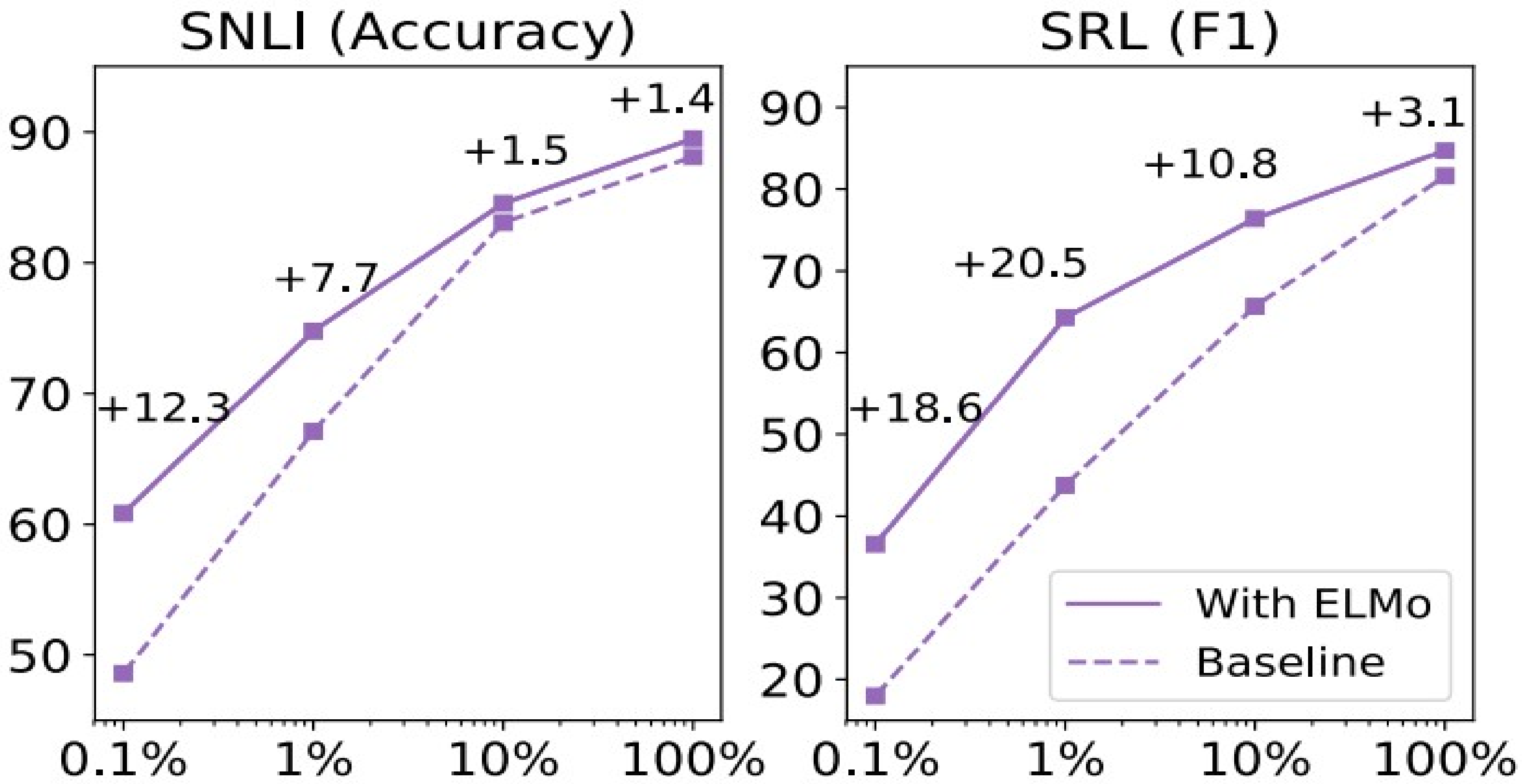
BERT (fine-tuning)

SOTA SQuAD v1.1 (and v2.0) Q&A

System	Dev EM	Dev F1	Test EM	Test F1
Top Leaderboard Systems (Dec 10th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
Published				
BiDAF+ELMo (Single)	-	85.6	-	85.8
R.M. Reader (Ensemble)	81.2	87.9	82.3	88.5
Ours				
$\text{BERT}_{\text{BASE}}$ (Single)	80.8	88.5	-	-
$\text{BERT}_{\text{LARGE}}$ (Single)	84.1	90.9	-	-
$\text{BERT}_{\text{LARGE}}$ (Ensemble)	85.8	91.8	-	-
$\text{BERT}_{\text{LARGE}}$ (Sgl.+TriviaQA)	**84.2**	**91.1**	**85.1**	**91.8**
$\text{BERT}_{\text{LARGE}}$ (Ens.+TriviaQA)	**86.2**	**92.2**	**87.4**	**93.2**

(Devlin et al. 2019)

Pretraining reduces need for annotated data



[(Peters et al, NAACL 2018)]

Scaling up pretraining

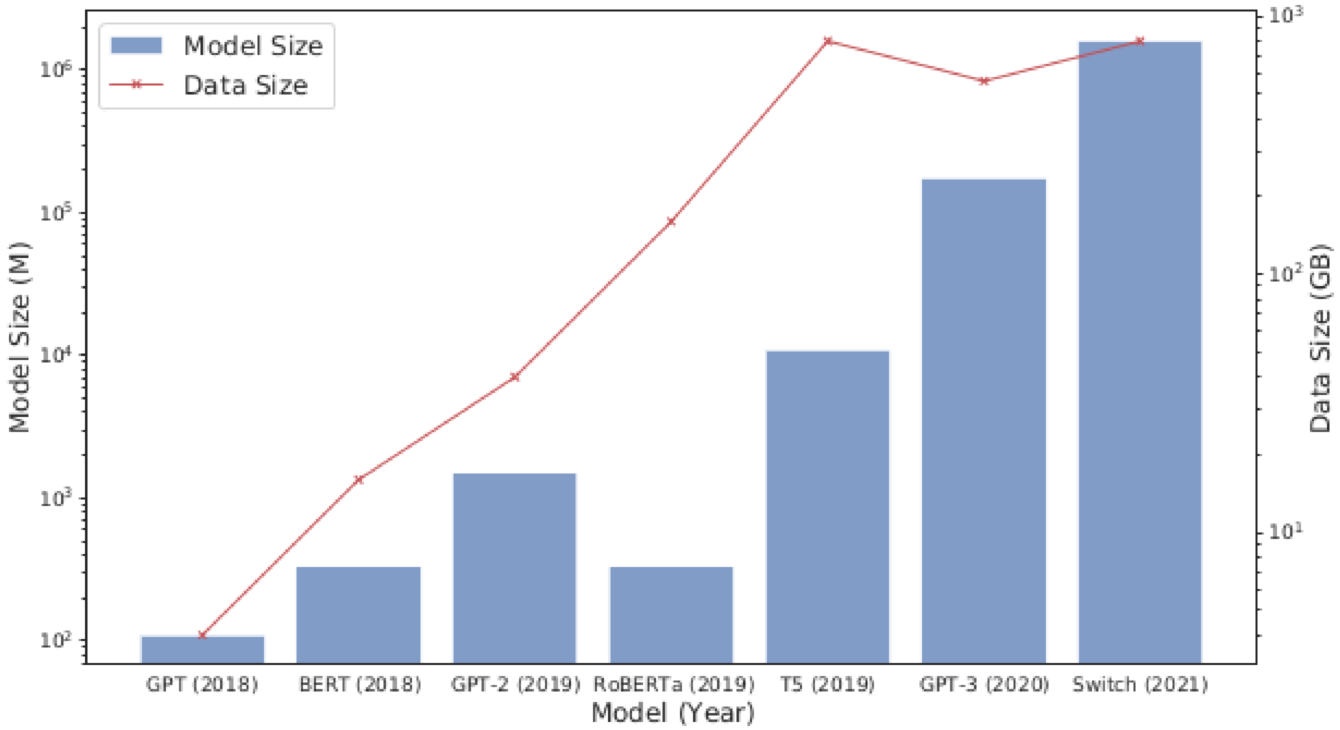
Hyperparams				Dev Set Accuracy		
#L	#H	#A	LM (ppl)	MNLI-m	MRPC	SST-2
3	768	12	5.84	77.9	79.8	88.4
6	768	3	5.24	80.6	82.2	90.7
6	768	12	4.68	81.9	84.8	91.3
12	768	12	3.99	84.4	86.7	92.9
12	1024	16	3.54	85.7	86.9	93.3
24	1024	16	3.23	86.6	87.8	93.7

Table 6: Ablation over BERT model size. #L = the number of layers; #H = hidden size; #A = number of attention heads. "LM (ppl)" is the masked LM perplexity of held-out training data.

Bigger model → better results

(Devlin et al 2019)

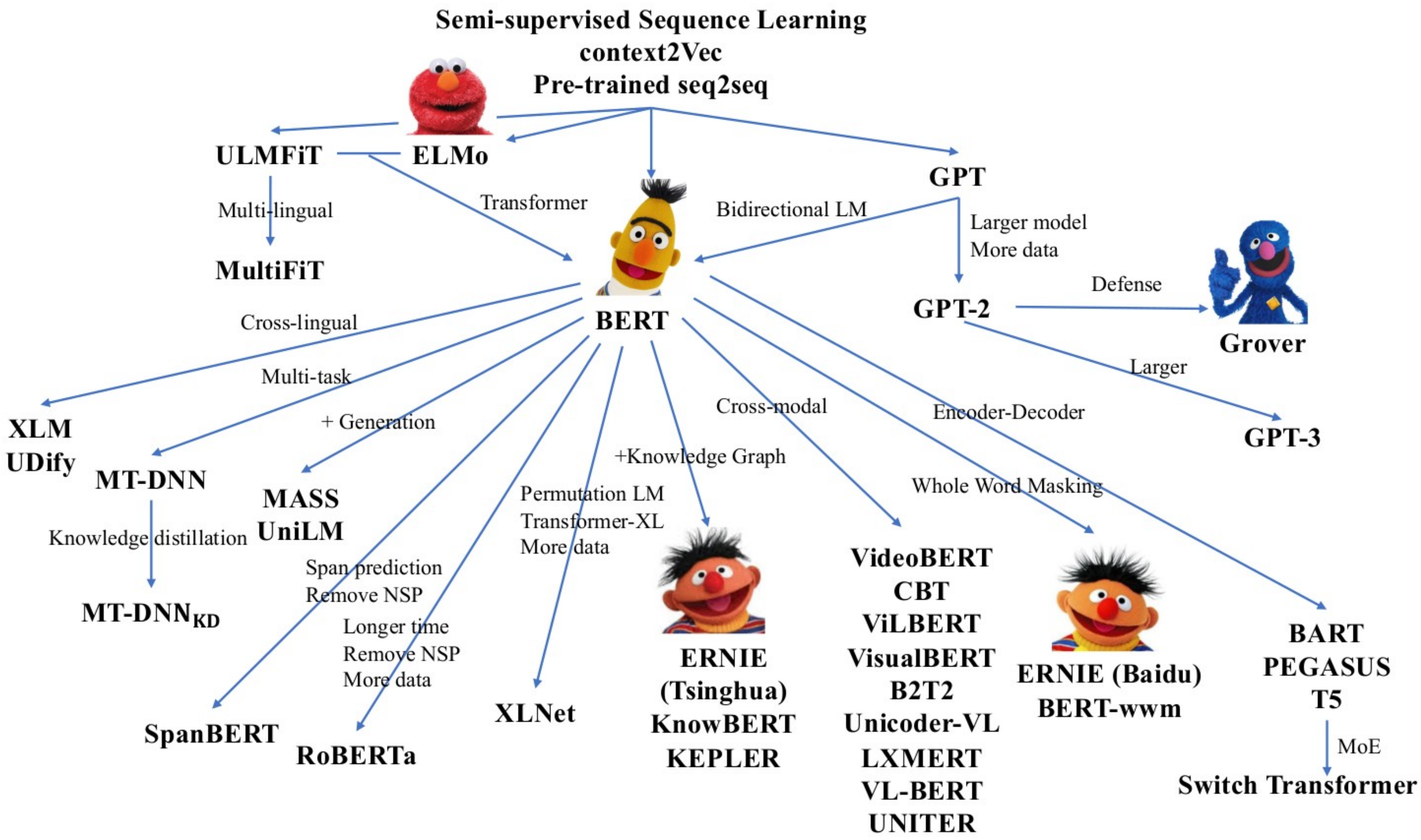
Scaling up pretraining



(b) The model size and data size applied by recent NLP PTMs. A base-10 log scale is used for the figure.

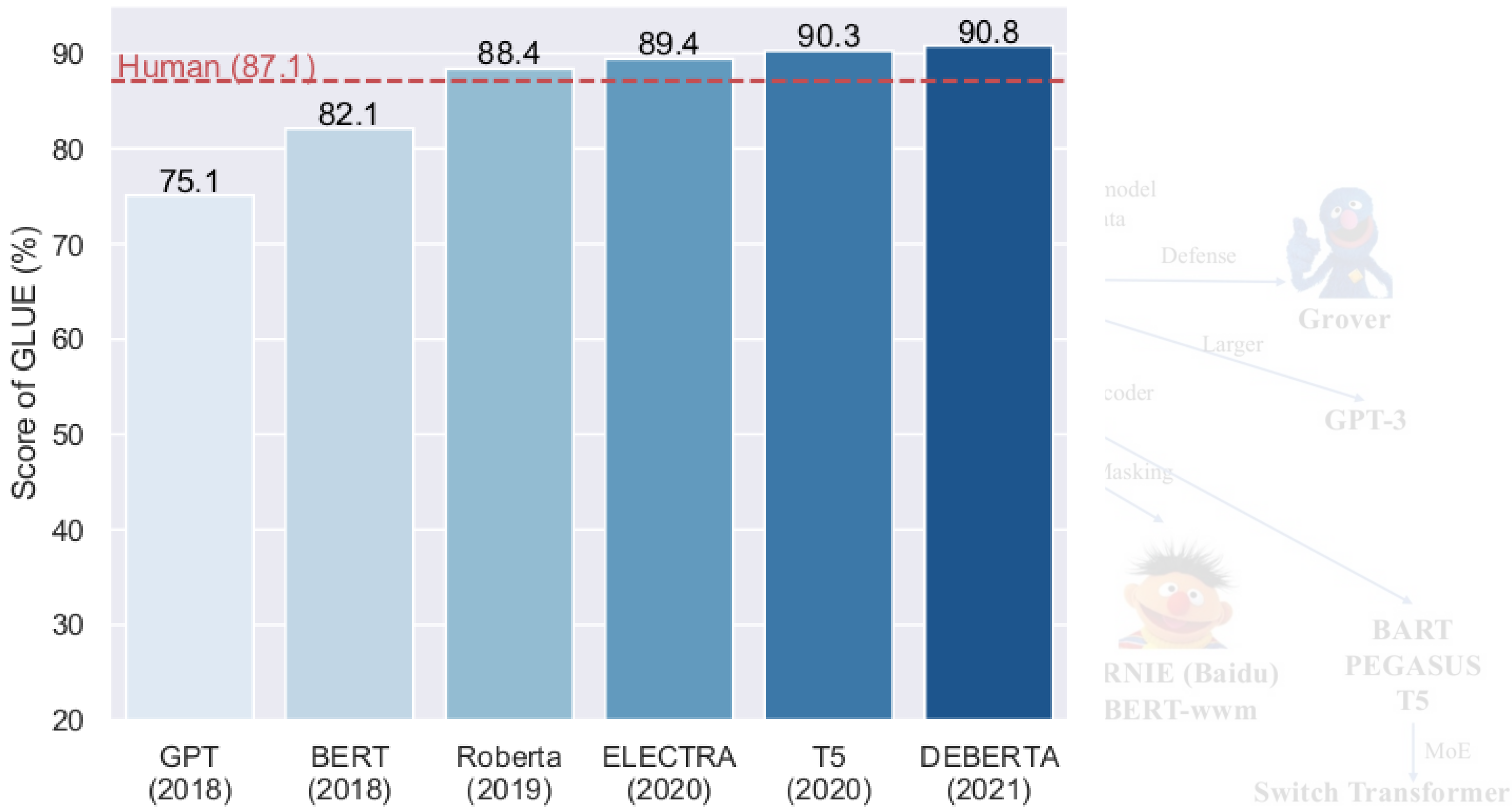
Pre-Trained Models: Past, Present and Future (Han et al. 2021)

BERT's family



Pre-Trained Models: Past, Present and Future (Han et al. 2021)

BERT's family



Pre-Trained Models: Past, Present and Future (Han et al. 2021)

BERT's family: RoBERTa

"Minor" variation of Bert:

(Liu et al. 2019)

- remove next-sentence objective
- larger mini-batches
- larger learning rates
- larger corpus
- dynamic masking

Results on GLUE (dev)

	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
Single-task single models on dev										
$\text{BERT}_{\text{LARGE}}$	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
$\text{XLNet}_{\text{LARGE}}$	89.8/-	93.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa	**90.2/90.2**	**94.7**	**92.2**	**86.6**	**96.4**	**90.9**	**68.0**	**92.4**	**91.3**	-

BERT's family: DeBERTa

Evolution of BERT and RoBERTa: (He et al. 2021)

- Two vectors per token:
  content and (relative) position
- Attention weights in two matrices (disentangled)
- Incorporate absolute position into decoder

Improves over RoBERTa with 50% less corpus.

