

# Classification and sequence tagging with transformers.

*Using encoders to generate feature  
representations for various NLU tasks.*

# Outline

- Typology of NLU tasks
- Sentiment analysis
- Natural language inference
- Named entity recognition
- Question answering

# Types of NLU problems

# Types of NLU problems

NLU = natural language understanding = extracting meta data from texts

## Single text classification

- spam detection
- sentiment analysis
- tagging articles by topic
- toxic comment identification
- intent classification in dialogue systems



## Relations between texts

- paraphrase detection
- natural language inference (NLI)
- matching documents with queries
- translation identification
- multiple choice question answering



## Token classification

- named entity recognition (NER)
- part of speech tagging
- slot filling in dialogue systems
- toxic spans detection
- in-text question answering
- grammatical errors detection (and sometimes correction)



## Relations between tokens

- dependency or constituency parsing
- coreference resolution
- extraction of relations between entities
- nested named entity recognition



# NLP problems NOT related to NLU

- Text-to-text (sequence-to-sequence = seq2seq) problems
  - translation
  - summarization
  - dialogue response generation
  - generative question answering
- Text-to-structure problems: seq2seq with a special output format
  - text to code (e.g. SQL, SPARQL, various programming languages)
  - text to meaning representation (e.g. AMR)
- These problems typically require text decoders
  - We will consider them next time
  - Today, **we focus on the problems that can be solved with an encoder-only model** (such as BERT)

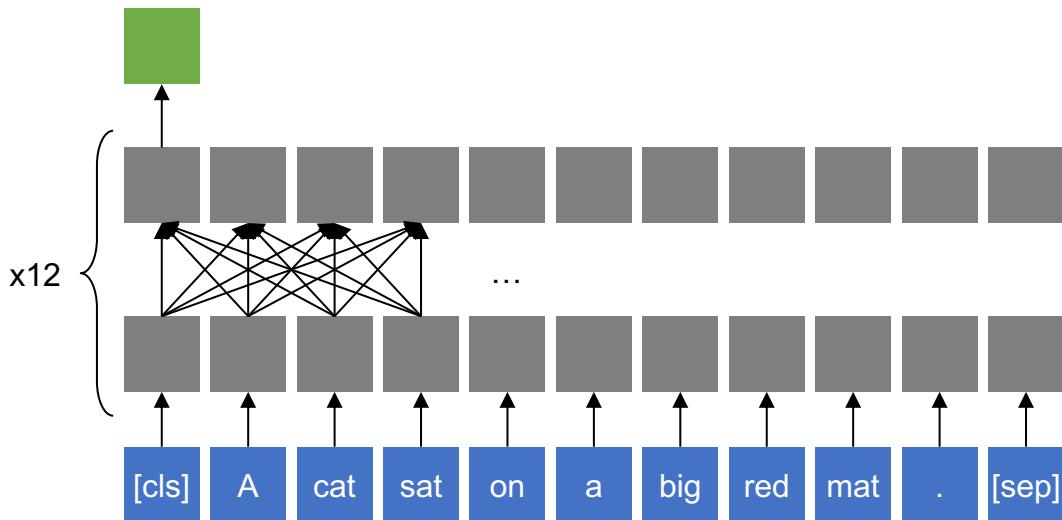
Why not use LLMs (*aka GPT-4, GigaChat*) all the time?



# Architectures for typical NLU problems

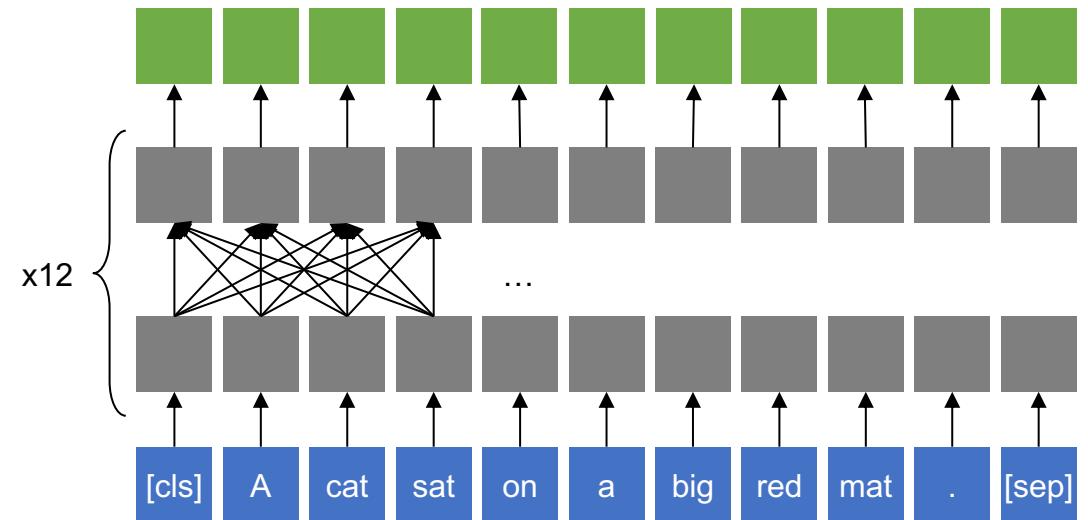
## BERT\* for sequence classification

- Add a linear layer on top of **the first token**
- Fine-tune the whole model
  - The model will automatically learn to deliver all necessary information to the first token



