IR recap and Enc-Dec Transformers

NLP Week 10

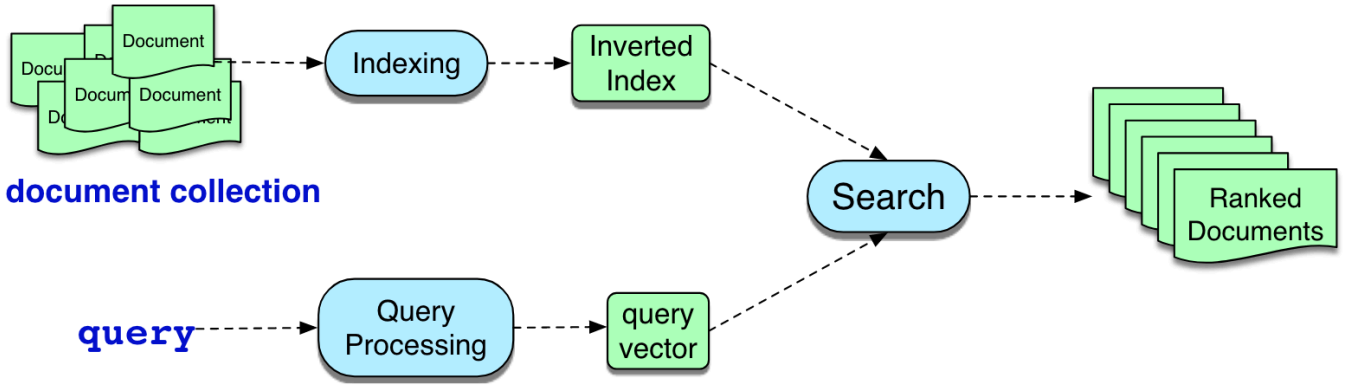
Thanks to Dan Jurafsky for most of the slides this week!

Plan for today

1. **Recap: Information retrieval powered by language models**
2. **Enc-dec Transformers**
3. **Machine Translation**
4. **T5: text-to-text everywhere**
5. ***No group exercise today (Marius says sorry)***

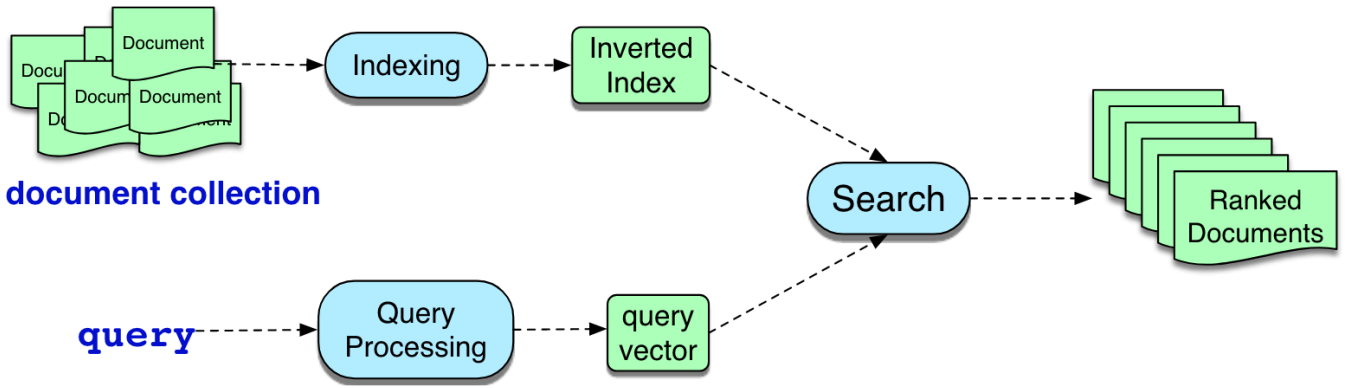
Information retrieval

- Retrieve *media* based on a user's *information need*
- We consider a special case of IR: **ad-hoc retrieval**
- A user provides a question to a system, which then returns one or more documents from a collection of documents