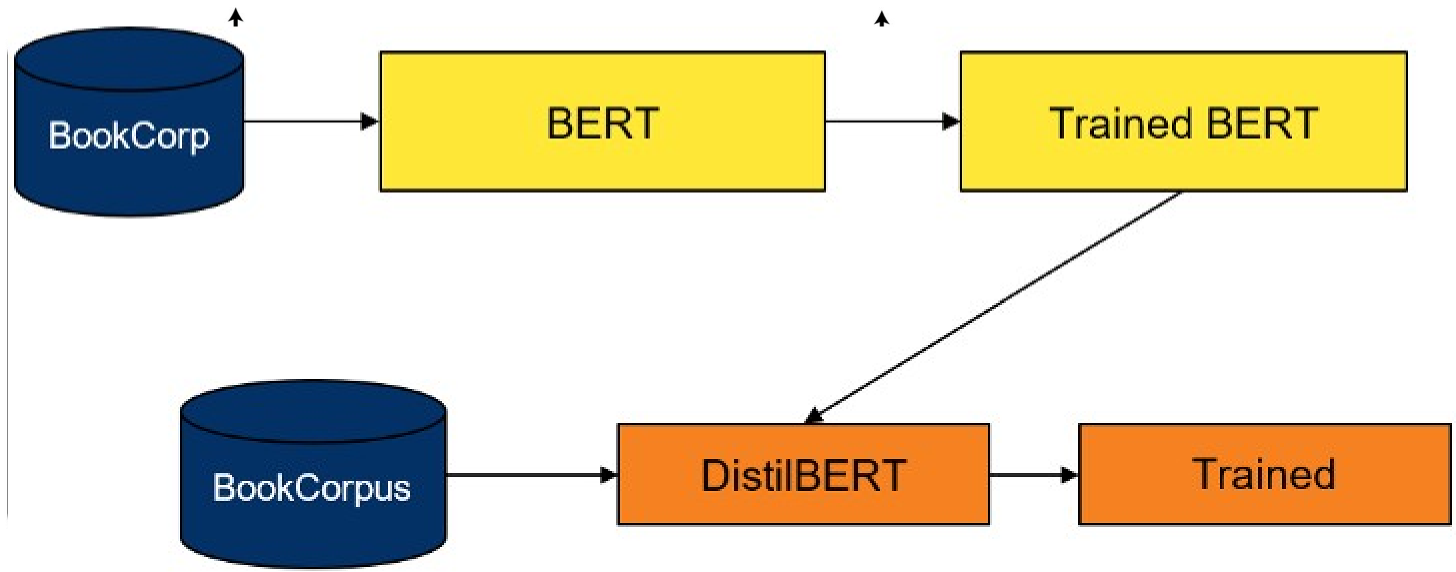
Results on GLUE (dev)

Model	CoLA Mcc	QQP Acc	MNLI-m/mm Acc	SST-2 Acc	STS-B Corr	QNLI Acc	RTE Acc	MRPC Acc	Avg.
BERT_{large}	60.6	91.3	86.6/-	93.2	90.0	92.3	70.4	88.0	84.05
RoBERTa_{large}	68.0	92.2	90.2/90.2	96.4	92.4	93.9	86.6	90.9	88.82
XLNet_{large}	69.0	92.3	90.8/90.8	**97.0**	92.5	94.9	85.9	90.8	89.15
$\text{ELECTRA}_{\text{large}}$	69.1	**92.4**	90.9/-	96.9	92.6	95.0	88.0	90.8	89.46
DeBERTa_{large}	**70.5**	92.3	**91.1/91.1**	96.8	**92.8**	**95.3**	**88.3**	**91.9**	**90.00**

BERT's family: smaller

Models are so large it's costly to fine-tune

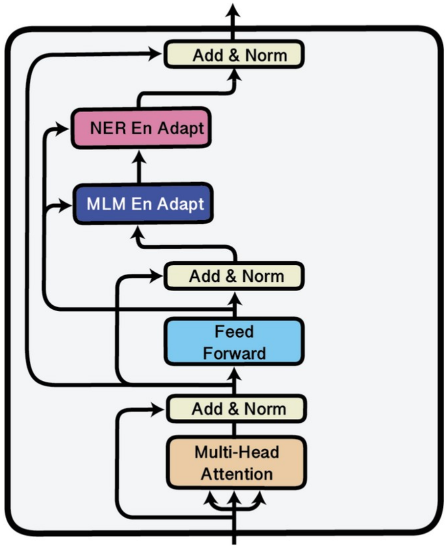
- **Smaller models**: DistilBERT

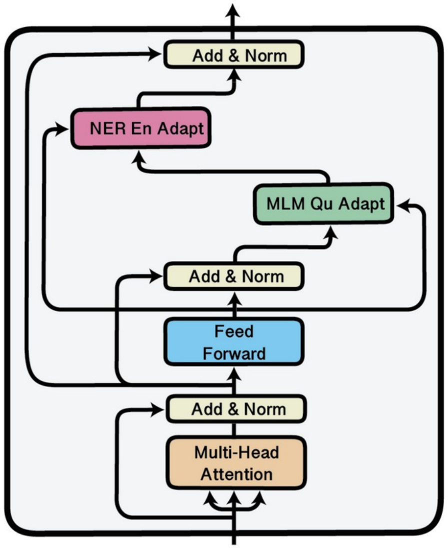


BERT's family: faster fine-tuning

Models are so large it's costly to fine-tune

- **Adapters**: task or language adapters, can be replaced (mix & match)





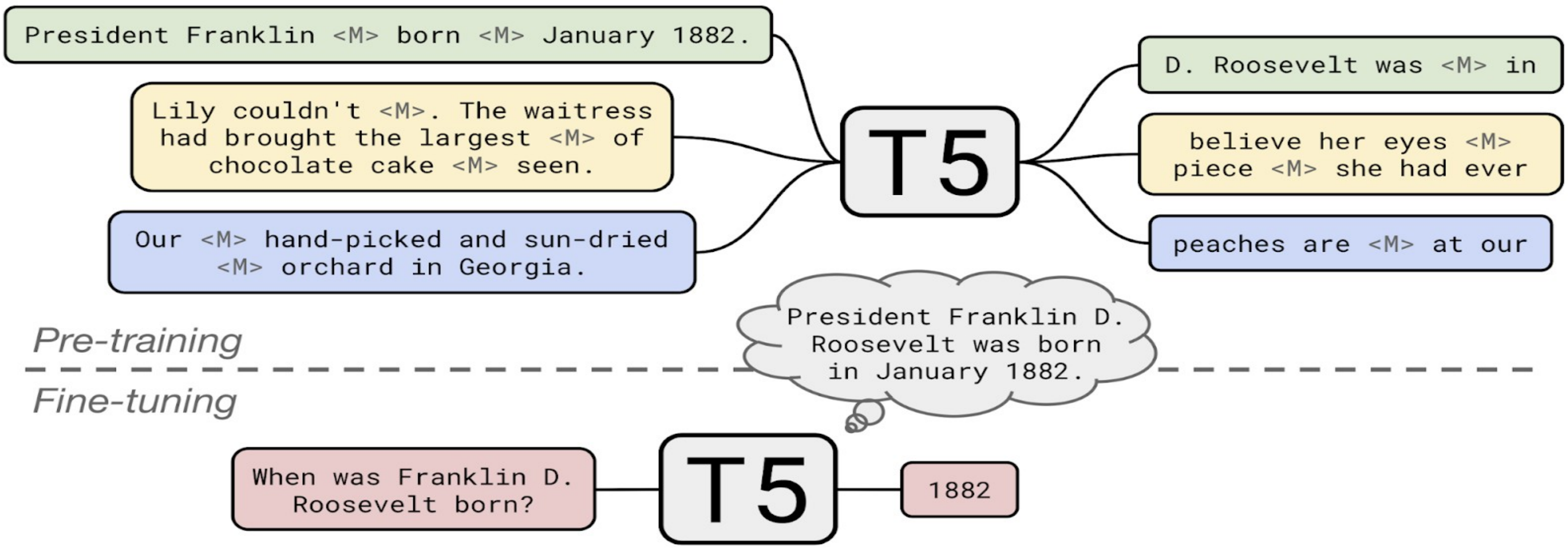
https://ruder.io/recent-advances-lm-fine-tuning/

Other neighbours: seq2seq

Pre-trained sequence to sequence models:

- T5: mask spans in input, generate those spans
- BART: corrupt input, generate input

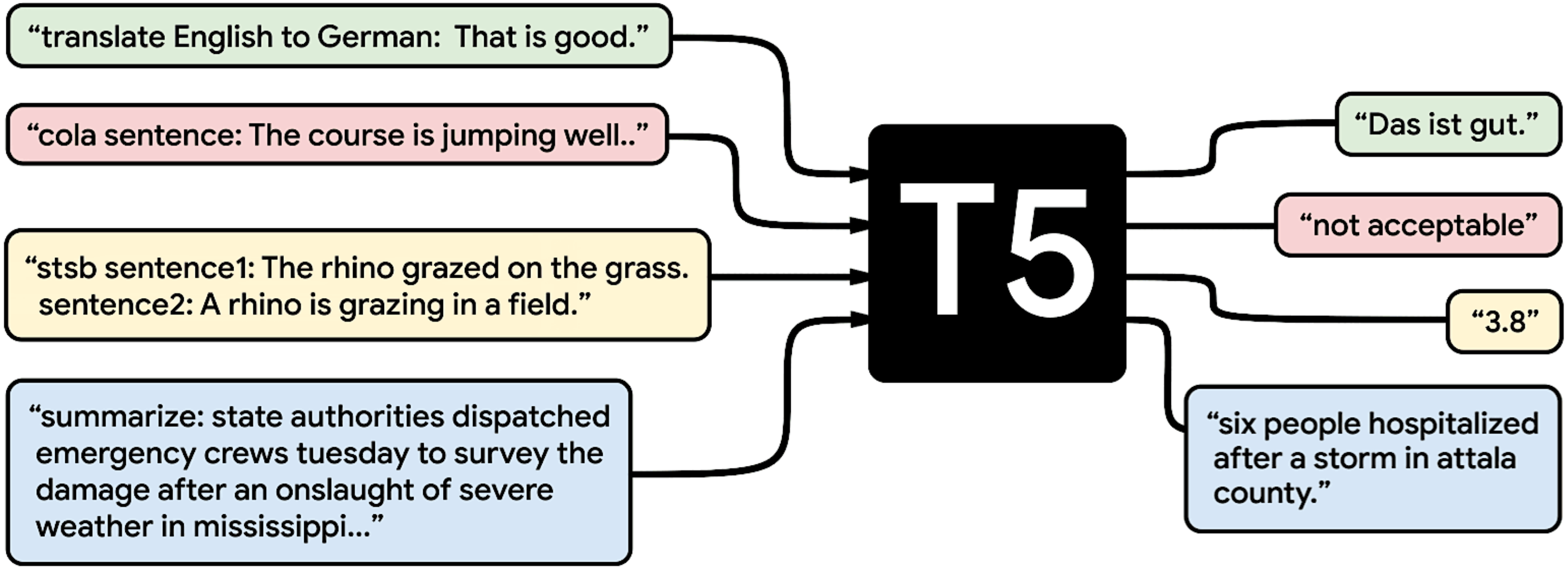
Other neighbours: seq2seq



https://ai.googleblog.com/2020/02/exploring-transfer-learning-with-t5.html

- T5: mask spans in input, generate those spans
- BART: corrupt input, generate input

Other neighbours: seq2seq



https://ai.googleblog.com/2020/02/exploring-transfer-learning-with-t5.html

- T5: mask spans in input, generate those spans
- BART: corrupt input, generate input

Other neighbours: GPT

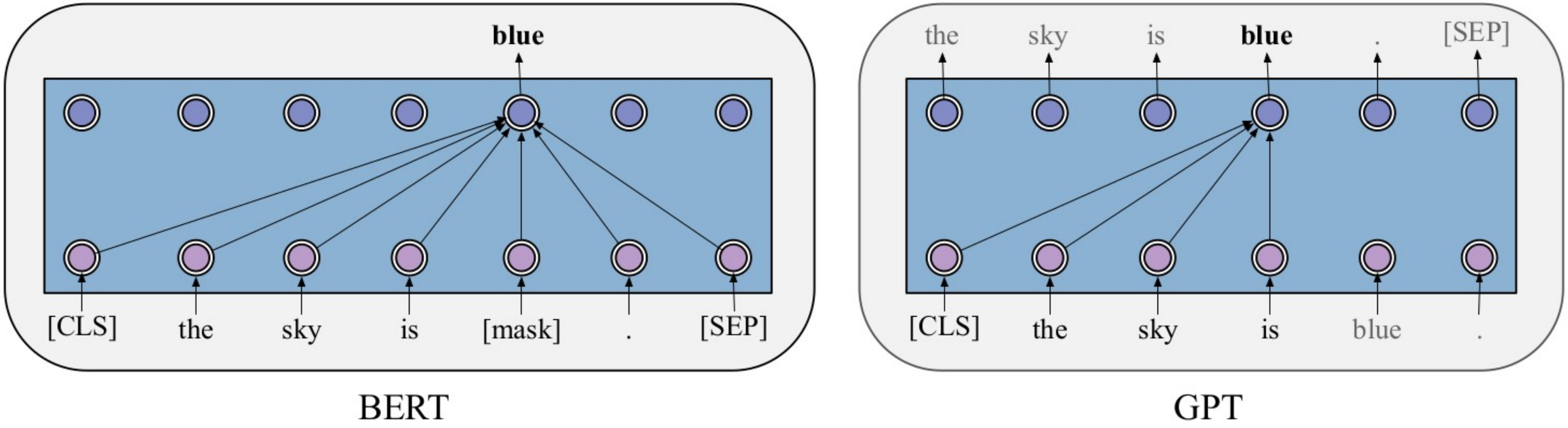
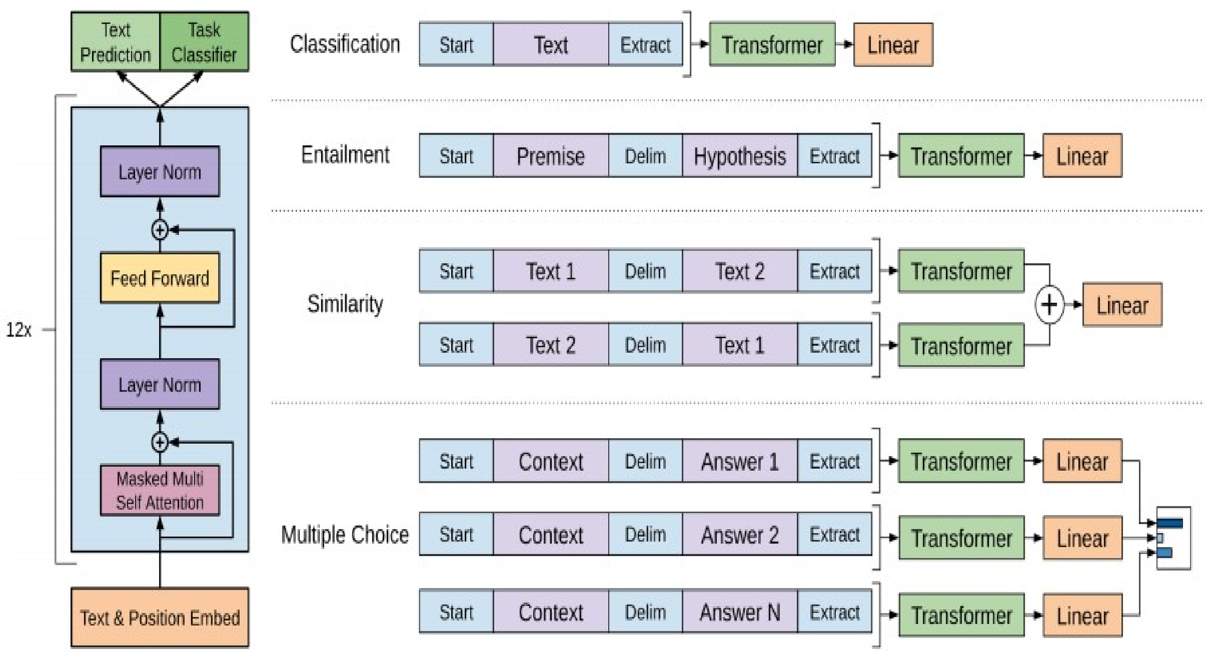


Figure 6: The difference between GPT and BERT in their self-attention mechanisms and pre-training objectives.