## BERT\* for token classification

- Add a linear layer on top of **each token**
  - Use the same weights for each token
- Fine-tune the whole model

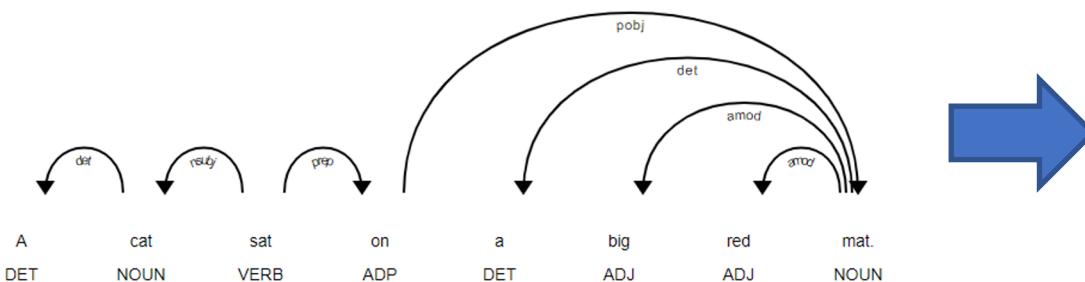


\* Instead of BERT, any other pretrained transformer encoder can be applied

# Biaffine layers for relations between tokens

**The goal:** predict pairwise relations between tokens

- Use a binary classifier for each pair of tokens (or multiclass, to predict relation type)
- Biaffine layer is a simple way to combine two vectors



	A	cat	sat	on	A	big	red	mat
A	0	1	0	0	0	0	0	0
cat	0	0	1	0	0	0	0	0
sat	0	0	0	0	0	0	0	0
on	0	0	0	1	0	0	0	0
a	0	0	0	0	0	0	0	1
big	0	0	0	0	0	0	0	1
red	0	0	0	0	0	0	0	1
mat	0	0	0	0	1	0	0	0

Biaffine function for predicting the  $(i, j)$  arc:

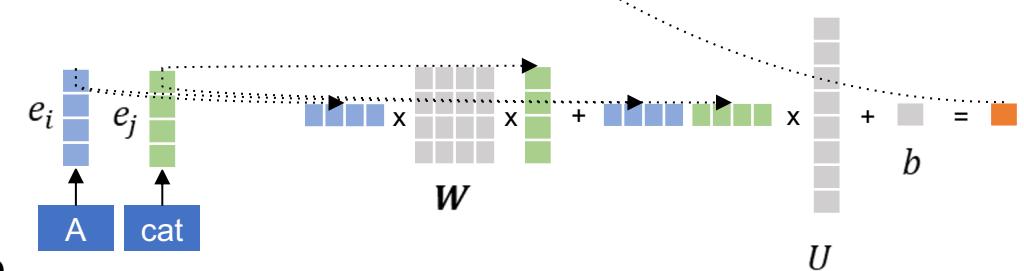
$$score_{ij} = e_i^T \mathbf{W} e_j + (e_i \oplus e_j)^T \mathbf{U} + b$$

$e_i, e_j$  – contextual token embeddings

Parameters:  $\mathbf{W}$  – a matrix,  $\mathbf{U}$  – a vector,  $b$  – a scalar

“Affine” means “linear with bias”.

The function above is affine w.r.t. each argument, i.e. biaffine.



# Sentiment analysis

# Sentiment analysis

- “**Sentiment analysis**” is an umbrella term for multiple tasks:
  - Predict general polarity of a text (positive/neutral/negative)
    - *Coronet has the **best** lines of all day cruisers.*
    - *Pastel-colored 1980s day cruisers from Florida are **ugly**.*
  - Predict polarity w.r.t. a specific object of its *aspect*
    - *I was tempted to buy this product as I **really like** its design, but its **price** is not good.*
    - *Chris Craft is better looking than Limestone, but Limestone projects seaworthiness and reliability.*
  - Predict fine-grained emotions (joy/sadness/fear/anger/surprise etc.)
- Diverse levels of difficulty
  - In simple cases, lists of keywords suffice
  - In complex cases, one has to deal with negations, collocations, sarcasm etc.



Image: <https://github.com/srini047/text-based-sentiment-analysis>

# Related classification problems

There are multiple other emotional phenomena that occur in natural texts and can be detected.

Their degree of difficulty is diverse.