Information retrieval

- **Document**: the unit of text that the IR system indexes
- **Collection**: set of documents
- **Term**: a word (or paragraph) that occurs in the collection
- **Query**: information need expressed as a set of terms



Recall from Week 5

Embedding types

- **There are three main types of embeddings used in NLP and they are created using three different methods:**

  1. Sparse embeddings and term-document frequency

  2. Dense embeddings and Word2Vec algorithms

  3. Contextual word embeddings and language modeling

Recall from Week 5

Term frequency (tf) in the tf-idf algorithm

We could imagine using raw count:

$$\text{tf}_{t,d} = \text{count}(t,d)$$

But instead of using raw count, we usually squash a bit:

$$\text{tf}_{t,d} = \begin{cases} 1 + \log_{10} \text{count}(t,d) & \text{if } \text{count}(t,d) > 0 \\ 0 & \text{otherwise} \end{cases}$$

Recall from Week 5

Inverse document frequency (idf)

$$\text{idf}_t = \log_{10}\left(\frac{N}{\text{df}_t}\right)$$

df_t *is* the number of documents *t* occurs in.

(note this is not collection frequency: total count across all documents)

N is the total number of documents in the collection

tf-idf scoring

- Weighted value for each term

$$w_{t,d} = \text{tf}_{t,d} \times \text{idf}_f$$

- How do we score queries and documents?

$$\text{score}(q, d) = \cos(\mathbf{q}, \mathbf{d}) = \frac{\mathbf{q} \cdot \mathbf{d}}{|\mathbf{q}||\mathbf{d}|}$$

tf-idf scoring

$$\text{score}(q,d) = \cos(\mathbf{q},\mathbf{d}) = \frac{\mathbf{q}\cdot\mathbf{d}}{|\mathbf{q}||\mathbf{d}|}$$

- Rewrite the dot-product as a sum of products

$$\text{score}(q,d) = \sum_{t\in\mathbf{q}} \frac{\text{tf-idf}(t,q)}{\sqrt{\sum_{q_i\in q}\text{tf-idf}^2(q_i,q)}} \cdot \frac{\text{tf-idf}(t,d)}{\sqrt{\sum_{d_i\in d}\text{tf-idf}^2(d_i,d)}}$$

tf-idf scoring

$$\text{score}(q,d) = \sum_{t \in \mathbf{q}} \frac{\text{tf-idf}(t,q)}{\sqrt{\sum_{q_i \in q} \text{tf-idf}^2(q_i,q)}} \cdot \frac{\text{tf-idf}(t,d)}{\sqrt{\sum_{d_i \in d} \text{tf-idf}^2(d_i,d)}}$$

Query:	sweet love
Doc 1:	Sweet sweet nurse! Love?
Doc 2:	Sweet sorrow
Doc 3:	How sweet is love?
Doc 4:	Nurse!

	Query					
word	**cnt**	**tf**	**df**	**idf**	**tf-idf**	**n'lized** = tf-idf/$\lvert q \rvert$
sweet	1	1	3	0.125	0.125	0.383
nurse	0	0	2	0.301	0	0
love	1	1	2	0.301	0.301	0.924
how	0	0	1	0.602	0	0
sorrow	0	0	1	0.602	0	0
is	0	0	1	0.602	0	0
$\lvert q \rvert = \sqrt{.125^2 + .301^2} = .326$						

tf-idf scoring

$$\text{score}(q,d) = \sum_{t \in \mathbf{q}} \frac{\text{tf-idf}(t,q)}{\sqrt{\sum_{q_i \in q} \text{tf-idf}^2(q_i,q)}} \cdot \frac{\text{tf-idf}(t,d)}{\sqrt{\sum_{d_i \in d} \text{tf-idf}^2(d_i,d)}}$$

Query:	sweet love
Doc 1:	Sweet sweet nurse! Love?
Doc 2:	Sweet sorrow
Doc 3:	How sweet is love?
Doc 4:	Nurse!