Pre-Trained Models: Past, Present and Future (Han et al. 2021)

Fine-tuning GPT



Method	MNLI-m	MNLI-mm	SNLI	SciTail	QNLI	RTE
ESIM + ELMo [44] (5x)	-	-	89.3	-	-	-
CAFE [58] (5x)	80.2	79.0	89.3	-	-	-
Stochastic Answer Network [35] (3x)	80.6	80.1	-	-	-	-
CAFE [58]	78.7	77.9	88.5	83.3		
GenSen [64]	71.4	71.3	-	-	82.3	59.2
Multi-task BiLSTM + Attn [64]	72.2	72.1	-	-	82.1	**61.7**
Finetuned Transformer LM (ours)	**82.1**	**81.4**	**89.9**	**88.3**	**88.1**	56.0

Pretrain large 12-layer **left-to-right** Transformer,

100M model (GPT-3 370B)

Can be used to generate text, code, images

Fine-tune for sentence, sentence pair and multiple choice questions.

SOTA results before BERT

(Radford et al., 2018)

Alternatives to fine-tuning: prompts

GPT-3 exhibits amazing capabilities,
but it is so large (370B) it's costly to fine-tune

- **Prompts** for zero-shot learning

Task	Template	Label words
SST-2	$<S_1>$ It was [MASK] .	positive: great, negative: terrible
SST-5	$<S_1>$ It was [MASK] .	v.positive: great, positive: good, neutral: okay, negative: bad, v.negative: terrible
MR	$<S_1>$ It was [MASK] .	positive: great, negative: terrible
CR	$<S_1>$ It was [MASK] .	positive: great, negative: terrible
Subj	$<S_1>$ This is [MASK] .	subjective: subjective, objective: objective
TREC	[MASK] : $<S_1>$	abbreviation: Expression, entity: Entity, description: Description human: Human, location: Location, numeric: Number
COLA	$<S_1>$ This is [MASK] .	grammatical: correct, not_grammatical: incorrect
MNLI	$<S_1>$? [MASK] , $<S_2>$	entailment: Yes, netural: Maybe, contradiction: No
SNLI	$<S_1>$? [MASK] , $<S_2>$	entailment: Yes, netural: Maybe, contradiction: No
QNLI	$<S_1>$? [MASK] , $<S_2>$	entailment: Yes, not_entailment: No
RTE	$<S_1>$? [MASK] , $<S_2>$	entailment: Yes, not_entailment: No
MRPC	$<S_1>$ [MASK] , $<S_2>$	equivalent: Yes, not_equivalent: No
QQP	$<S_1>$ [MASK] , $<S_2>$	equivalent: Yes, not_equivalent: No
STS-B	$<S_1>$ [MASK] , $<S_2>$	y_u: Yes, y_l: No

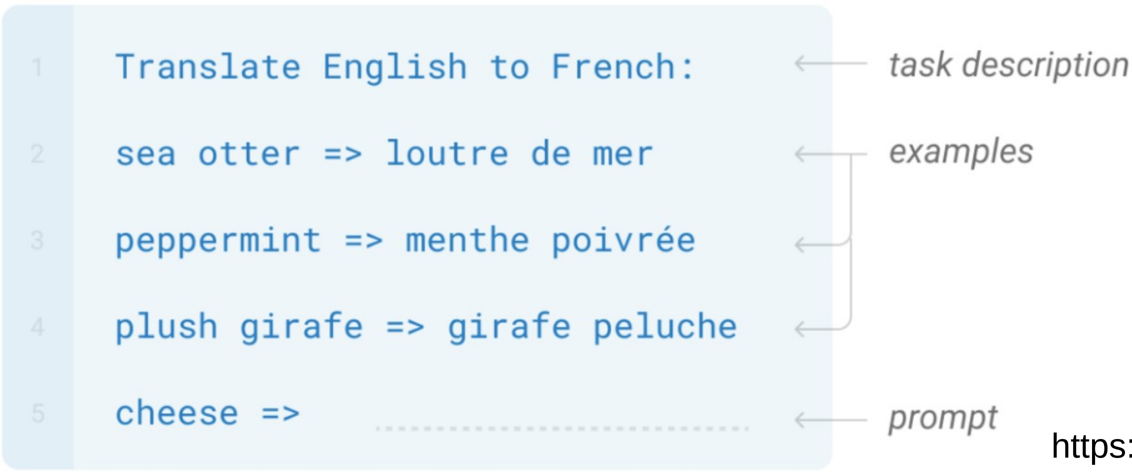
https://thegradient.pub/prompting/

Alternatives to fine-tuning: ICL

GPT-3 exhibits amazing capabilities,
but it is so large (370B) it's costly to fine-tune

- **In-context learning** for few-shot learning

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.



https://thegradient.pub/prompting/

Why does language modelling work so well?

- Language modelling is a very difficult task, even for humans.
- Language models are expected to compress any possible context into a vector that generalizes over possible completions.
  - “They walked down the street to ???”
- To have any chance at solving this task, a model is forced to learn syntax, semantics, encode facts about the world, etc.
- Given enough data, a huge model, and enough compute, can do a reasonable job!
- Empirically works better than translation, autoencoding: “Language Modelling Teaches You More Syntax than Translation Does” (Zhang et al. 2018)

Pretrained transformers in tf

- https://huggingface.co/transformers/ (code and models)

```
from transformers import TFBertForSequenceClassification, BertTokenizer

model = TFBertForSequenceClassification.from_pretrained("bert-base-cased")
tokenizer = BertTokenizer.from_pretrained("bert-base-cased")
```

Plan for this session

- Transfer learning:
  - Pre-trained LM, BERT, GPT
  - **Pre-trained multilingual LM**
  - Prompting
- Deep learning frameworks
- Last words

How to build a NLP application for language L

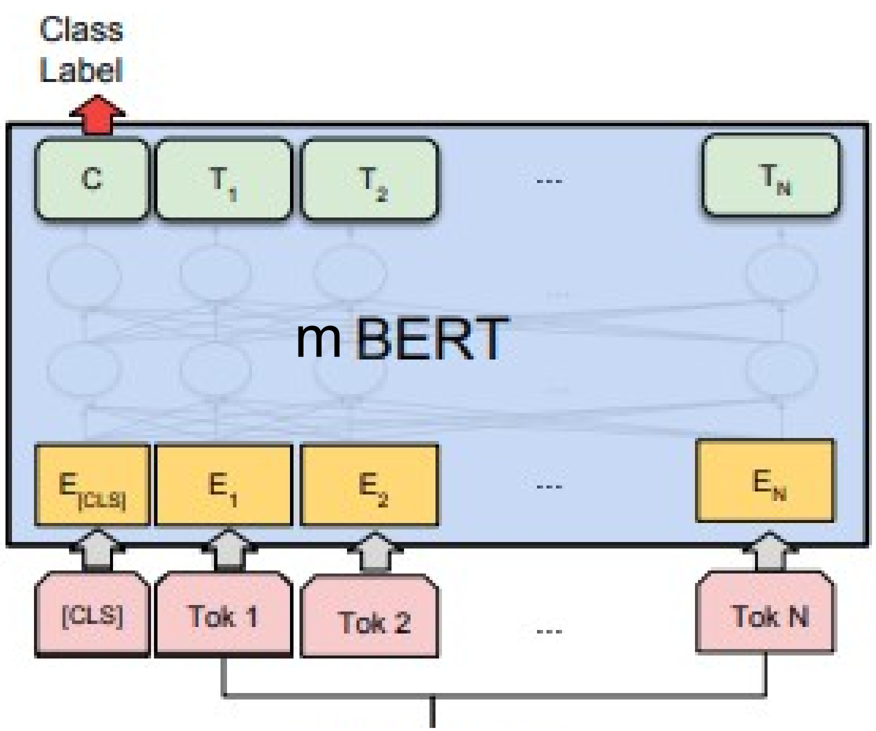
- Fine-tuning provides best results:
  - Assumes a **pre-trained LM** in your language
  - Assumes **labeled data** in your language
- If you **don't have a pre-trained LM**
  - Use multilingual pre-trained LM which includes your language, fine-tune
- If you **don't have labeled data (zero-shot!)**:
  - Use a multilingual pre-trained LM which includes your language
  - Fine-tune it with the labeled data in another language (English)
  - Test in your language
- If you **only have very small labeled data**:
  - Use a multilingual pre-trained LM which includes your language
  - Fine-tune it with the labeled data in another language (English)
  - Fine-tune it with the small labelled data in your language
  - Test in your language

Multilingual Pretraining

- Key idea: **Share vocabulary** and representations across languages by training one model on many languages.
  - Multilingual BERT: BERT trained jointly on 100 languages
  - **XLM-Roberta** (XLM-R, Lample et al. 2019) 100 languages
  - mT5, mBART, … (no mGPT yet)
- **It represents languages in common space out-of-the-box!!**
- Advantages:
  - Easy to implement, enables cross-lingual pretraining by itself
  - Widely used with languages for which monolingual BERTs are not available
  - Fine-tune on one language (e.g. English), test on another (e.g. Basque)
- Disadvantages: Under-representation of low-resource languages
  - Give your Text Representation Models some Love: the Case for Basque (Agerri et al. 2020)

Is it worth having a monolingual LM

Behagune tweet dataset



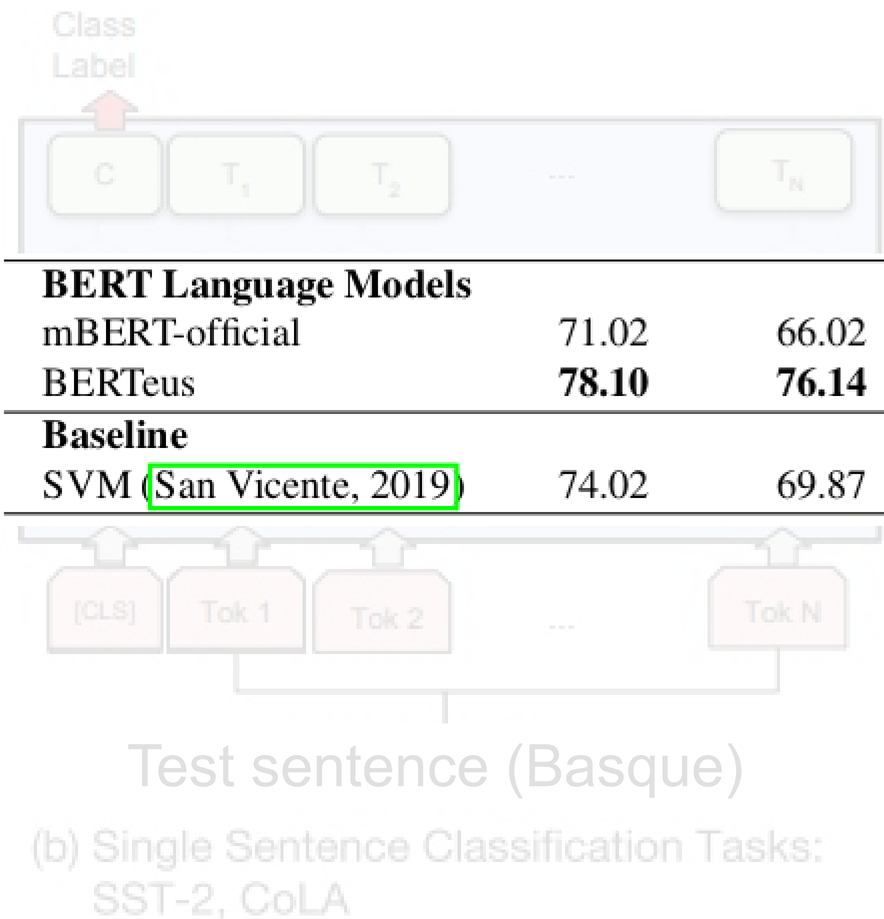
Sentence (Basque)

(b) Single Sentence Classification Tasks:
SST-2, CoLA

Give your Text Representation Models some Love: the Case for Basque
(Agerri et al. 2020)

Is it worth having a monolingual LM

Behagune tweet dataset



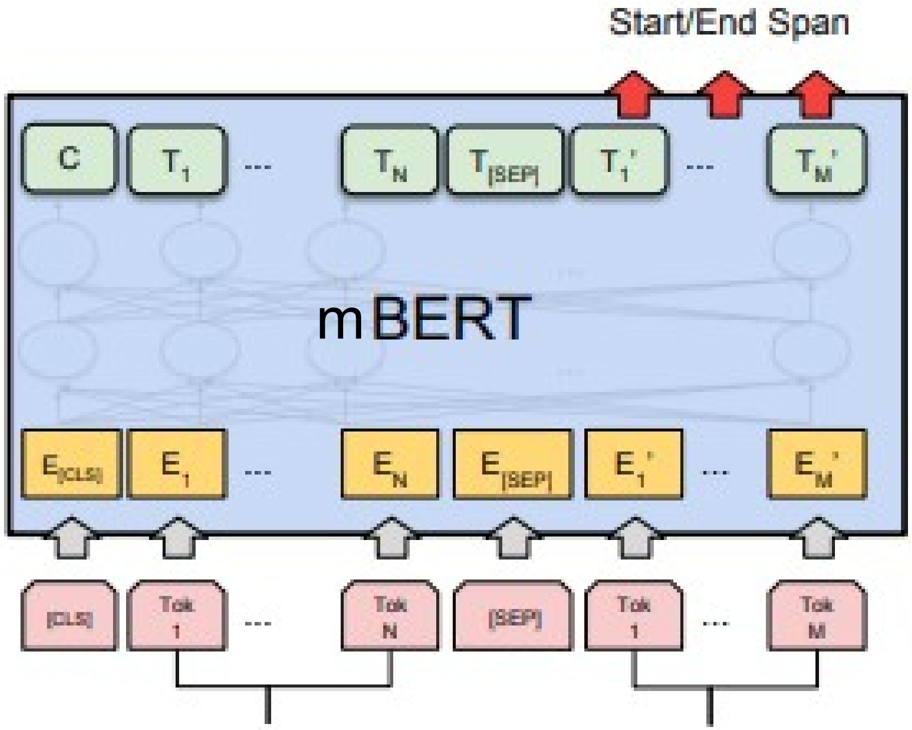
BERT Language Models		
mBERT-official	71.02	66.02
BERTeus	**78.10**	**76.14**
Baseline		
SVM (San Vicente, 2019)	74.02	69.87

YES!