- Subjectivity (Pang and Lee 2008)
- Bias (Recasens et al. 2013; Pryzant et al. 2020)
- Stance (Anand et al. 2011)
- Hate-speech (Nobata et al. 2016)
- Microaggressions (Breitfeller et al. 2019)
- Condescension (Wang and Potts 2019)
- Sarcasm (Khodak et al. 2017)
- Deception and betrayal (Niculae et al. 2015)
- Online trolls (Cheng et al. 2017)
- Polarization (Gentzkow et al. 2019)
- Politeness (Danescu-Niculescu-Mizil et al. 2013)
- Linguistic alignment (Doyle et al. 2016)

# Tips for sentiment analysis (and similar problems)

- *AutoModelForSequenceClassification* will do the job well
- Sometimes, up-sampling less frequent classes helps
  - Of course, it depends on the choice of metrics
- Both multiclass classification and regression work well
  - To convert between different sentiment scales, an [ordered logistic regression](#) head may be used
  - Sometimes, extra classes like “mixed sentiment” are required
- Data augmentation may help a lot
  - Randomly remove “too easy” features (e.g. punctuation, smileys)
  - Check how prediction change under simple transformations\*

\* A good example of such methodology is in [Beyond Accuracy: Behavioral Testing of NLP models with CheckList](#) by Ribeiro et al (2020)

# Aspect-based sentiment analysis

- Analyze “Great **food** but the **service** is dreadful”
- Aspect extraction: given a text, detect “food” and “service”
  - Multilabel text classification: for each possible aspect (from a fixed set), predict whether it is present in the text
  - Text pair classification (NLI-style): given an aspect and a text, predict whether it is relevant (or even its sentiment)
  - Token classification (NER-style): highlight tokens that describe aspects
- Opinion extraction: given a text and an aspect, classify opinion
  - Single text classification, with aspect in special tags
  - Text pair classification (NLI-style)

# Natural language inference

# Natural language inference

The goal of NLI: detect logical relation between two texts

Premise	Relation	Hypothesis
A turtle danced.	entails	A turtle moved.
turtle	contradicts	linguist
Every reptile danced.	neutral	A turtle ate.
Some turtles walk.	contradicts	No turtles move.
James Byron Dean refused to move without blue jeans.	entails	James Dean didn't dance without pants.
Mitsubishi Motors Corp's new vehicle sales in the US fell 46 percent in June.	contradicts	Mitsubishi's sales rose 46 percent.
Acme Corporation reported that its CEO resigned.	entails	Acme's CEO resigned.

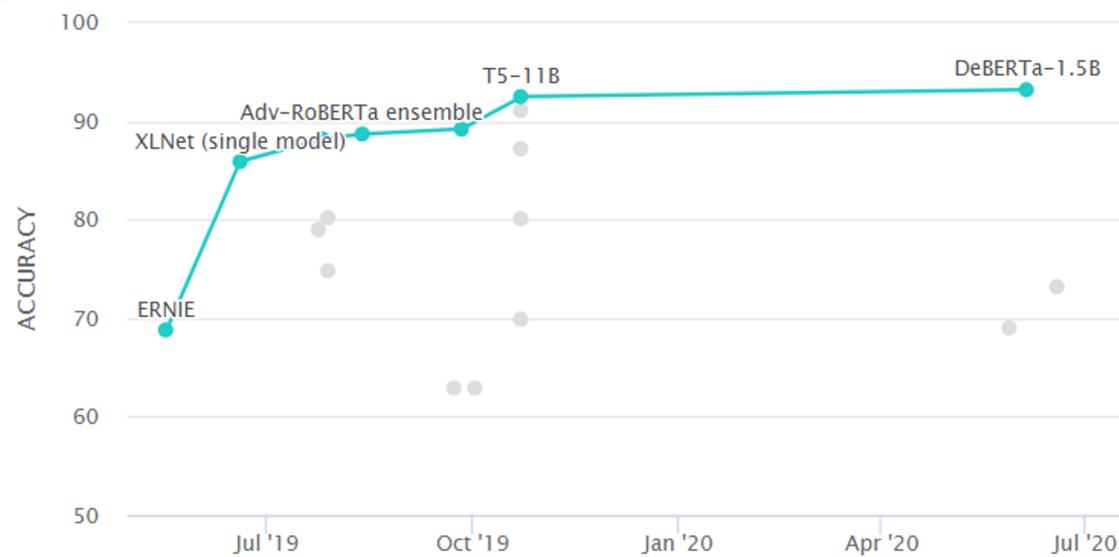
# Why NLI is important

- Diagnostics
  - Evaluate a model's understanding of various linguistic phenomena, logic and world knowledge
    - For example, such a task is a part of the SuperGLUE benchmark
- Applications
  - Controlling quality of generated texts in various tasks
  - Paraphrase detection
  - Zero-shot or few-shot text classification
    - Does "*The IAU downgrade Pluto as a dwarf planet*" imply "*This is science news*"?
  - Pre-training for similar tasks