	Document 1					Document 2				
word	**cnt**	**tf**	**tf-idf**	**n'lized**	**× q**	**cnt**	**tf**	**tf-idf**	**n'lized**	**×q**
sweet	2	1.301	0.163	0.357	**0.137**	1	1.000	0.125	0.203	**0.0779**
nurse	1	1.000	0.301	0.661	0	0	0	0	0	0
love	1	1.000	0.301	0.661	**0.610**	0	0	0	0	**0**
how	0	0	0	0	0	0	0	0	0	0
sorrow	0	0	0	0	0	1	1.000	0.602	0.979	0
is	0	0	0	0	0	0	0	0	0	0
$\lvert d_1 \rvert = \sqrt{.163^2 + .301^2 + .301^2} = .456$						$\lvert d_2 \rvert = \sqrt{.125^2 + .602^2} = .615$				
Cosine: $\sum$ of column:					**0.747**	Cosine: $\sum$ of column:				**0.0779**

BM25

$$\text{idf}_t = \log_{10}\left(\frac{N}{\text{df}_t}\right)$$

- Introduce parameters k and b

$$\text{score}(q,d) = \sum_{t \in q} \overbrace{\log\left(\frac{N}{df_t}\right)}^{\text{IDF}} \overbrace{\frac{tf_{t,d}}{k\left(1-b+b\left(\frac{|d|}{|d_{\text{avg}}|}\right)\right)+tf_{t,d}}}^{\text{weighted tf}}$$

Information retrieval

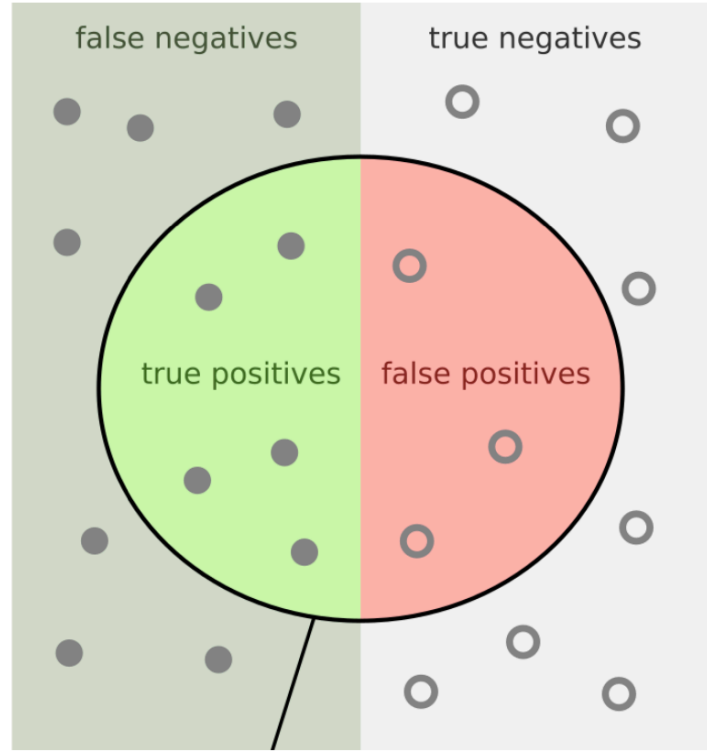
- Efficiently find documents that contain terms of interest
- **Inverted index**
  - Dictionary: list of terms (+ document frequency)
  - Postings list: list of document ids that contain the term (+ term frequency in each of the documents)

Query:	sweet love
Doc 1:	Sweet sweet nurse! Love?
Doc 2:	Sweet sorrow
Doc 3:	How sweet is love?
Doc 4:	Nurse!

how {1}	→	3 [1]
is {1}	→	3 [1]
love {2}	→	1 [1] → 3 [1]
nurse {2}	→	1 [1] → 4 [1]
sorry {1}	→	2 [1]
sweet {3}	→	1 [2] → 2 [1] → 3 [1]

Precision and Recall

- Heuristic classifiers tend to have high **precision** but very low **recall.**
- **Classifier accuracy is measured with precision and recall:**



How many relevant items are retrieved?