Give your Text Representation Models some Love: the Case for Basque
(Agerri et al. 2020)

Pre-train on 100 languages
Fine-tune on QA (English)

SQUAD
Q&A
dataset

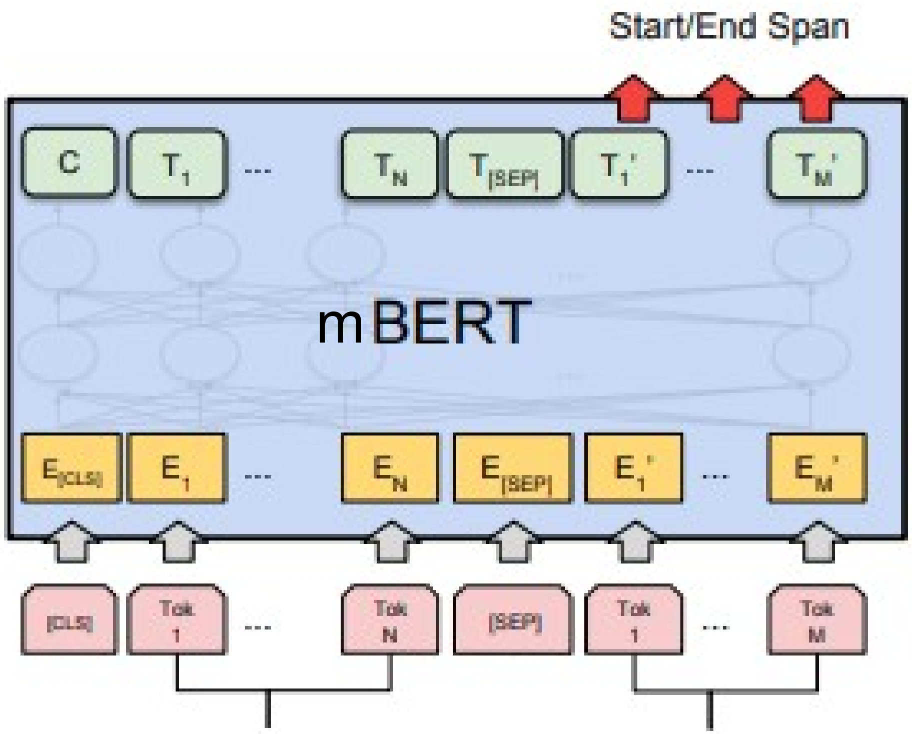


Training question and paragraph (English)

(Devlin et al. 2019)

Test QA (Basque - zero-shot)

Elkarhizketak
QA dataset

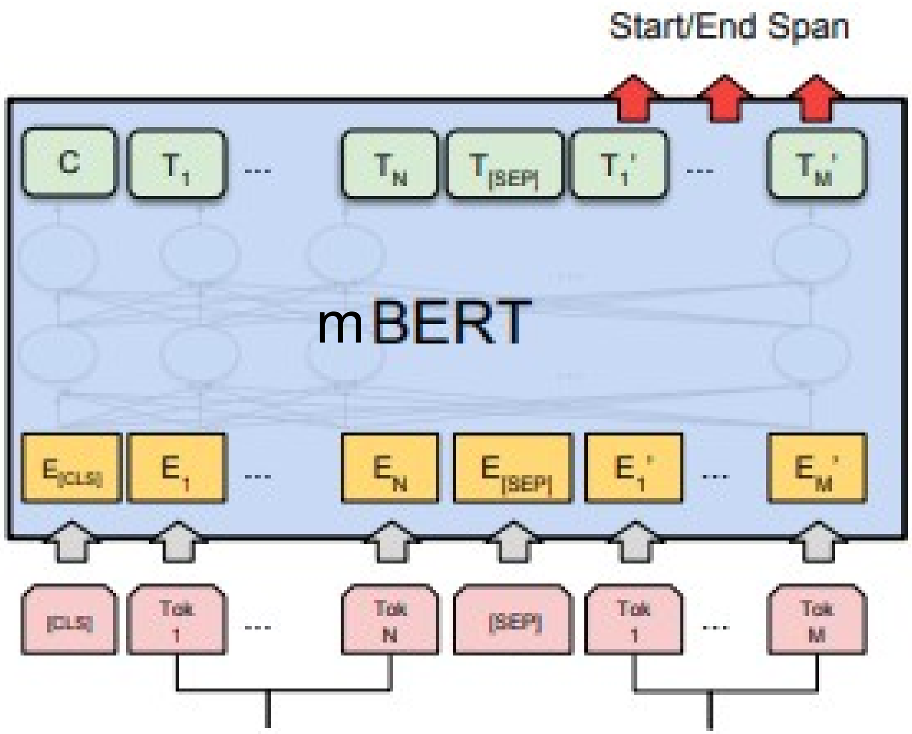


Test question and paragraph (Basque)

(Devlin et al. 2019)

Further fine-tune in Basque QA, then test on Basque QA

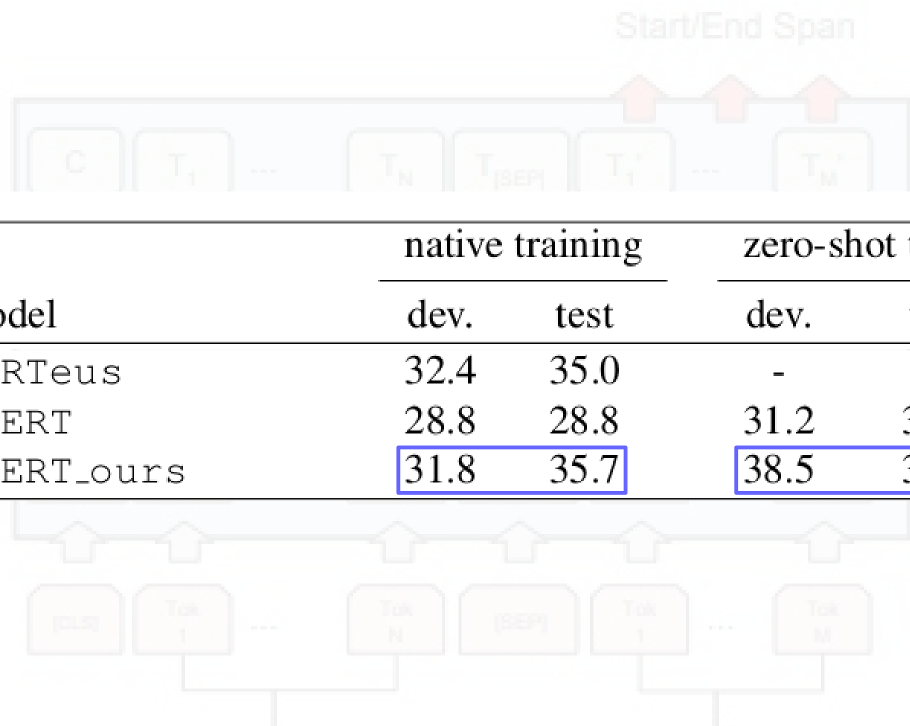
Elkarhizketak
QA dataset



Training question and paragraph (Basque)

(Devlin et al. 2019)

Further fine-tune in Basque QA, then test on Basque QA



Elkarhizketa QA dataset

Model	native training		zero-shot transf.		low resource transf.	
	dev.	test	dev.	test	dev.	test
BERTeus	32.4	35.0	-	-	-	-
mBERT	28.8	28.8	31.2	31.5	37.0	37.4
mBERT_ours	31.8	35.7	38.5	38.9	42.7	**41.2**

(Devlin et al. 2019)

Plan for this session

- Transfer learning:
  - Pre-trained LM, BERT, GPT
  - Pre-trained multilingual LM
  - **Prompting**
- Deep learning frameworks
- Last words

CS11-711 Advanced NLP

Prompting

(+ Encoder-Decoder Pre-training)

Graham Neubig

Carnegie Mellon University

Language Technologies Institute

Site

https://phontron.com/class/anlp2021/

Most Slides by Pengfei Liu

Recommended Reading:

Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing

Pengfei Liu
Carnegie Mellon University
pliu3@cs.cmu.edu

Weizhe Yuan
Carnegie Mellon University
weizhey@cs.cmu.edu

Jinlan Fu
National University of Singapore
jinlanjonna@gmail.com

Zhengbao Jiang
Carnegie Mellon University
zhengbaj@cs.cmu.edu

Hiroaki Hayashi
Carnegie Mellon University
hiroakih@cs.cmu.edu

Graham Neubig
Carnegie Mellon University
gneubig@cs.cmu.edu

Prompt engineering

Four Paradigms of NLP Technical Development

- Feature Engineering
- Architecture Engineering
- Objective Engineering
- Prompt Engineering

Feature Engineering

- **Paradigm**: Fully Supervised Learning (Non-neural Network)
- **Time Period:** Most popular through 2015
- **Characteristics**:
  - Non-neural machine learning models mainly used
  - Require manually defined feature extraction
- **Representative Work**:
  - Manual features -> linear or kernelized support vector machine (SVM)
  - Manual features -> conditional random fields (CRF)

Prompt engineering

Architecture Engineering

- **Paradigm**: Fully Supervised Learning (Neural Networks)
- **Time Period**: About 2013-2018
- **Characteristics**:
  - Rely on neural networks
  - Do not need to manually define features, but should modify the network structure (e.g.: LSTM v.s CNN)
  - Sometimes used pre-training of LMs, but often only for shallow features such as embeddings
- **Representative Work**:
  - CNN for Text Classification

Prompt engineering

Objective Engineering

- **Paradigm**: Pre-train, Fine-tune
- **Time Period**: 2017-Now
- **Characteristics**:
  - Pre-trained LMs (PLMs) used as initialization of full model - both shallow and deep features
  - Less work on architecture design, but engineer objective functions
- **Typical Work**:
  - BERT → Fine Tuning

Prompt engineering

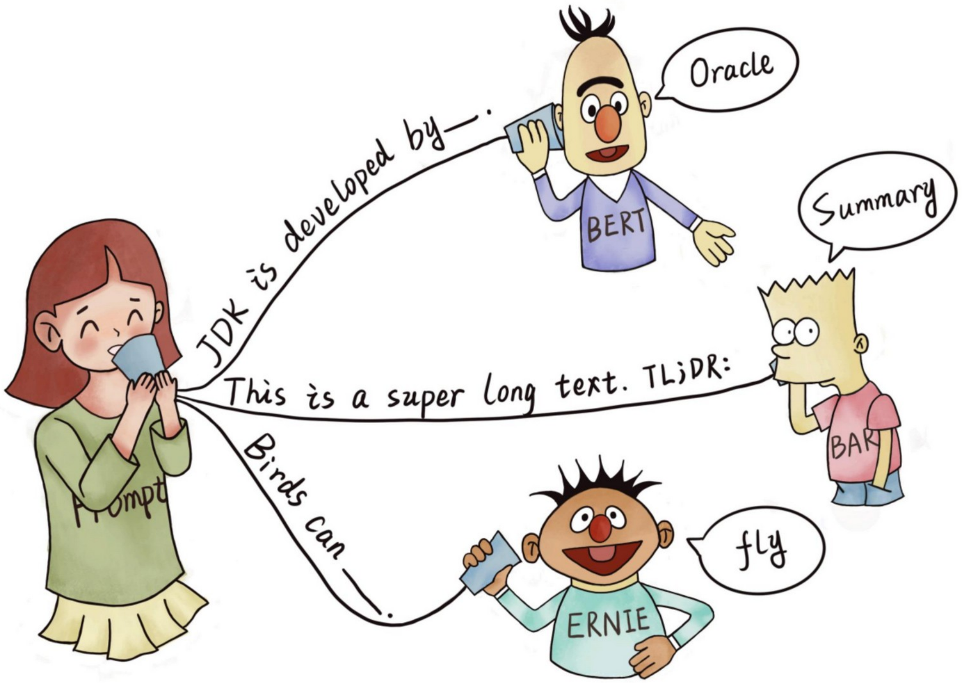
Prompt Engineering

- **Paradigm**: Pre-train, Prompt, Predict
- **Date**: 2019-Now
- **Characteristic**:
  - NLP tasks are modeled entirely by relying on LMs
  - The tasks of shallow and deep feature extraction, and prediction of the data are all given to the LM
  - Engineering of prompts is required
- **Representative Work:**
  - GPT3

Prompt engineering

What is Prompting?

- Encouraging a pre-trained model to make particular predictions by providing a "prompt" specifying the task to be done.



Prompt engineering

What is the general workflow of Prompting?

- Prompt Addition
- Answer Prediction
- Answer-Label Mapping

Prompt engineering

Prompt Addition

- **Prompt Addition**: Given input x, we transform it into prompt x’ through two steps:
  - Define a template with two slots, one for input [x], and one for the answer [z]
  - Fill in the input slot [x]

Prompt engineering

Example: Sentiment Classification

Input: x = "I love this movie"

⇩

Template: [x] Overall, it was a [z] movie

⇩

Prompting: x' = "I love this movie. Overall it was a [z] movie."

Prompt engineering

Answer Prediction

- Answer Prediction: Given a prompt, predict the answer [z]
  - Fill in [z]

Prompt engineering

Example

Input: x = “I love this movie”

⇩

Template: [x] Overall, it was a [z] movie

⇩

Prompting: x’ = “I love this movie. Overall it was a [z] movie.”

⇩

Predicting: x’ = “I love this movie. Overall it was a fantastic movie.”

Prompt engineering

Mapping

- Mapping: Given an answer, map it into a class label

Prompt engineering

Example

Input: x = "I love this movie"

⇩

Template: [x] Overall, it was a [z] movie

⇩

Prompting: x' = "I love this movie. Overall it was a [z] movie."

⇩

Predicting: x' = "I love this movie. Overall it was a fantastic movie."

⇩

Mapping: fantastic => Positive

Prompt engineering

Design Considerations for Prompting

- Pre-trained Model Choice
- Prompt Engineering
- Answer Engineering
- Expanding the Paradigm
- Prompt-based Training Strategies

Prompt engineering

Pre-trained Language Models

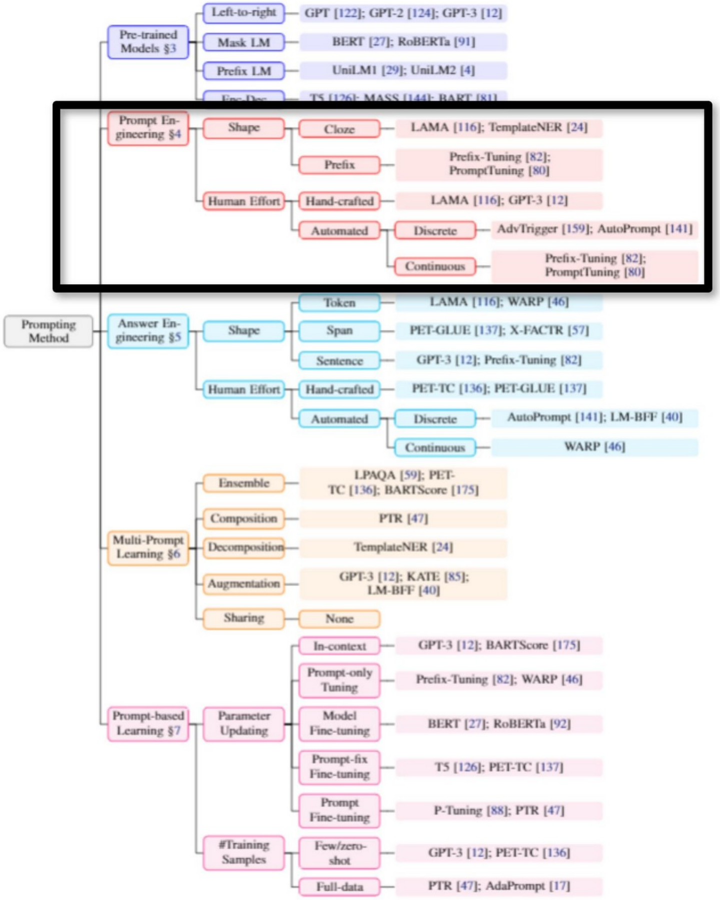
Popular Frameworks

▪ Left-to-Right LM	GPT-3
▪ Masked LM	BERT
▪ Encoder-decoder	BART, T5

Prompt engineering

Design Considerations for Prompting

- Pre-trained Model Choice
- Prompt Engineering
- Answer Engineering
- Expanding the Paradigm
- Prompt-based Training Strategies



Prompt engineering

Traditional Formulation V.S Prompt Formulation

Traditional Formulation

Input: x = "I love this movie"

⇩

Predicting: y = Positive

Prompt Formulation

Input: x = "I love this movie"

⇩

Template: [x] Overall, it was a [z] movie

⇩

Prompting: x' = "I love this movie. Overall it was a [z] movie."

⇩

Predicting: x' = "I love this movie. Overall it was a fantastic movie."

⇩

Mapping (answer -> label):
fantastic => Positive

Prompt engineering

Traditional Formulation V.S Prompt Formulation

Traditional Formulation

Input: x = "I love this movie"

⇩

Predicting: y = Positive

Prompt Formulation

Input: x = "I love this movie"

⇩

Template: [x] Overall, it was a [z] movie

⇩

Prompting: x' = "I love this movie. Overall it was a [z] movie."

⇩

Predicting: x' = "I love this movie. Overall it was a fantastic movie."