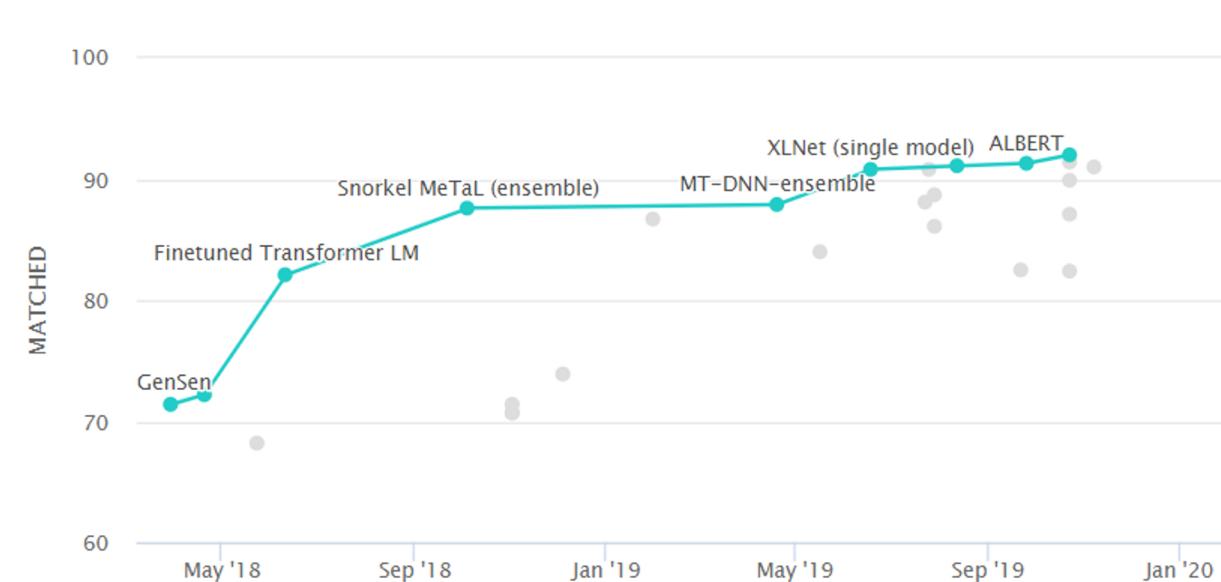
Task	NLI framing
Paraphrase	$\text{text} \equiv \text{paraphrase}$
Summarization	$\text{text} \sqsubset \text{summary}$
Information retrieval	$\text{query} \sqsupset \text{document}$
Question answering	$\text{question} \sqsupset \text{answer}$
	<i>Who left? ⇒ Someone left</i> <i>Someone left ⊃ Sandy left</i>

# NLI benchmarks

RTE (part of GLUE and SuperGLUE)



MultiNLI



- Transformers perform significantly better than alternative architectures
- Traditional benchmarks saturate fast

# Adversarial NLI: a moving target

If standard NLI benchmarks are too easy, one can create an intentionally difficult dataset, using human-in-the-loop data collection

Repeat:

- Ask humans to create examples on which the current model makes mistakes
- Train a new model using these new difficult examples

Results:

- New difficult datasets
- Improvement even on the old datasets

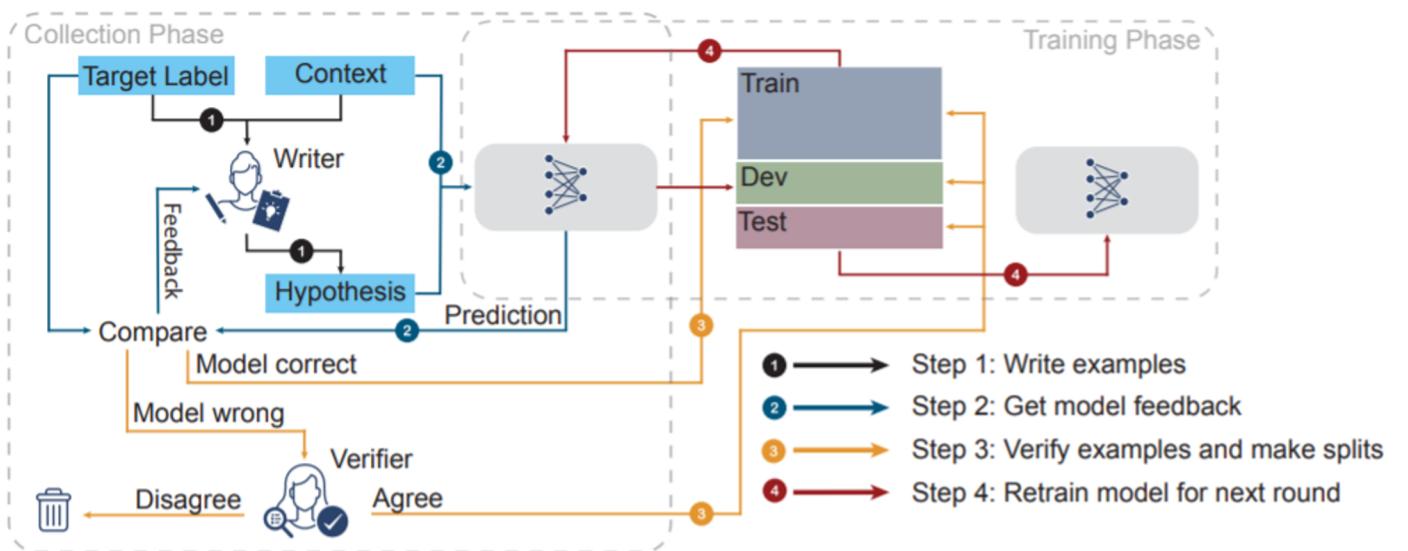


Figure 1: Adversarial NLI data collection via human-and-model-in-the-loop enabled training (HAMLET). The four steps make up one round of data collection. In step 3, model-correct examples are included in the training set; development and test sets are constructed solely from model-wrong verified-correct examples.

# Named entity recognition

# Named entity recognition

**Multiclass classification:** for each token we predict if it is a named entity (and the type of the entity) or not.

## News NER example

### Entity types:

- Locations,
- Organizations,
- Dates, time,
- Names,
- etc.

Kofi Atta Annan is a Ghanaian diplomat who served as the seventh Secretary General of the United Nations from January 1, 1997, to January 1, 2007, serving two five-year terms. Annan was the co-recipient of the Nobel Peace Prize in October 2001.

Kofi Annan was born on April 8, 1938, to Victoria and Henry Reginald Annan in Kumasi, Ghana. He is a twin, an occurrence that is regarded as special in Ghanaian culture. Efua Atta, his twin sister, shares the same middle name, which means 'twin'. As with most Akan names, his first name indicates the day of the week he was born: 'Kofi' denotes a boy born on a Friday. The name Annan can indicate that a child was the fourth in the family, but in his family it was simply a name which Annan inherited from his parents.

In 1962, Annan started working as a Budget Officer for the World Health Organization, an agency of the United Nations. From 1974 to 1976, he the Director of Tourism in Ghana. Annan then returned to work for the Nations as an Assistant Secretary General in three consecutive position

Person
Location
Organization
Date
Nationality
Title

# Why do we need it?



# Why do we need it?

- Text understanding,
- Information retrieval,
- Data preprocessing,
- Machine translation,
- Coreference resolution:  
*“Прискакал Иванушка на белом коне. Принцесса выбежала ему навстречу и поцеловала его.”*
- QA systems,
- *Chat-bots & digital assistants, etc.*

# Not so easy as it may sound

## 1. Polysemantic words and homonyms:

- *Вашингтон* – столица США.
- Джордж *Вашингтон* – первый президент США.

## 2. Entity variations

- *Вас приветствует Магазин Профессиональных Металлоискателей.*
- *Вас приветствует Волхонка Престиж* — ваш любимый магазин брендов по доступным ценам.
- *Вам пишет магазин зоотоваров Немо.*

## 3. We need labelled data for training

# Named entity recognition

- NER is the problem of finding named entities\* within a text

Alexander Sergeyevich Pushkin PERSON was born in Moscow GPE in 1799 year DATE .

- Typically, IO, BIO or BIOES schemes are used for labelling each token
  - BIO helps to distinguish adjacent entities
  - BIOES provides additional information (redundant but sometimes helpful) about entity boundaries

	Alexander	Sergeyevich	Pushkin	was	born	in	Moscow	in	1799	year
IO	PER	PER	PER	O	O	O	GPE	O	DATE	DATE
BIO	B-PER	I-PER	I-PER	O	O	O	B-GPE	O	B-DATE	I-DATE
BIOES	B-PER	I-PER	E-PER	O	O	O	S-GPE	O	B-DATE	E-DATE

\* “Named entity” means proper names. However, in practice we train models to recognize whatever type of entity we need, not necessarily named ones.

# CoNLL-U format

Datasets are often stored in CoNLL-U

1	В	в	PR	PR=	OUT
2	январе	январь	S	S, муж, неод=пр, ед	OUT
3	1995	1995	UNDEF	UNDEF	OUT
4	года	год	S	S, муж, неод=(вин, мн род, ед им, мн)	OUT
5	в	в	PR	PR=	OUT
6	Подмосковье	подмосковье	S	S, гео, сред, неод=(пр, ед вин, ед им, ед)	S-LOC
7	состоялся	состояться	V	V, сов, нп=прош, ед, изъяв, муж	OUT
8	учредительный	учредительный	A	A=(вин, ед, полн, муж, неод им, ед, полн, муж)	OUT
9	съезд	съезд	S	S, муж, неод=(вин, ед им, ед)	OUT
10	Общероссийского	общероссийский	A	A=(вин, ед, полн, муж, од род, ед, полн, муж род, ед, полн, сред)	B-ORG
11	общественного	общественный	A	A=(вин, ед, полн, муж, од род, ед, полн, муж род, ед, полн, сред)	I-ORG
12	движения	движение	S	S, сред, неод=(вин, мн род, ед им, мн)	I-ORG
13	"	"	UNDEF	UNDEF	I-ORG
14	Яблоко	яблоко	S	S, сред, неод=(вин, ед им, ед)	I-ORG
15	" .	".	UNDEF	UNDEF	E-ORG