Recall = $\frac{\text{■}}{\text{■}}$

How many retrieved items are relevant?

Precision = $\frac{\text{■}}{\text{■}}$

Recall = TP / (**FN** +TP) Precision = TP / (**FP** +TP)

Precision and recall in IR

- Each document is either relevant or not relevant
- **Precision**: Fraction of retrieved documents that are relevant
- **Recall**: Fraction of all relevant documents that are retrieved
- Let T be the set of returned documents, R are the relevant documents in T, N are the irrelevant documents in T, U are all relevant documents in the collection

$$Precision = \frac{|R|}{|T|} \qquad Recall = \frac{|R|}{|U|}$$

Evaluation of information retrieval systems

- We care about the rank of the target document(s)
- **Precision@k:** fraction of relevant documents seen at a particular rank k
- **Recall@k:** fraction of relevant documents found at a particular rank k

Rank	Judgment	Precision_{Rank}	Recall_{Rank}
1	R	1.0	.11
2	N	.50	.11
3	R	.66	.22
4	N	.50	.22
5	R	.60	.33
6	R	.66	.44
7	N	.57	.44
8	R	.63	.55
9	N	.55	.55
10	N	.50	.55

Evaluation of information retrieval systems

- **Mean average precision** (MAP)
  - Iterate over the ranked list top to bottom
  - Note the precision **only** at positions where a relevant item has been encountered
  - Average these precisions over the return set
  - Formally: Let R_r be the set of relevant documents at or above rank r
  - **Average precision**:

$$\mathrm{AP} = \frac{1}{|R_r|} \sum_{d \in R_r} \mathrm{Precision}_r(d)$$

    - $\mathrm{Precision}_r(d)$ precision measured at rank where d was found

Evaluation of information retrieval systems

- **Mean average precision** (MAP)
  - **Average precision**:

$$\mathrm{AP} = \frac{1}{|R_r|} \sum_{d \in R_r} \mathrm{Precision}_r(d)$$

    - $\mathrm{Precision}_r(d)$ precision measured at rank where d was found
- Given a set of queries Q:

$$\mathrm{MAP} = \frac{1}{|Q|} \sum_{q \in Q} \mathrm{AP}(q)$$

Evaluation of information retrieval systems

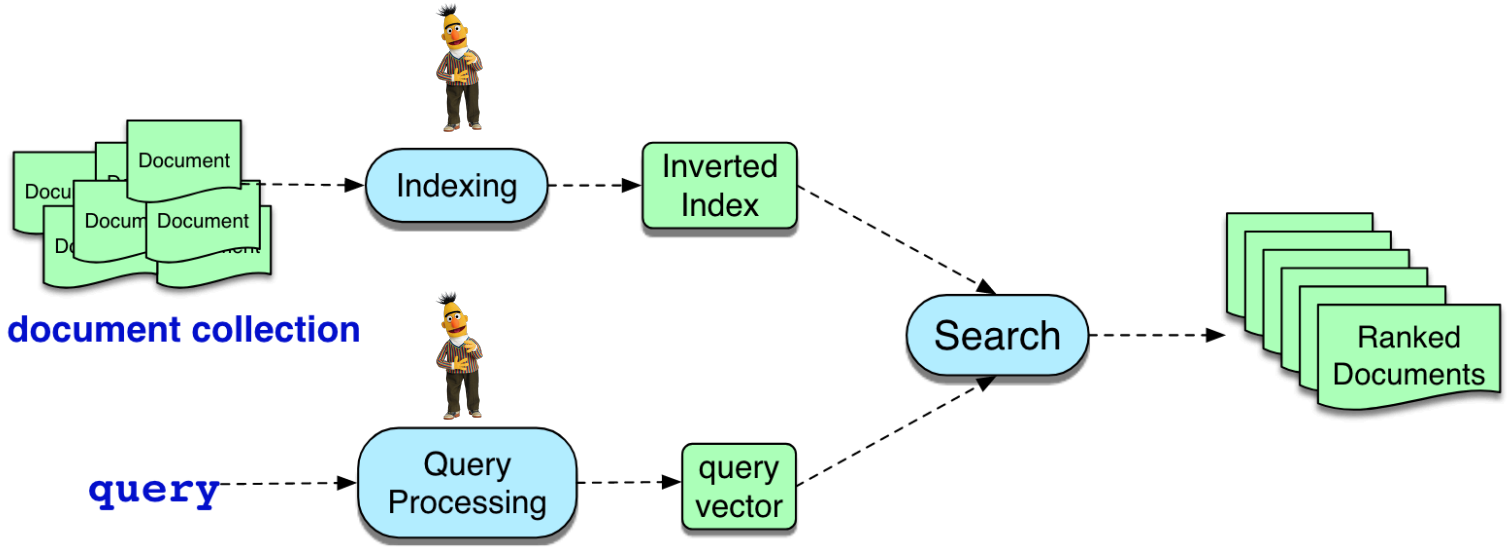
- **Mean average precision** (MAP)