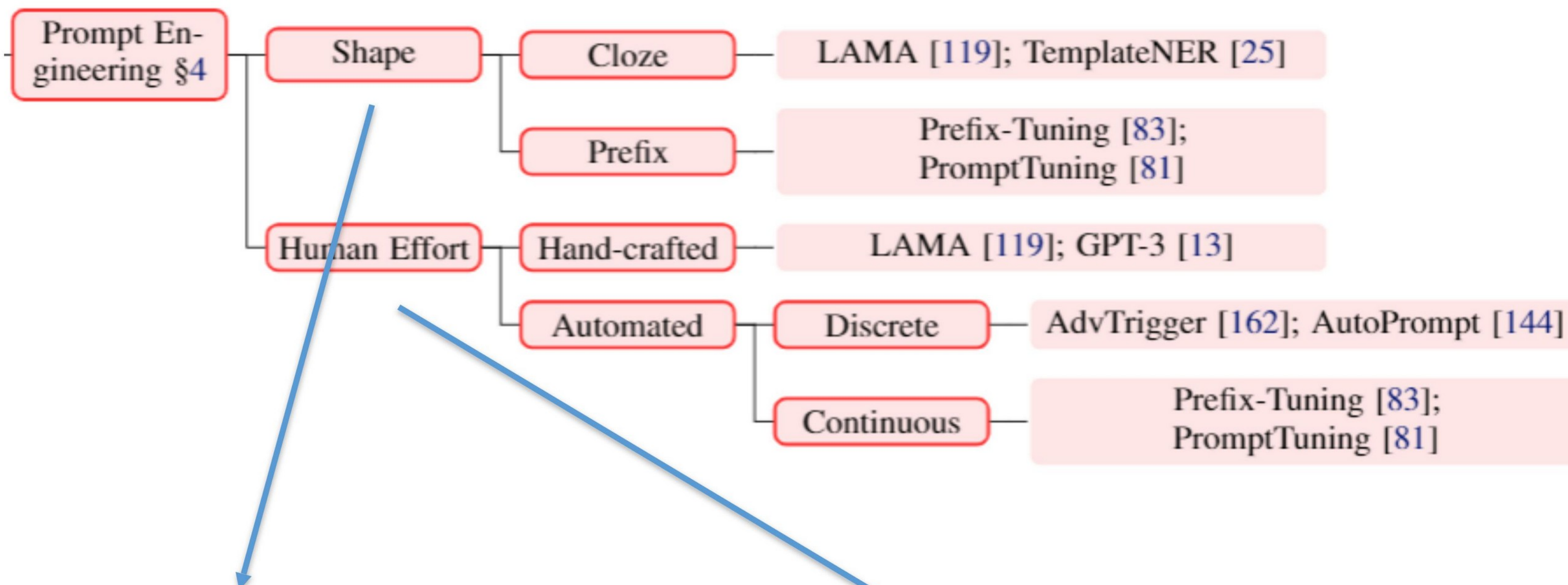
⇩

Mapping (answer -> label):
fantastic => Positive

How to define a suitable prompt template?

Prompt engineering

Prompt Template Engineering



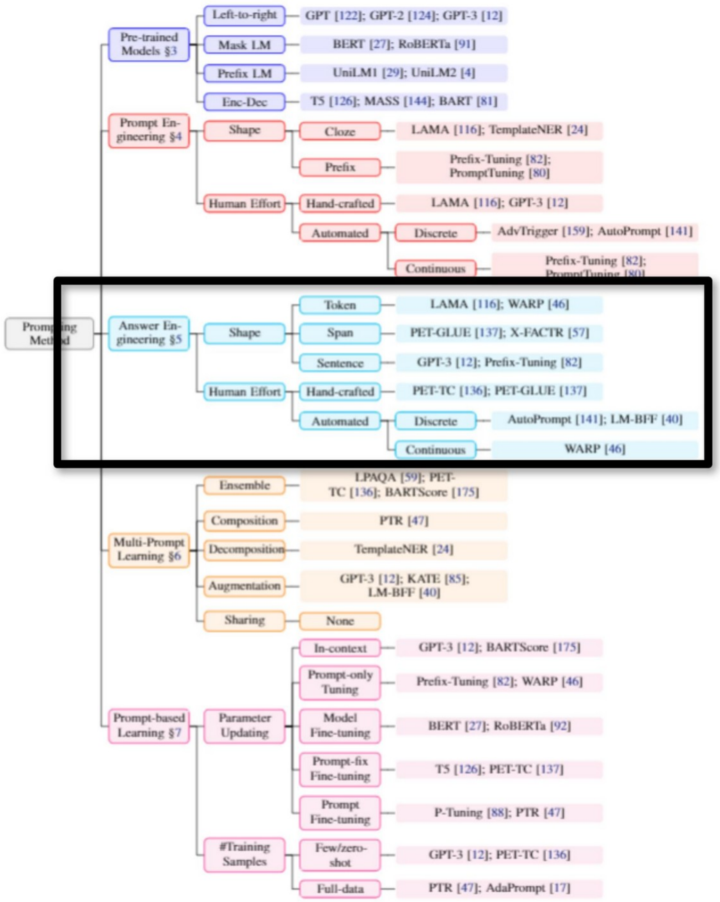
How to define the shape of a prompt template?

How to search for appropriate prompt templates?

Prompt engineering

Design Considerations for Prompting

- Pre-trained Model Choice
- Prompt Template Engineering
- Answer Engineering
- Expanding the Paradigm
- Prompt-based Training Strategies



Prompt engineering

Answer Engineering

- Why do we need answer engineering?
  - We have reformulate the task! We also should re-define the "ground truth labels"

Prompt engineering

Traditional Formulation V.S Prompt Formulation

Traditional Formulation

Input: x = "I love this movie"

⇩

Predicting: y = Positive

Prompt Formulation

Input: x = "I love this movie"

⇩

Template: [x] Overall, it was a [z] movie

⇩

Prompting: x' = "I love this movie. Overall it was a [z] movie."

⇩

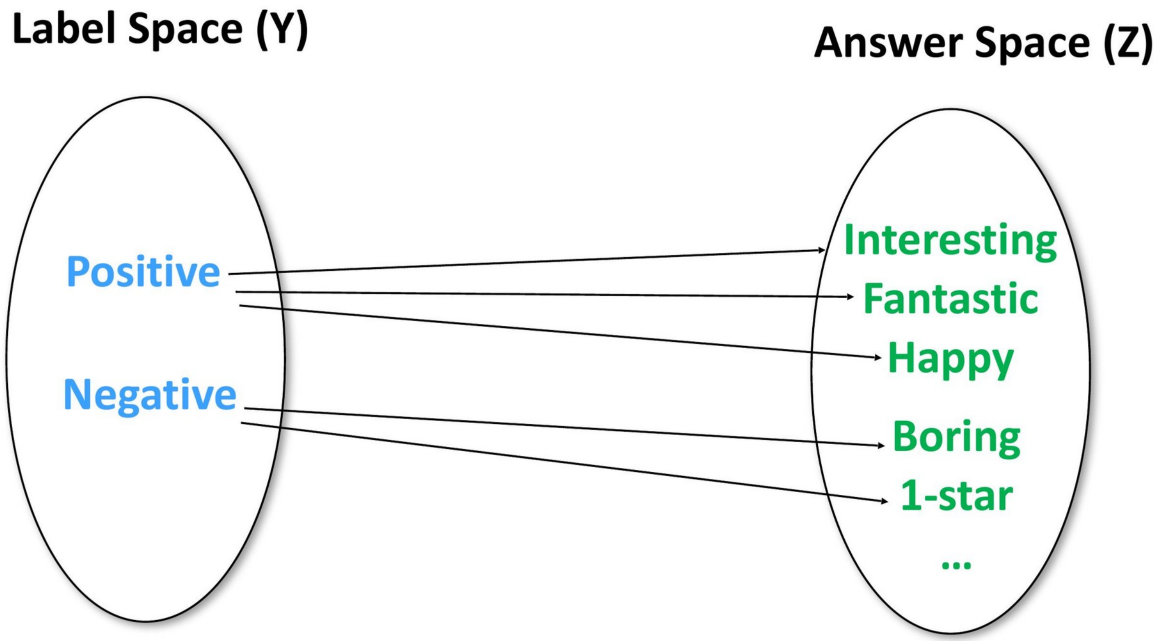
Predicting: x' = "I love this movie. Overall it was a fantastic movie."

⇩

Mapping (answer -> label):
fantastic => Positive

Prompt engineering

Traditional Formulation V.S Prompt Formulation



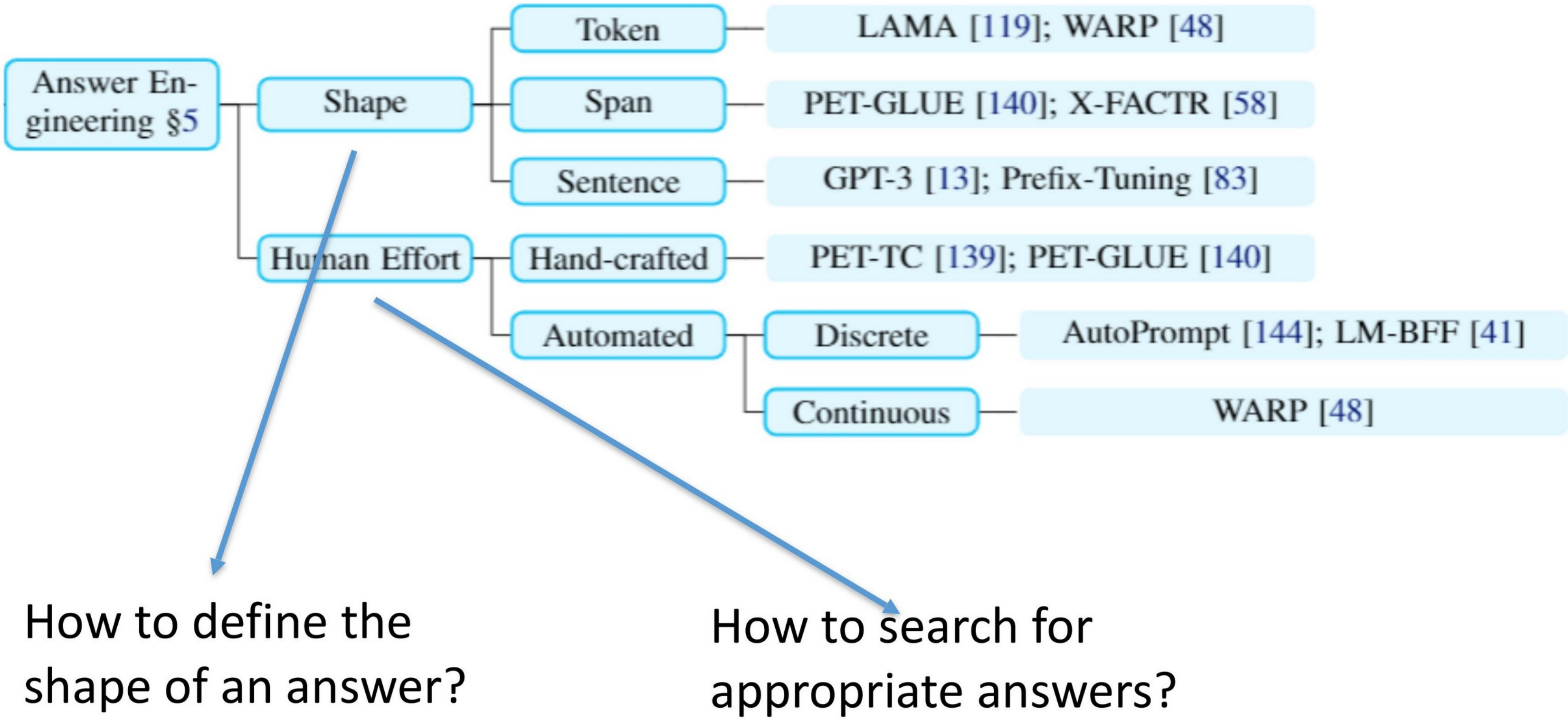
Prompt engineering

Answer Engineering

- Why do we need answer engineering?
  - We have reformulate the task! We also should re-define the "ground truth labels"
- Definition:
  - aims to search for an answer space and a map to the original output Y that results in an effective predictive model

Prompt engineering

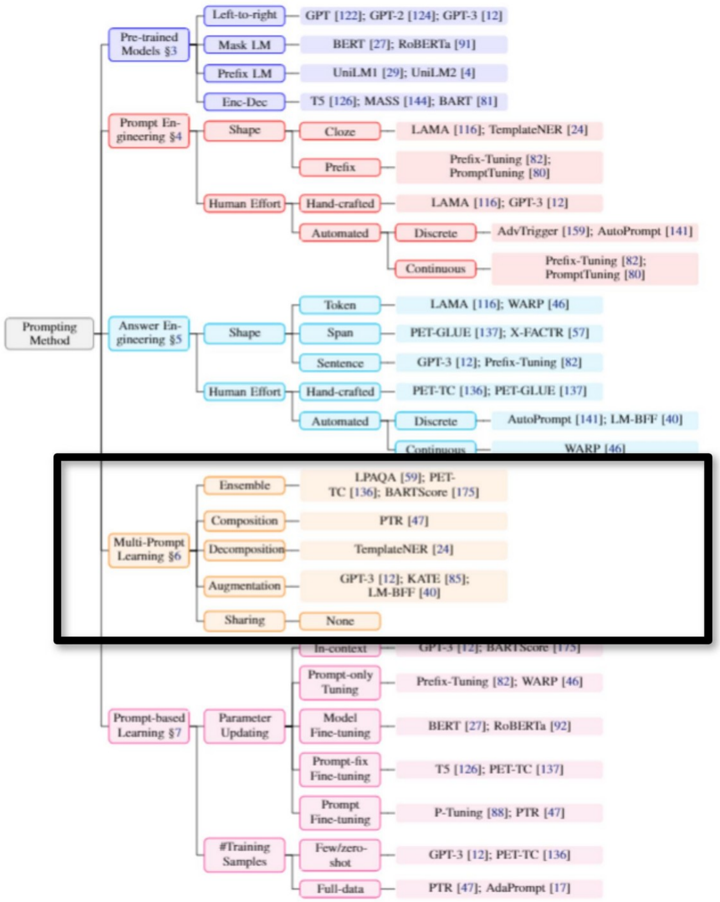
Design of Prompt Answer



Prompt engineering

Design Considerations for Prompting

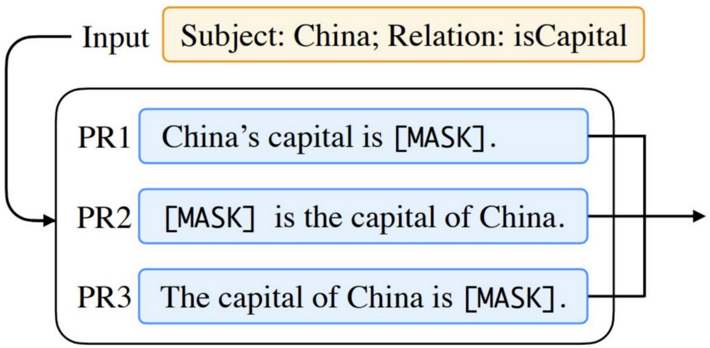
- Pre-trained Model Choice
- Prompt Template Engineering
- Answer Engineering
- Expanding the Paradigm
- Prompt-based Training Strategies



Prompt engineering

Prompt Ensembling

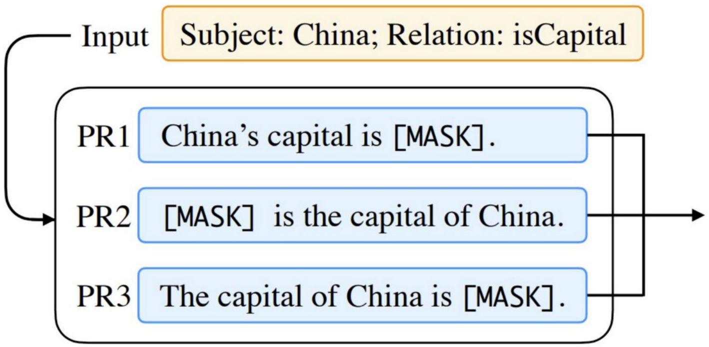
- Definition
  - using multiple unanswered prompts for an input at inference time to make predictions
- Advantages
  - Utilize complementary advantages
  - Alleviate the cost of prompt engineering
  - Stabilize performance on downstream tasks



Prompt engineering

Prompt Ensembling

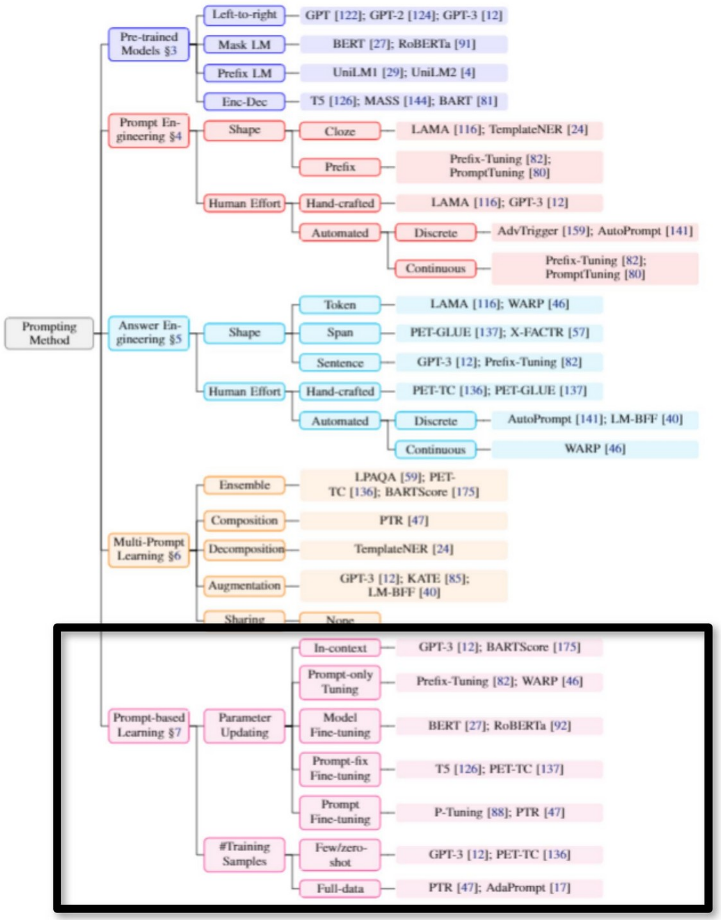
- Typical Methods
  - Uniform Averaging
  - Weighted Averaging
  - Majority Voting



Prompt engineering

Design Considerations for Prompting

- Pre-trained Model Choice
- Prompt Template Engineering
- Answer Engineering
- Expanding the Paradigm
- Prompt-based Training Strategies



Prompt engineering

Prompt-based Training Strategies

- Data Perspective
  - How many training samples are used?
- Parameter Perspective
  - Whether/How are parameters updated?