*В январе 1995 года в Подмосковье состоялся учредительный съезд Общероссийского движения «Яблоко».*

# How to solve?

- Regular expressions for searching fixed pattern formants (emails, phones, etc.)
- Dictionaries and morphological parsers
- Simple ML models
- Fine-tune your own transformer model (e. g. BERT, RoBERTa)
- Use fine-tuned models from HuggingFace
- Specialized architectures
- Prompt engineering with LLMs (e. g. GPT-4, GigaChat)

For standard cases it is often simpler to use a fine-tuned open-source NER model.

# Question answering

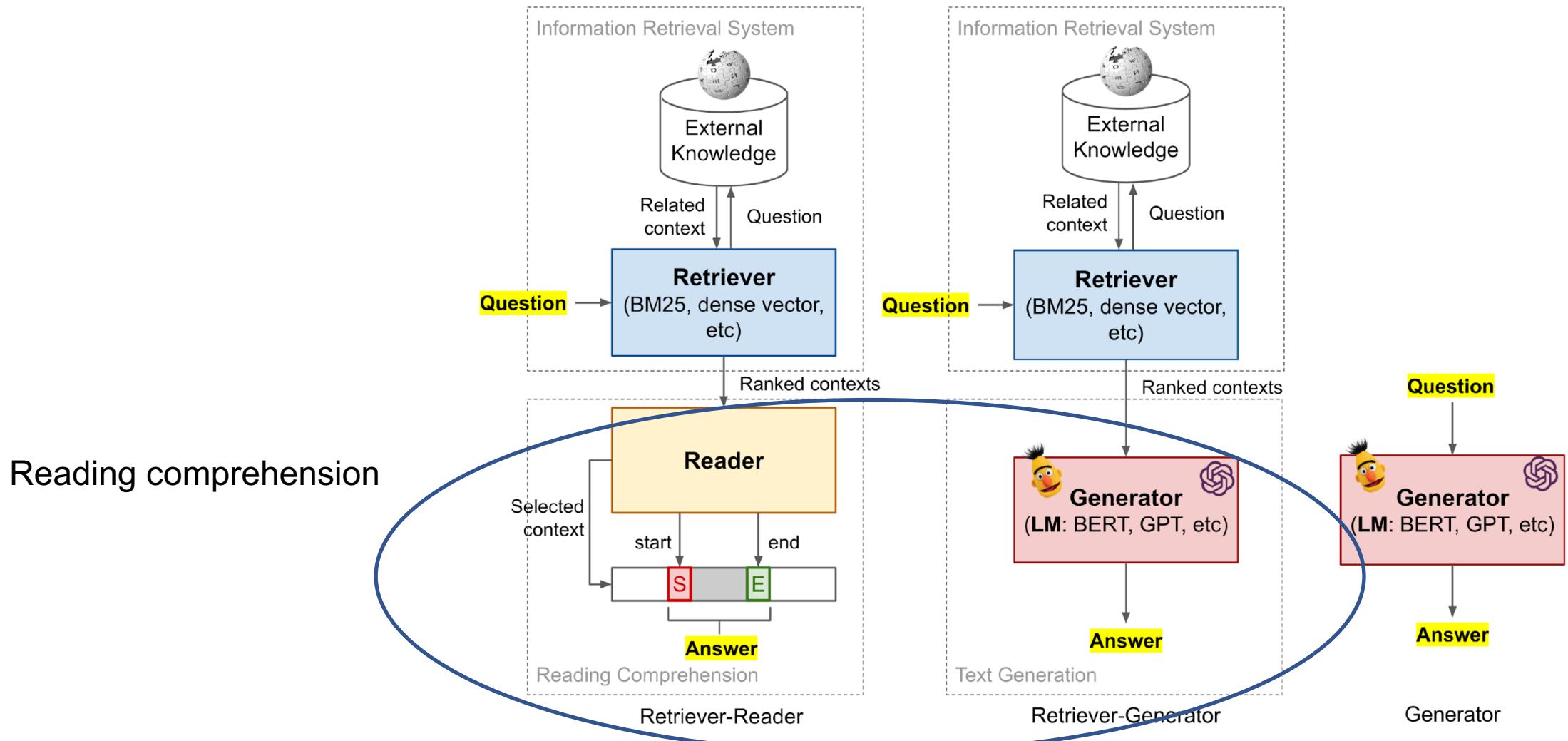
# Typical QA problems and benchmarks

- Extractive QA
  - SQuAD: passage + question; the answer is within the passage
- Open generative QA
  - CoQA: passage + dialogue with questions; the answer is free-form text, but for each answer there is evidence within the passage
  - TriviaQA: question + long evidence documents; the answer is free-form text
- Closed generative QA
  - Just questions are given
- KBQA
  - Question + knowledge base, the answer should be selected or generated