$$\text{MAP} = \frac{1}{|Q|} \sum_{q \in Q} \text{AP}(q)$$

Rank	Judgment	Precision_{Rank}	Recall_{Rank}
1	R	1.0	.11
2	N	.50	.11
3	R	.66	.22
4	N	.50	.22
5	R	.60	.33
6	R	.66	.44
7	N	.57	.44
8	R	.63	.55
9	N	.55	.55
10	N	.50	.55

- Relevant docs at: 1, 3, 5, 6, 8
- Precisions: 1.0, 0.66, 0.60, 0.66, 0.63
- AP = (1.0 + 0.66 + 0.60 + 0.66 + 0.63) / 5 = 3.55 / 5 = 0.71

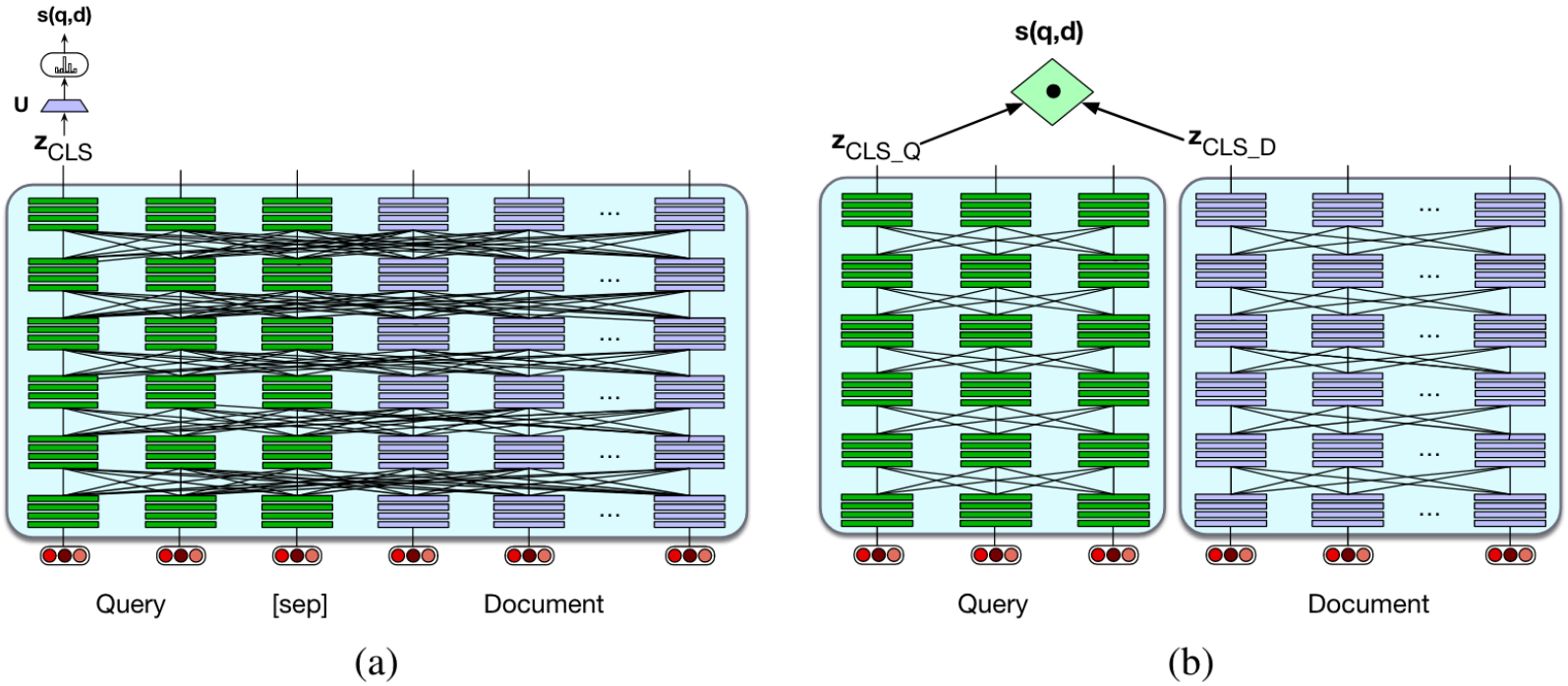
Information retrieval with pre-trained LMs