Prompt-based Training: Data Perspective

- **Zero-shot:** without any explicit training of the LM for the downstream task
- **Few-shot:** few training samples (e.g., 1-100) of downstream tasks
- **Full-data:** lots of training samples (e.g., 10K) of downstream tasks

Prompt engineering

Prompt-based Training: Parameter Perspective

Strategy	LM Params Tuned	Additional Prompt Params	Prompt Params Tuned	Examples
Promptless Fine-Tuning	Yes	N/A	N/A	BERT Fine-tuning
Tuning-free Prompting	No	No	N/A	GPT-3
Fixed-LM Prompt Tuning	No	Yes	Yes	Prefix Tuning
Fixed-prompt LM Tuning	Yes	No	N/A	PET
Prompt+LM Fine-tuning	Yes	Yes	Yes	PADA

OUR OWN WORK

Plan for this session

- Transfer learning:
  - Pre-trained LM, BERT, GPT
  - Pre-trained multilingual LM
  - **Prompting with entailment models**
- Deep learning frameworks
- Last words

Label Verbalization and Entailment for Effective Zero- and Few-Shot Relation Extraction

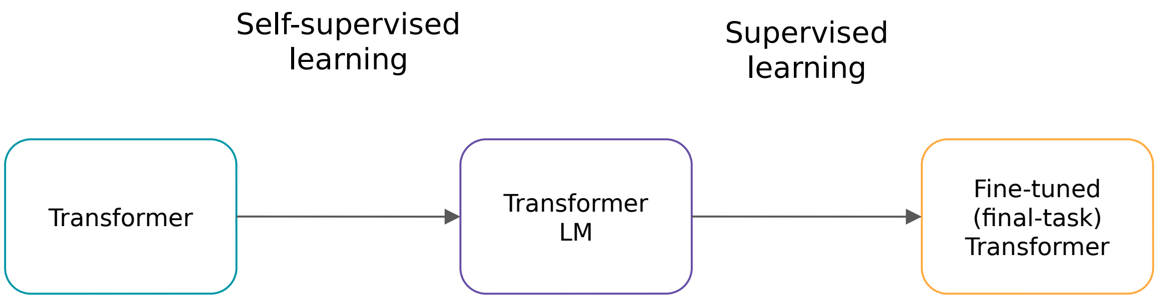
EMNLP 21

Oscar Sainz, Oier Lopez de Lacalle, Gorka Labaka,
Ander Barrena and Eneko Agirre

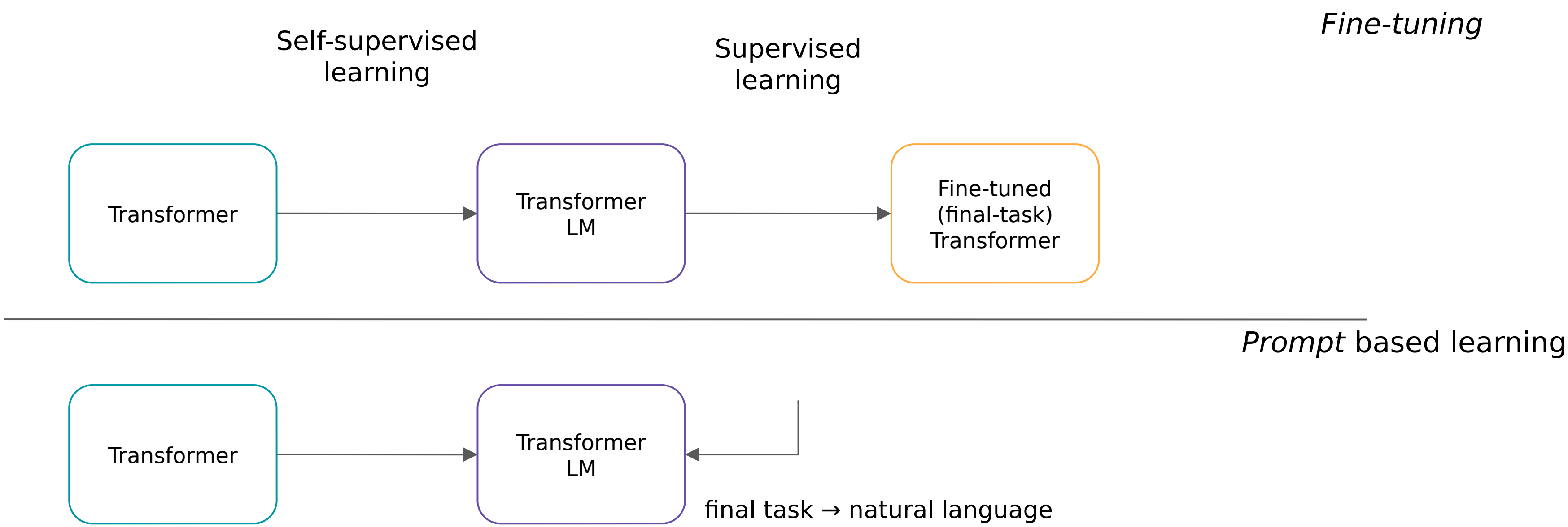
hitz.eus
HiTZ Basque Center on Language Technology
University of the Basque Country (EHU)

Beyond fine-tuning pre-trained language models (PLM)

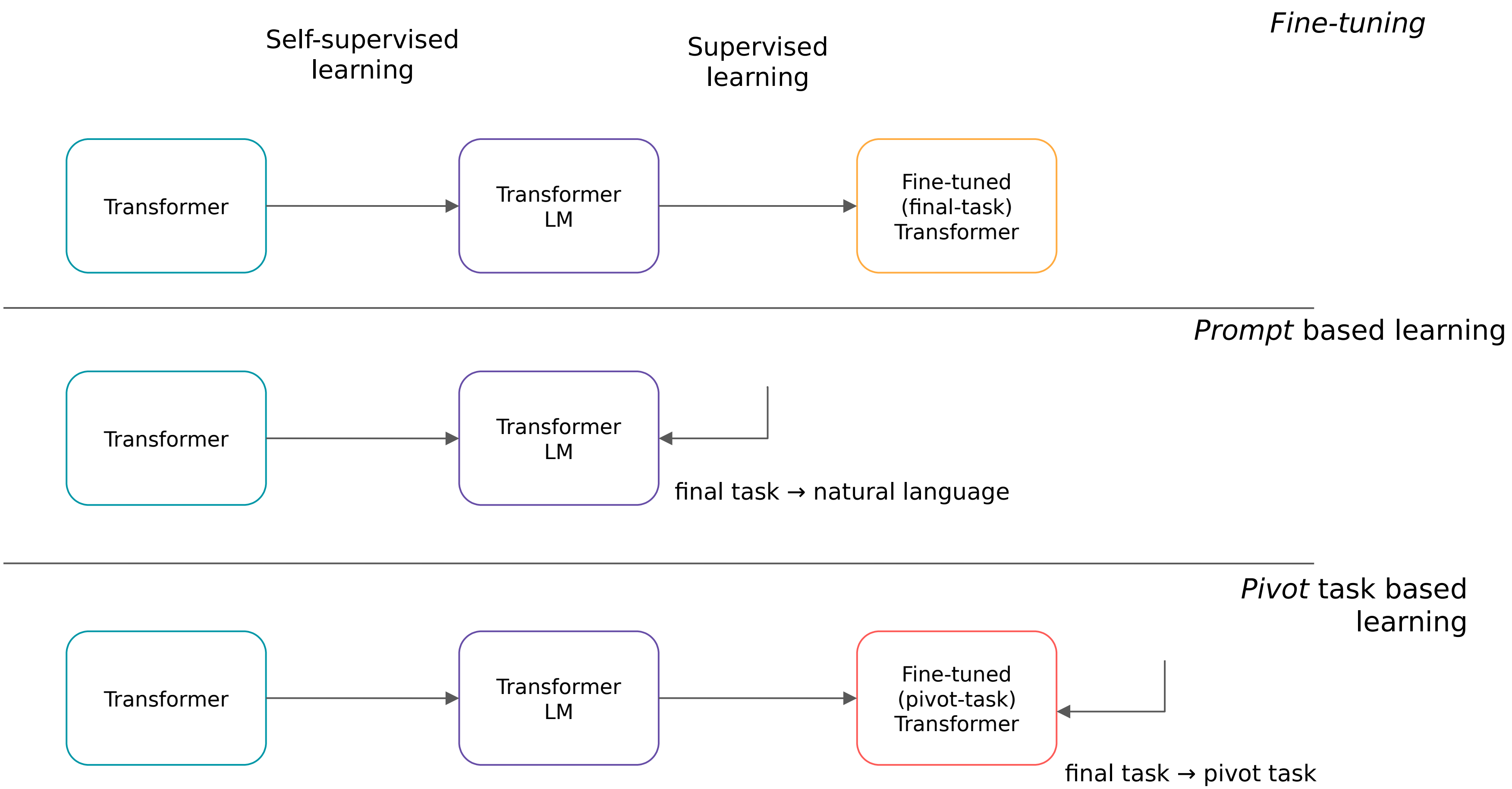
Fine-tuning



Beyond fine-tuning pre-trained language models (PLM)

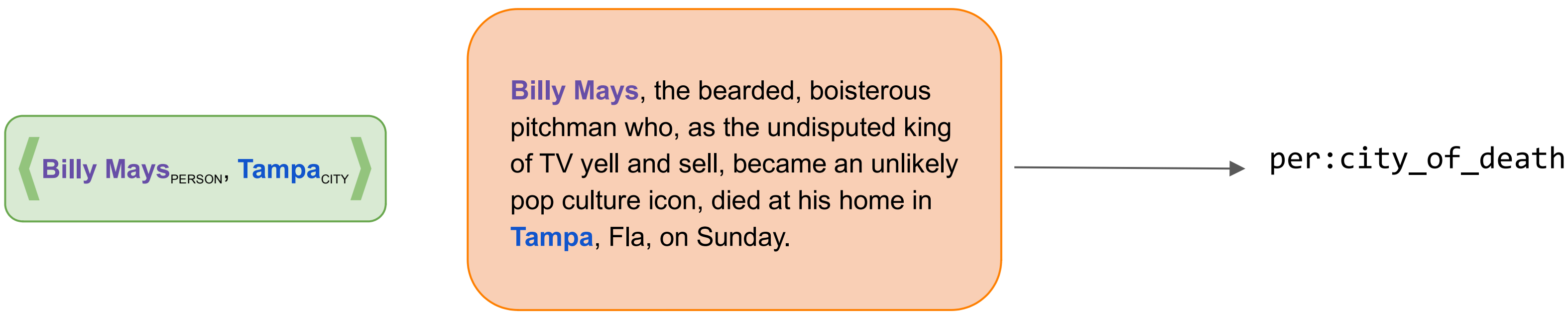


Beyond fine-tuning pre-trained language models (PLM)



Entailment as pivot task for relation extraction

Given 2 entities $e1$ and $e2$ and a context c, predict the semantic relation (if any) holding between the two entities in the context.



Entailment as pivot task for relation extraction

Given 2 entities e1 and e2 and a context c, predict the semantic relation (if any) holding between the two entities in the context.

⟨**Billy Mays**$_{\text{PERSON}}$, **Tampa**$_{\text{CITY}}$⟩

Billy Mays, the bearded, boisterous pitchman who, as the undisputed king of TV yell and sell, became an unlikely pop culture icon, died at his home in **Tampa**, Fla, on Sunday.

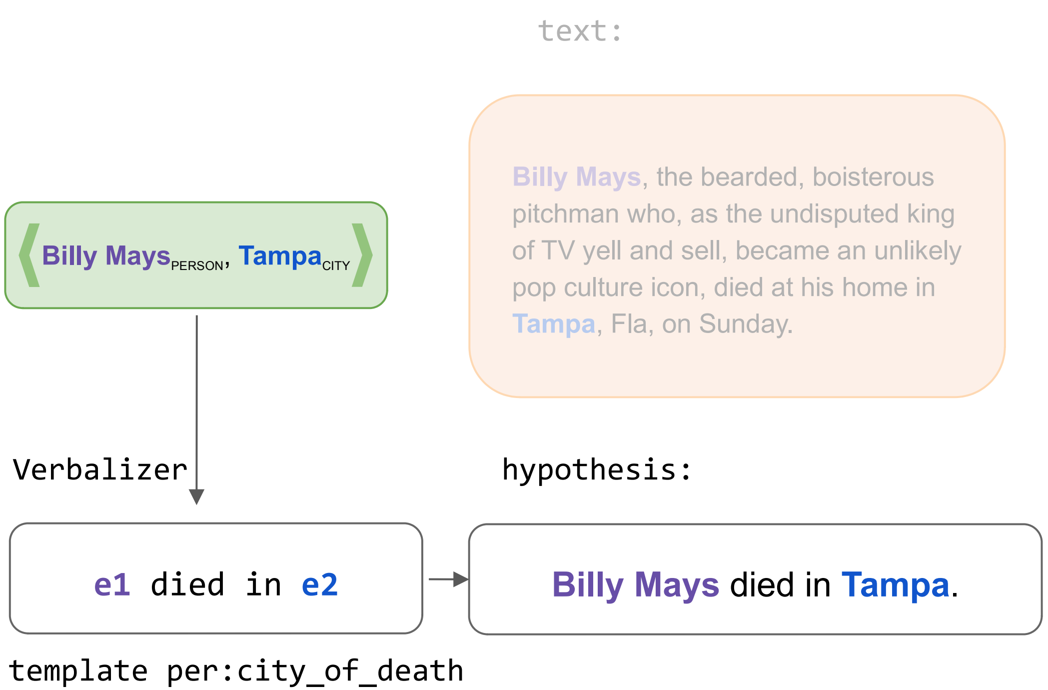
Verbalizer

```
e1 died in e2
```

template per:city_of_death

Entailment as pivot task for relation extraction

Given 2 entities e1 and e2 and a context c, predict the semantic relation (if any) holding between the two entities in the the context.



Entailment as pivot task for relation extraction

Given 2 entities $e1$ and $e2$ and a context c, predict the semantic relation (if any) holding between the two entities in the context.

text:

Billy Mays, the bearded, boisterous pitchman who, as the undisputed king of TV yell and sell, became an unlikely pop culture icon, died at his home in **Tampa**, Fla, on Sunday.