**Question:** How many parameters does BERT-large have?

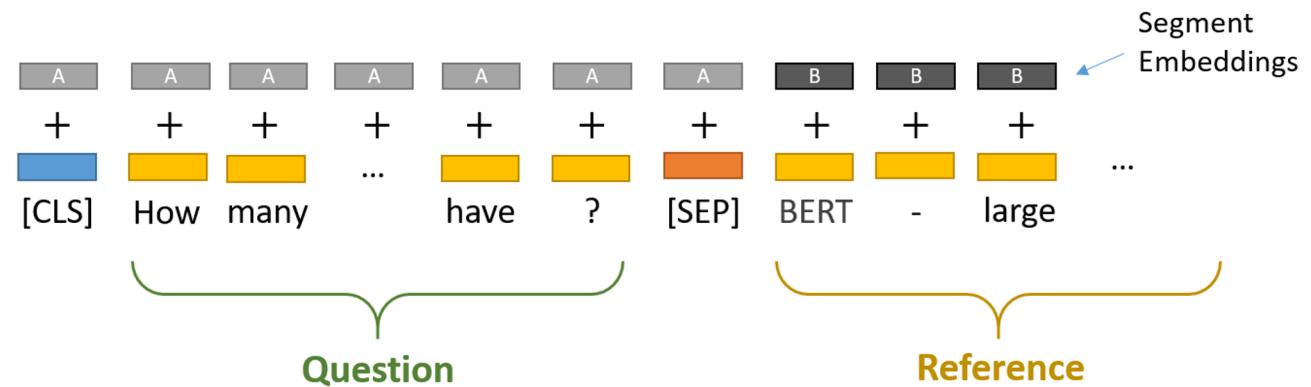
**Reference Text:** BERT-large is really big... it has 24 layers and an embedding size of 1,024, for a total of 340M parameters! Altogether it is 1.34GB, so expect it to take a couple minutes to download to your Colab instance.

# Question answering system architectures



# Extractive question answering

- Used for SQuAD-like tasks: given the question and the context, select the segment of the context which is the right answer
  - If there is no answer, select the first zero-length segment (i.e. [CLS] token)
- Inputs: *the question + separator + the context*
- Outputs: *probabilities of each segment*
- Evaluation: *exact match or F1 over words*