- Instead of using word-count vectors, use dense vectors from pre-trained language models, e.g., BERT

Information retrieval with pre-trained LMs

- Two potential ways to index queries and documents
  - Jointly encode query and each document
  - Bi-encoder: encode query and documents independently

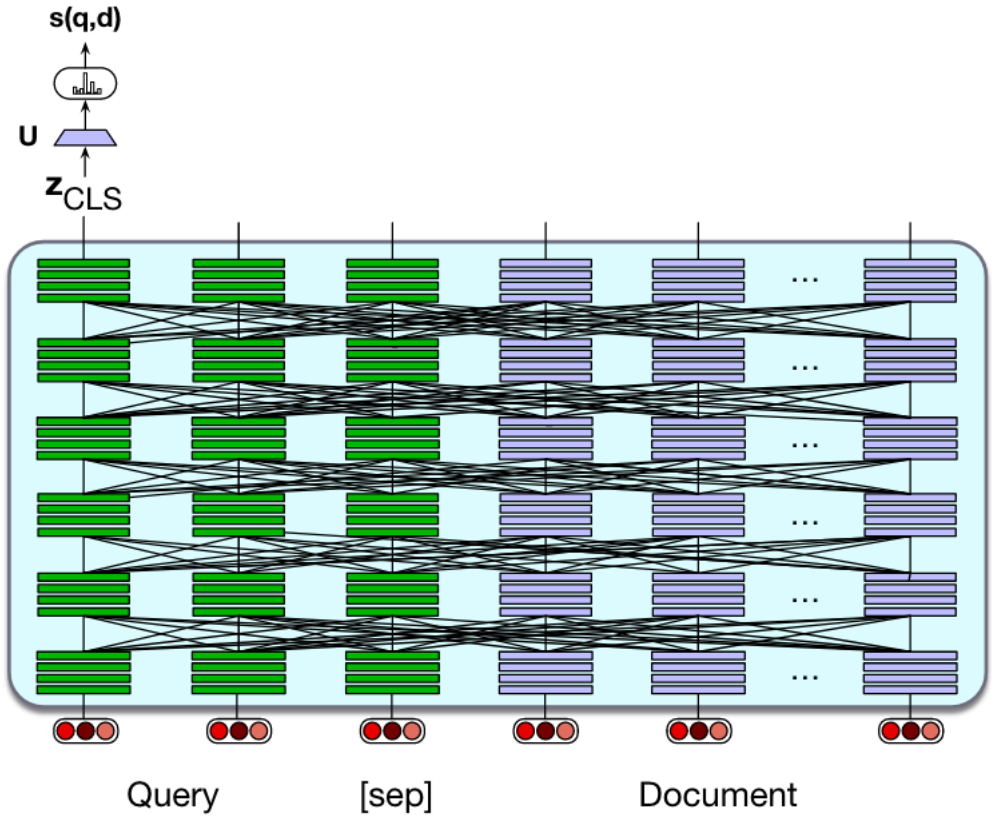


Information retrieval with pre-trained LMs

- Jointly encode query and every document

$$\mathbf{z} = \text{BERT}(q; [\text{SEP}]; d)[\text{CLS}]$$