⟨**Billy Mays**$_{PERSON}$, **Tampa**$_{CITY}$⟩

Verbalizer

hypothesis:

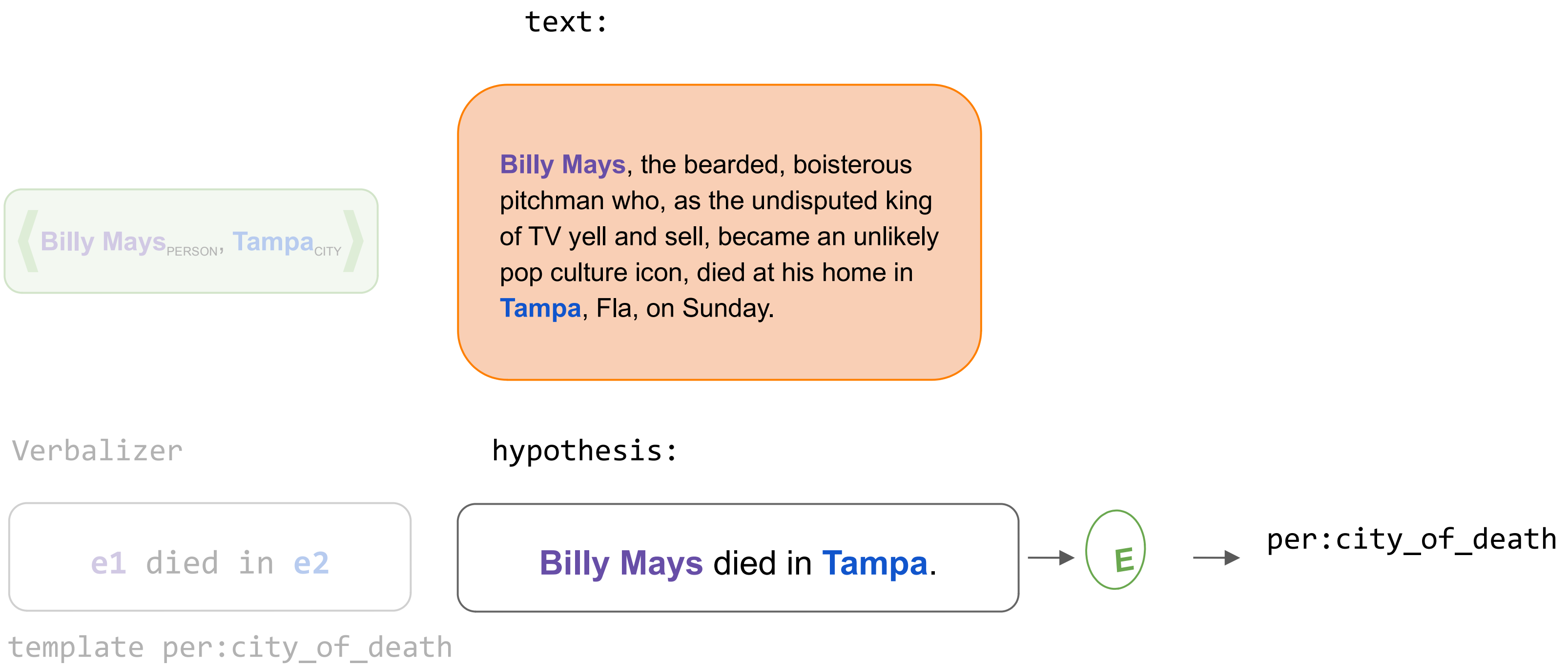
`e1 died in e2`

Billy Mays died in **Tampa**.

template per:city_of_death

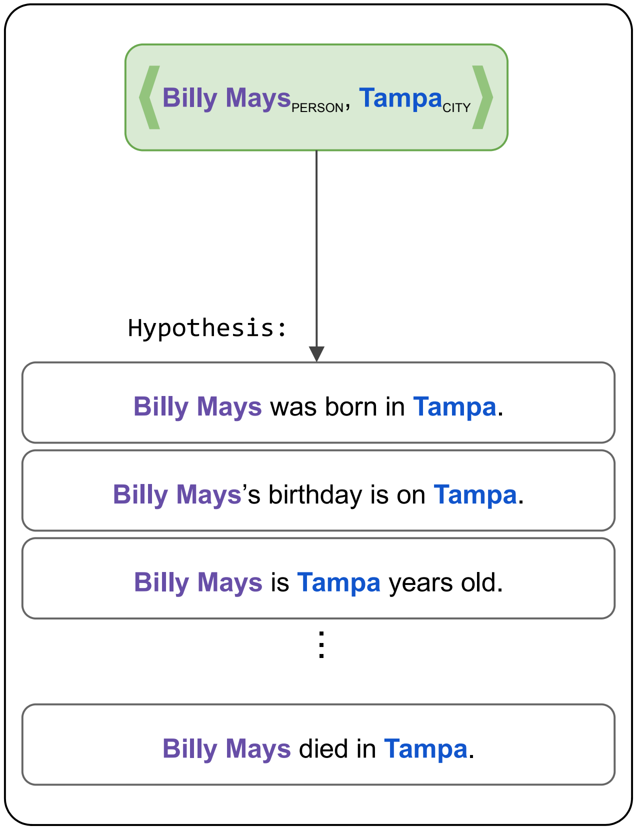
Entailment as pivot task for relation extraction

Given 2 entities $e1$ and $e2$ and a context c, predict the semantic relation (if any) holding between the two entities in the context.



Entailment based Relation Extraction

Verbalizer



- Function that combines entity pairs with templates to generate textual hypotheses for relations:

$$hyp = \text{VERBALIZE}(t, x_{e1}, x_{e2})$$

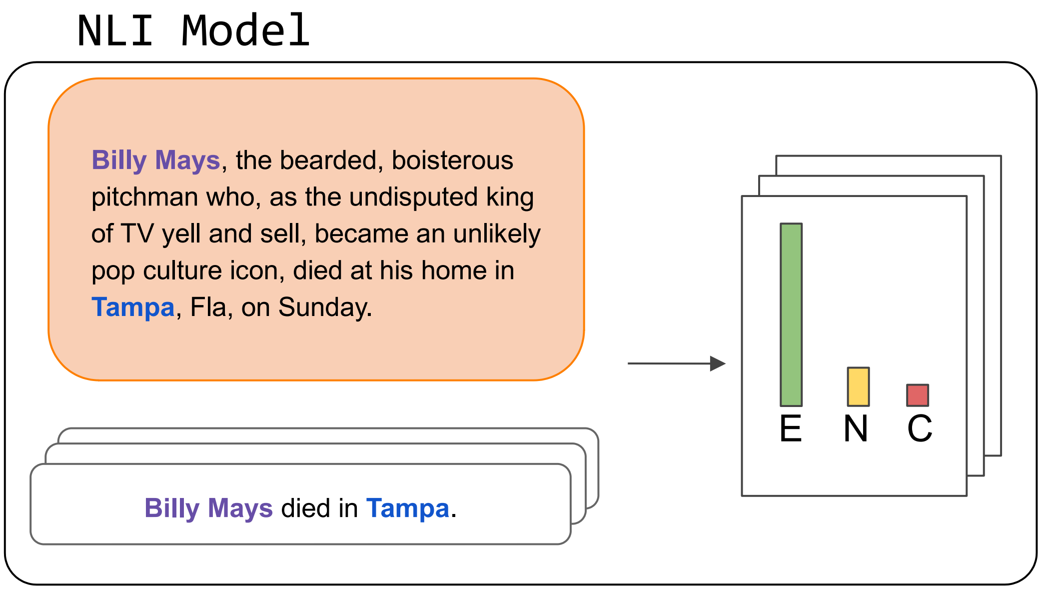
- N:M relation between templates and relations
- Also, type constraints for entities

Relation	Templates	Valid argument types
per:alternate_names	{subj} is also known as {obj}	PERSON, MISC
per:date_of_birth	{subj}'s birthday is on {obj}	DATE
	{subj} was born on {obj}	
per:age	{subj} is {obj} years old	NUMBER, DURATION
per:country_of_birth	{subj} was born in {obj}	COUNTRY
per:stateorprovince_of_birth	{subj} was born in {obj}	STATE_OR_PROVINCE
per:city_of_birth	{subj} was born in {obj}	CITY, LOCATION

Entailment based Relation Extraction

Next, we compute the entailment probabilities for each of the hypothesis independently.

$$P_{NLI}(x, hyp)$$



Entailment based Relation Extraction

$$hyp = \text{VERBALIZE}(t, x_{e1}, x_{e2})$$

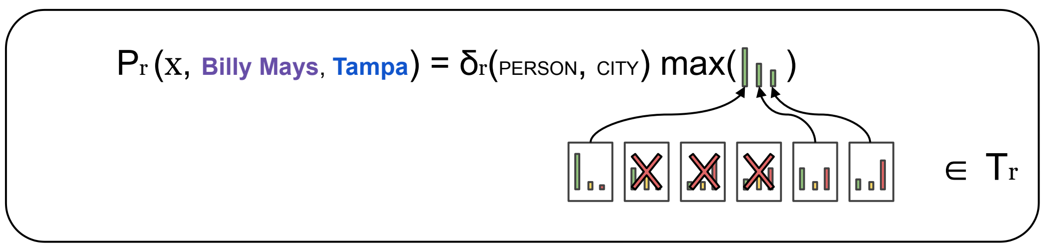
- We compute the probability of relation r based on the hypothesis probabilities and entity constraints:

$$\mathsf{P}_r(x, x_{e1}, x_{e2}) = \delta_r(e_1, e_2) \max_{t \in T_r} \mathsf{P}_{NLI}(x, hyp)$$

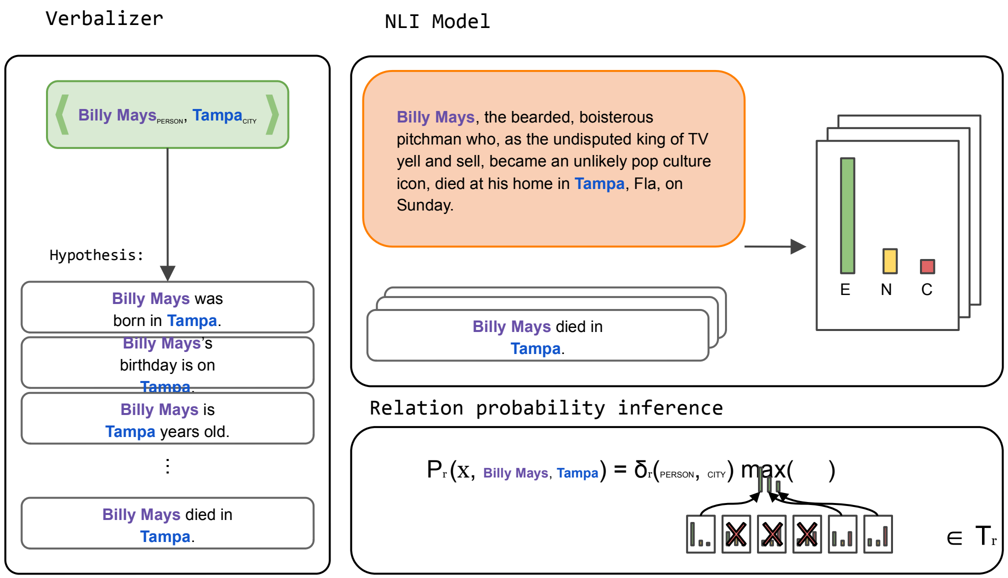
- The δ_r function describes the entity constraints of the relation r:

$$\delta_r(e_1, e_2) = \begin{cases} 1 & e_1 \in E_{r1} \wedge e_2 \in E_{r2} \\ 0 & \text{otherwise} \end{cases}$$

Relation probability inference



Entailment based Relation Extraction

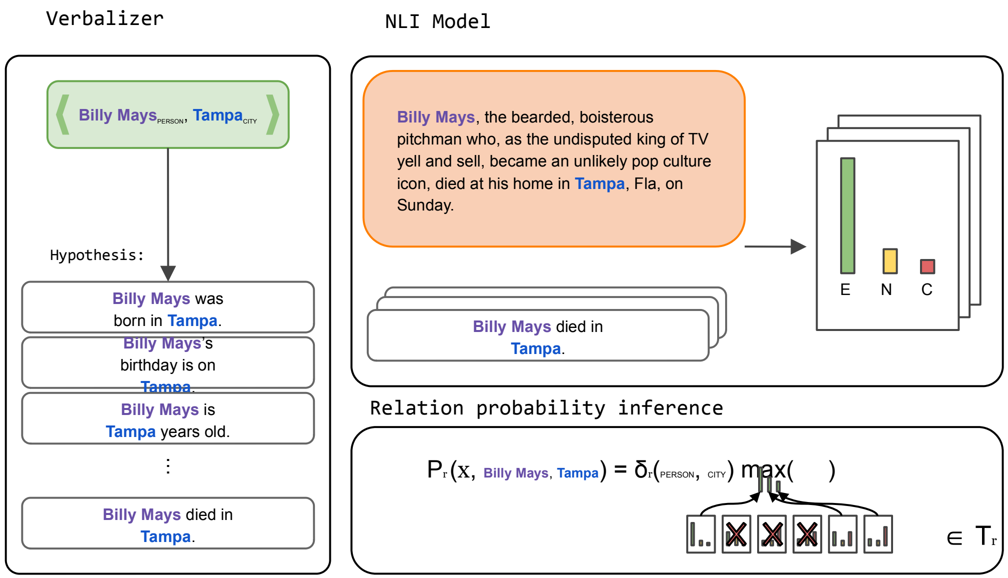


Finally, we return the relation with the highest probability:

$$\hat{r} = \arg\max_{r \in R} \mathrm{P}_r(x, x_{e1}, x_{e2})$$

If no relation has higher entailment probability than threshold (*T*, default 0.5), *r* = **`no_relation`**

Entailment based Relation Extraction



Finally, we return the relation with the highest probability:

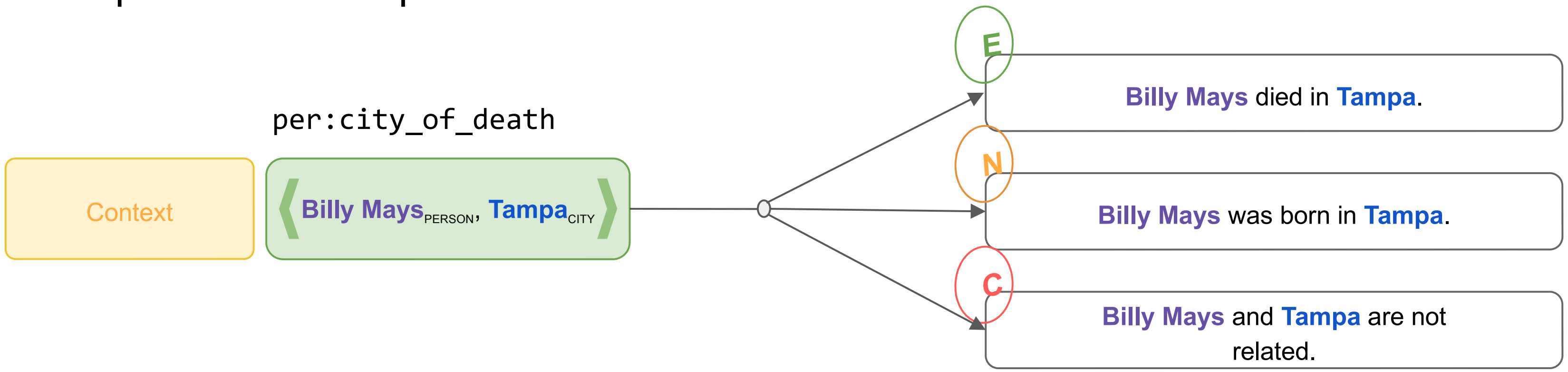
$$\hat{r} = \arg\max_{r \in R} \mathrm{P}_r(x, x_{e1}, x_{e2})$$

If no relation has higher entailment probability than threshold (*T*, default 0.5), *r* = **`no_relation`**