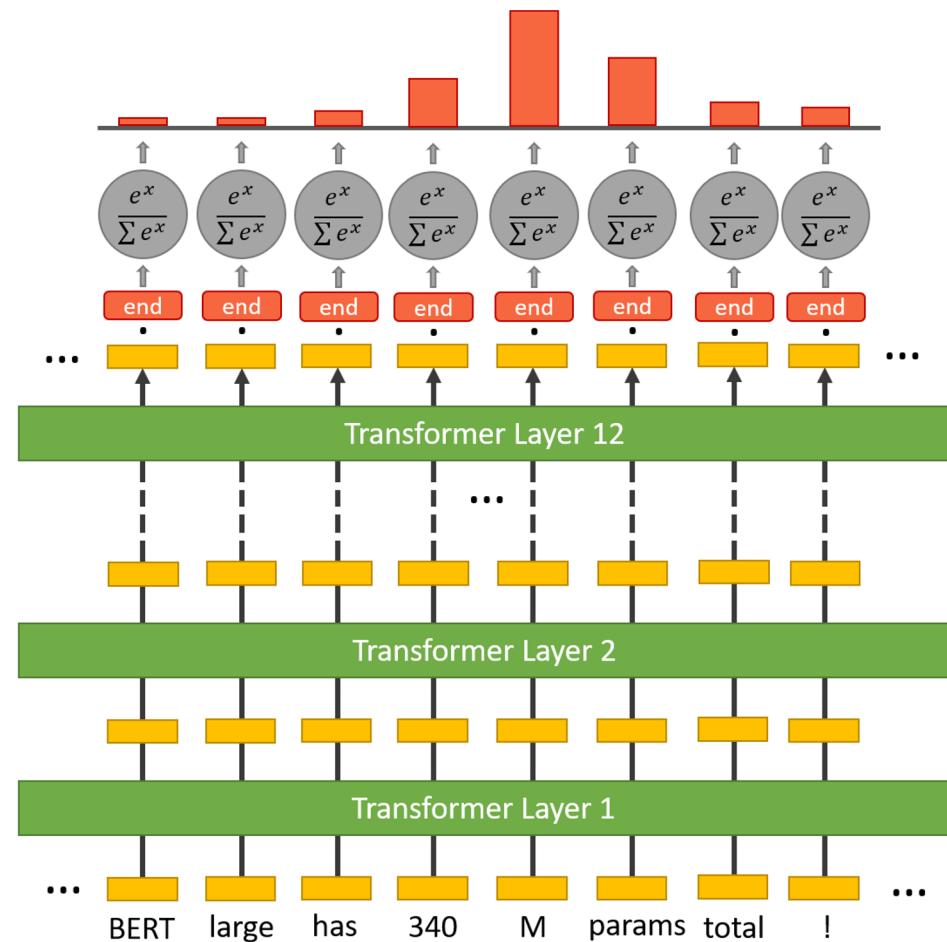
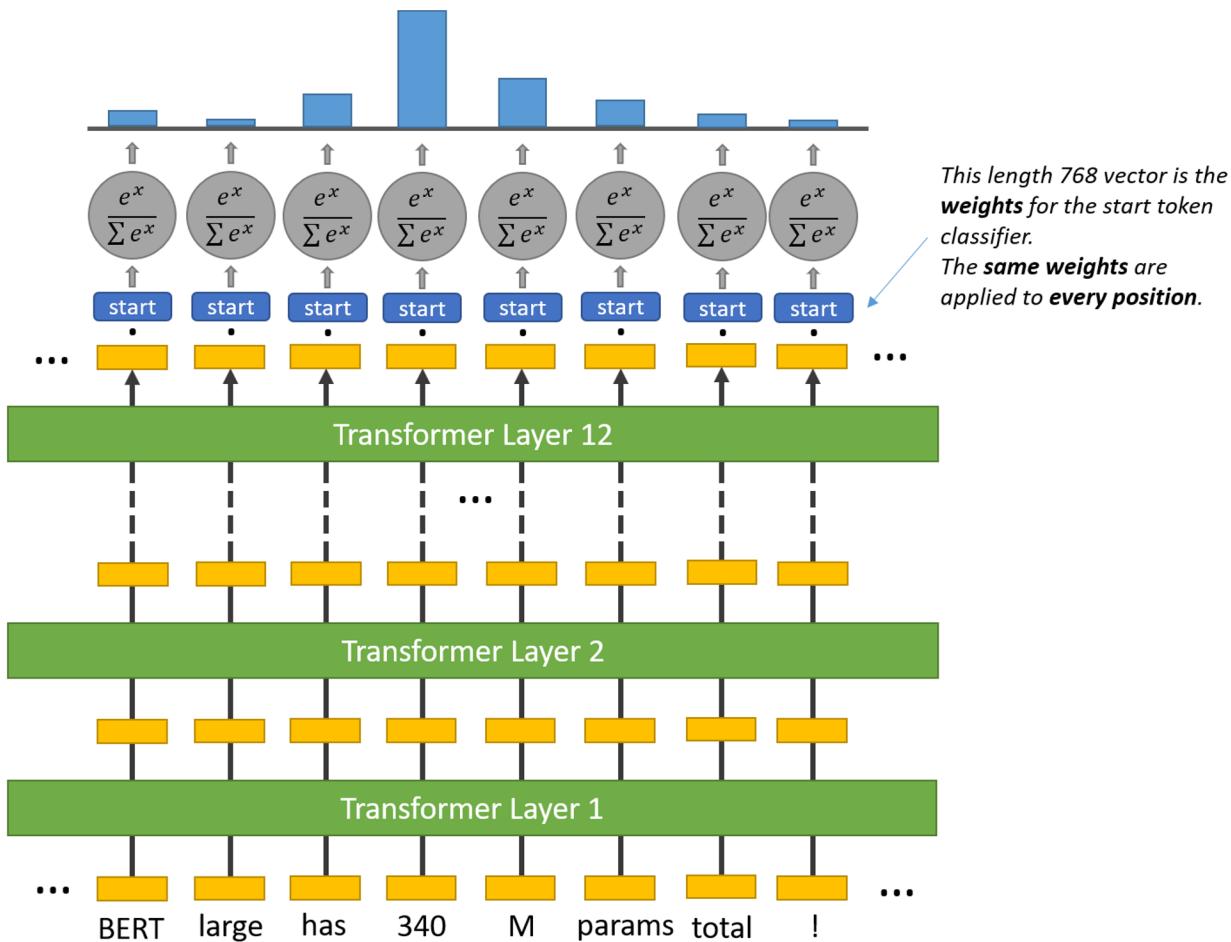


**Question:** How many parameters does BERT-large have?

**Reference Text:** BERT-large is really big... it has 24 layers and an embedding size of 1,024, for a total of 340M parameters! Altogether it is 1.34GB, so expect it to take a couple minutes to download to your Colab instance.

# Extractive question answering

The most likely answer is the span with the maximal product of start and end probabilities.



# Summary

- Encoder-based models (such as BERT) are good for NLU:
  - Classification of texts, tokens, or relations between them
  - Encoding texts into informative vectors
- Many diverse NLP problems can be reformulated as sequence or token classification and solved by standardized models
- Systems based on transformer encoders are widely used for practical applications, such as information retrieval or chatbots