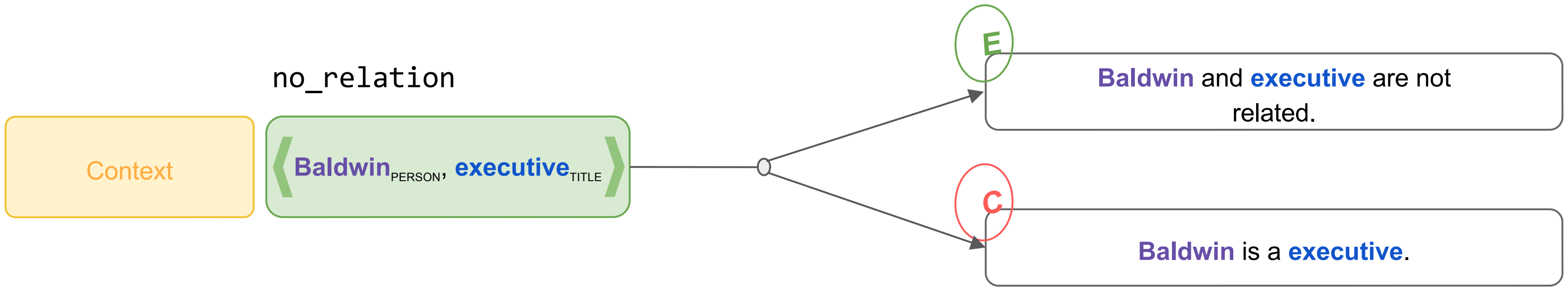
ZERO-SHOT

Fine-tuning NLI on Relation Extraction data

For positive examples:

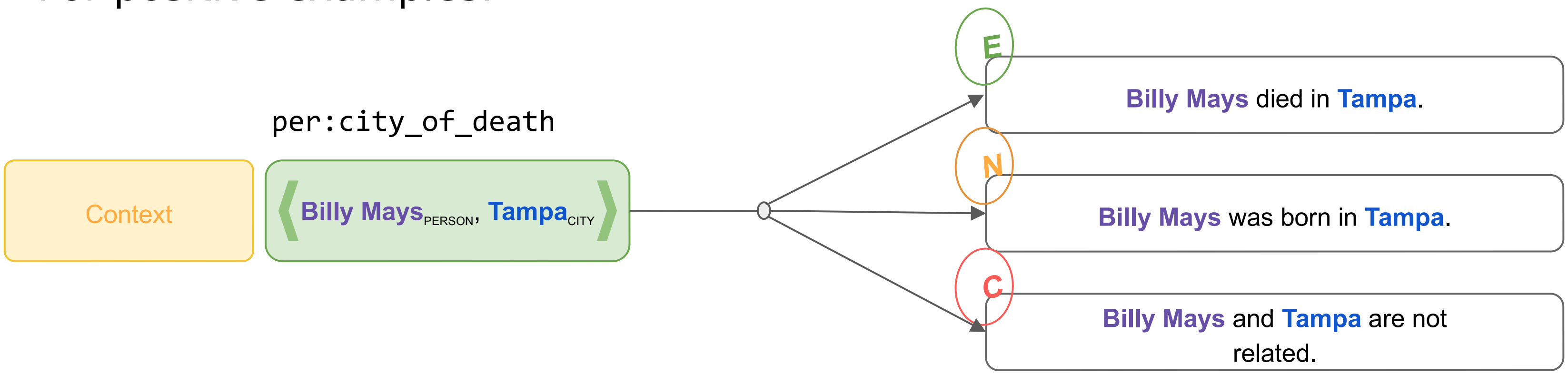


For negative examples:

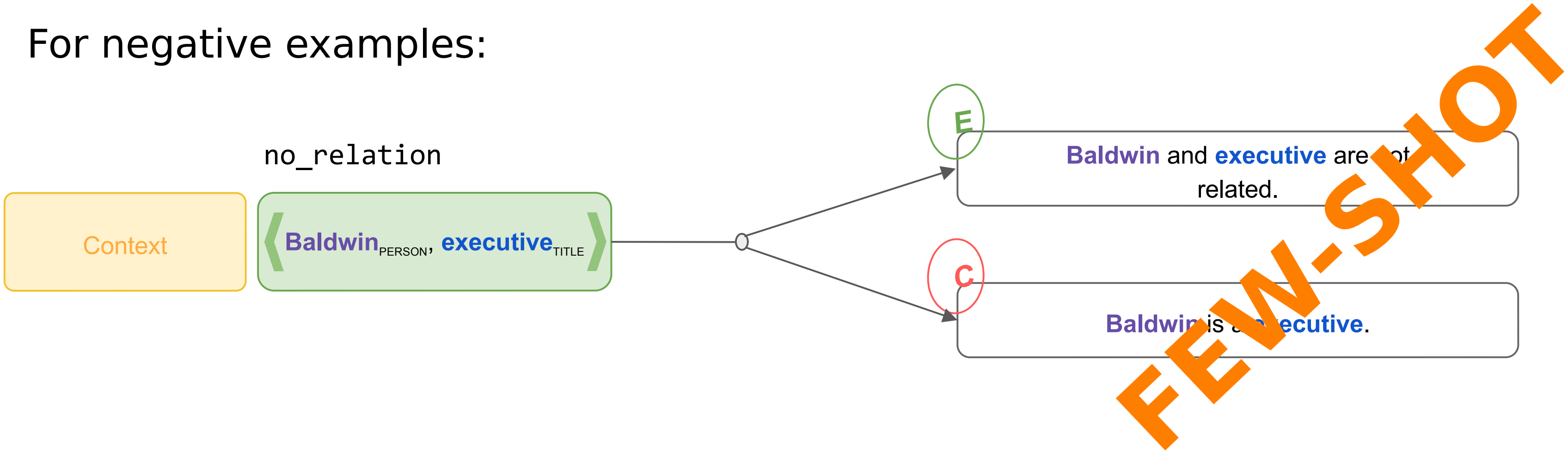


Fine-tuning NLI on Relation Extraction data

For positive examples:



For negative examples:



Dataset

TACRED (Zhang et al., 2017). 42 relation labels.

Scenario	Split	Train (Gold) # Pos mean	Train (Gold) # Pos total	Train (Gold) # Neg total	Development # Pos mean	Development # Pos total	Development # Neg total
Full training	100%	317.4	13013	55112	132.6	5436	17195
Zero-Shot	No Dev	-	-	-	0	0	0
	1% Dev	-	-	-	1.3	54	173
Few-Shot	1%	3.6	130	552	1.3	54	173
	5%	16.3	651	2756	7.0	272	861
	10%	32.6	1302	5513	13.6	544	1721

Zero-Shot

NLI Model	# Param.	MNLI Acc.
$\text{ALBERT}_{\text{xxLarge}}$	223M	90.8
RoBERTa	355M	90.2
BART	406M	89.9
$\text{DeBERTa}_{\text{xLarge}}$	900M	91.7
$\text{DeBERTa}_{\text{xxLarge}}$	1.5B	91.7

Zero-Shot relation extraction:

Zero-Shot

NLI Model	# Param.	MNLI Acc.	No Dev ($\mathcal{T} = 0.5$) Pr.	Rec.	F1
$\text{ALBERT}_{\text{xxLarge}}$	223M	90.8	32.6	**79.5**	46.2
RoBERTa	355M	90.2	32.8	75.5	45.7
BART	406M	89.9	39.0	63.1	48.2
$\text{DeBERTa}_{\text{xLarge}}$	900M	91.7	40.3	77.7	53.0
$\text{DeBERTa}_{\text{xxLarge}}$	1.5B	91.7	**46.6**	76.1	**57.8**

Zero-Shot relation extraction:

- Best results with DeBERTa

Zero-Shot

NLI Model	# Param.	MNLI Acc.	No Dev ($\mathcal{T} = 0.5$) Pr.	Rec.	F1
$\text{ALBERT}_{\text{xxLarge}}$	223M	90.8	32.6	**79.5**	46.2
RoBERTa	355M	90.2	32.8	75.5	45.7
BART	406M	89.9	39.0	63.1	48.2
$\text{DeBERTa}_{\text{xLarge}}$	900M	91.7	40.3	77.7	53.0
$\text{DeBERTa}_{\text{xxLarge}}$	1.5B	91.7	**46.6**	76.1	**57.8**

Zero-Shot relation extraction:

- Best results with DeBERTa
- Note that minor variations in MNLI (±2) produce large variations in F1.

Few-Shot

Model	1%			5%			10%		
	Pr.	Rec.	F1	Pr.	Rec.	F1	Prec.	Rec.	F1
SpanBERT	0.0	0.0	0.0 ±0.0	36.3	23.9	28.8 ±13.5	3.2	1.1	1.6 ±20.7
RoBERTa	56.8	4.1	7.7 ±3.6	52.8	34.6	41.8 ±3.3	61.0	50.3	55.1 ±0.8
K-Adapter	73.8	7.6	13.8 ±3.4	56.4	37.6	45.1 ±0.1	62.3	50.9	56.0 ±1.3
LUKE	61.5	9.9	17.0 ±5.9	57.1	47.0	51.6 ±0.4	60.6	60.6	60.6 ±0.4

Few-Shot relation extraction:

- State of the art systems have difficulties to learn the task where very small amount of data is annotated.

Few-Shot

Model	1%			5%			10%		
	Pr.	Rec.	F1	Pr.	Rec.	F1	Prec.	Rec.	F1
SpanBERT	0.0	0.0	0.0 ±0.0	36.3	23.9	28.8 ±13.5	3.2	1.1	1.6 ±20.7
RoBERTa	56.8	4.1	7.7 ±3.6	52.8	34.6	41.8 ±3.3	61.0	50.3	55.1 ±0.8
K-Adapter	73.8	7.6	13.8 ±3.4	56.4	37.6	45.1 ±0.1	62.3	50.9	56.0 ±1.3
LUKE	61.5	9.9	17.0 ±5.9	57.1	47.0	51.6 ±0.4	60.6	60.6	60.6 ±0.4
$NLI_{RoBERTa}$ (ours)	56.6	55.6	56.1 ±0.0	60.4	68.3	64.1 ±0.2	**65.8**	69.9	67.8 ±0.2
$NLI_{DeBERTa}$ (ours)	**59.5**	**68.5**	**63.7** ±0.0	**64.1**	**74.8**	**69.0** ±0.2	62.4	**74.4**	**67.9** ±0.5

Few-Shot relation extraction:

- State of the art systems have difficulties to learn the task where very small amount of data is annotated. Indeed, SpanBERT does not even work.
- NLI systems large improvements over SOTA systems. 1% > 10%
- As in the zero-shot setting, DeBERTa model score the best.

Conclusions

- Verbalization+entailment
  for zero- and few-shot relation-extraction
- Writing templates is easy, 10 hours
  - Zero-shot
    > fine-tuned PLM 1800 ex. (5%)
  - Few-shot 900 ex. (1%)
    > fine-tuned PLM with 9000 ex. (10%)

Plan for this session

- Transfer learning:
  - Pre-trained LM, BERT, GPT
  - Pre-trained multilingual LM
  - Prompting
- **Deep learning frameworks**
- Last words

Deep Learning frameworks

- Tensorflow (Google): most used in industry
- Pytorch (Facebook): most used in research
- Far far beyond:
  - MXNet (Apache): open source
  - CNTK (Microsoft): mostly used in vision
  - Dynet (Carnegie Mellon University): designed for NLP
  - DeepLearning4J (Eclipse): written in Java

Pytorch

```
class Model(nn.Module):

    def __init__(self):
        super(Model, self).__init__()
        self.embedding_layer = nn.Embedding(num_embeddings=20000,
                                            embedding_dim=50)
        self.pooling_layer = nn.AvgPool1d(kernel_size=50)
        self.fc_layer = nn.Linear(in_features=42, out_features=1)


    def forward(self, inputs):

        x = self.embedding_layer(inputs)
        x = self.pooling_layer(x).view(32, 42)

        return torch.sigmoid(self.fc_layer(x))

model = Model()
```

https://towardsdatascience.com/pytorch-vs-tensorflow-in-code-ada936fd5406

Pytorch

```
criterion = nn.BCELoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

batch_losses = []
for epoch in range(1):
    dataiter = iter(dataloader)
    for batch in dataiter:
        optimizer.zero_grad()
        out = model(batch["tweets"])
        loss = criterion(out, batch["sentiments"])
        batch_losses.append(loss.item())
        loss.backward()
        optimizer.step()
```

https://towardsdatascience.com/pytorch-vs-tensorflow-in-code-ada936fd5406

Tensorflow

```
class Subclass_Model(tf.keras.Model):

    def __init__(self, embedding_dim=25):

        super(Subclass_Model, self).__init__()
        self.embedding_layer = tf.keras.layers.Embedding(input_dim=20000,
                                                         output_dim=50,
                                                         input_length=42)
        self.pool1D_layer = tf.keras.layers.GlobalAveragePooling1D()
        self.fc_layer =  tf.keras.layers.Dense(1, activation='sigmoid')

    def call(self, inputs):

        x = self.embedding_layer(inputs)
        x = self.pool1D_layer(x)
        return self.fc_layer(x)

model = Subclass_Model()
```

https://towardsdatascience.com/pytorch-vs-tensorflow-in-code-ada936fd5406

Tensorflow

```
model.compile(loss='binary_crossentropy', optimizer='Adam', metrics=['accuracy'])
model.fit(x=seq,
          y=labels,
          batch_size=32,
          epochs=1,
          verbose=2,
          validation_split=0.2)
```

https://towardsdatascience.com/pytorch-vs-tensorflow-in-code-ada936fd5406

Deep Learning frameworks

- Main frameworks have converged:
  - **Pytorch**: no need to compile graph!
    Pythonic – object oriented, object reuse
    dynamic graphs, high level abstraction,
    graph features for performance, jit (similar to TF)
  - **Tensorflow**: graph mode fast XLA, portable, autograph
    eager execution (TF 2.0 default similar to PT),
    Pythonic (TF 2.0 similar to PT)
    Keras: high level abstraction on top of TF

Plan for this session

- Transfer learning:
  - Pre-trained LM, BERT, GPT
  - Pre-trained multilingual LM
  - Prompting
- Deep learning frameworks
- **Last words**

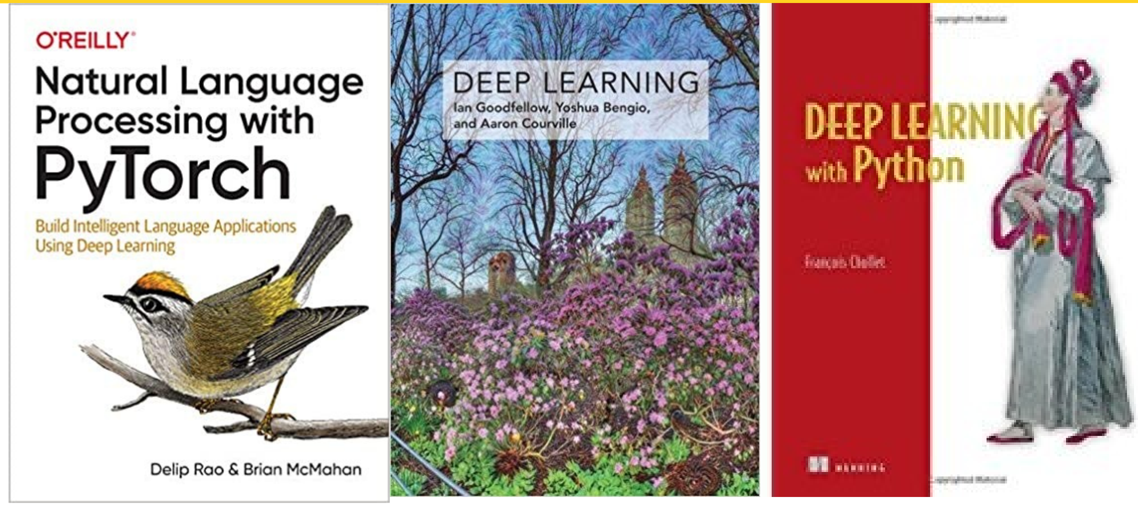
Last words

- We have only scratched the surface of representation learning and NLP
- Plenty of research in multilingual language understanding
- Also applied research (transfer to industry)
- If interested (e.agirre@ehu.eus)
  - Course on July DL4NLP in 3 days Keras high-level API
  - HAP-LAP Master program http://ixa.eus/master/
  - PhD program for students with a master, visiting researchers
  - Collaborations with industry: consulting, on-demand contracts, projects

Last words

- Ixa Research group: ixa.eus
- HiTZ Basque Center for Language Technology: hitz.eus
- Local projects with deep learning
  - Prompting
  - Interactive learning
  - Question answering and dialogue
  - Common-sense using knowledge bases and LMs
  - Multimodal grounding and common-sense acquisition
  - Medical domain
  - Social media

Thanks!



Further Resources:

- Books!!
- Tutorials and implementations in Tensorflow, Pytorch websites
- Hugging Face Course: https://huggingface.co/course
- NLP and DL courses: Stanford https://nlp.stanford.edu/teaching/
  NYU https://www.nyu.edu/projects/bowman/teaching.shtml
  CMU http://phontron.com/class/anlp2021/