$$\text{score}(\mathbf{z}) = \text{softmax}(\mathbf{U}\mathbf{z})$$

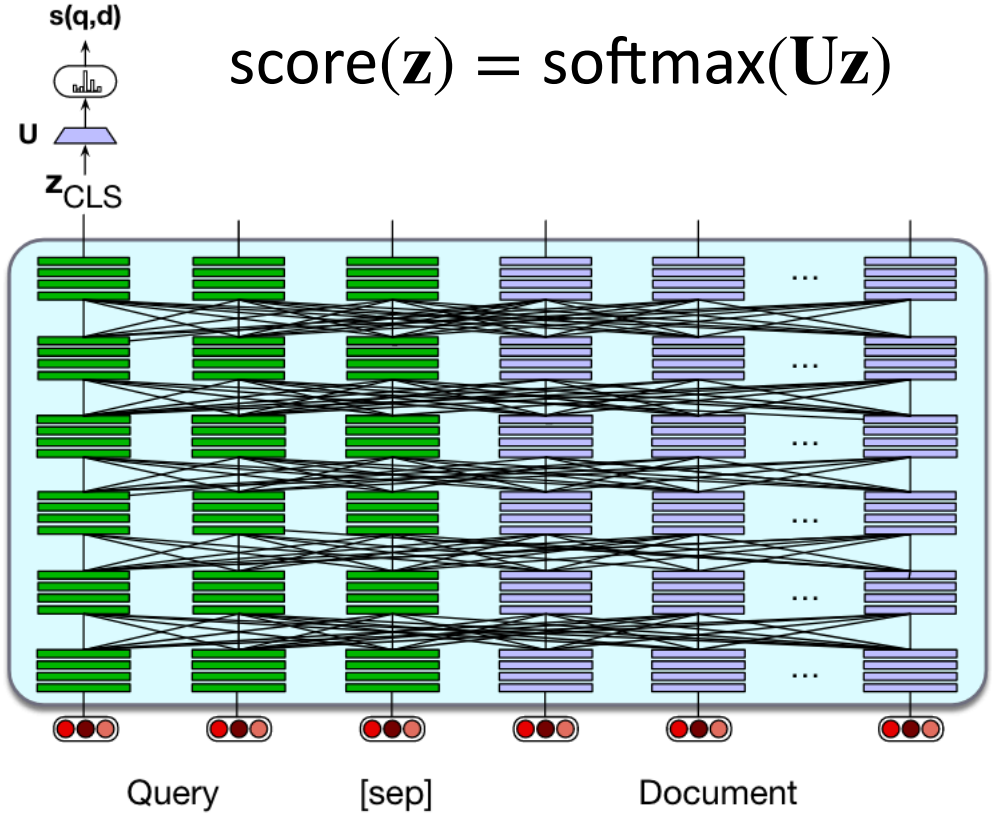


Information retrieval with pre-trained LMs

- This approach is expensive
- Need to re-encode every document for each query
- This quickly becomes impractical when dealing with large document collections

$$\mathbf{z} = \text{BERT}(q; [\text{SEP}]; d)[\text{CLS}]$$

$$\text{score}(\mathbf{z}) = \text{softmax}(\mathbf{U}\mathbf{z})$$



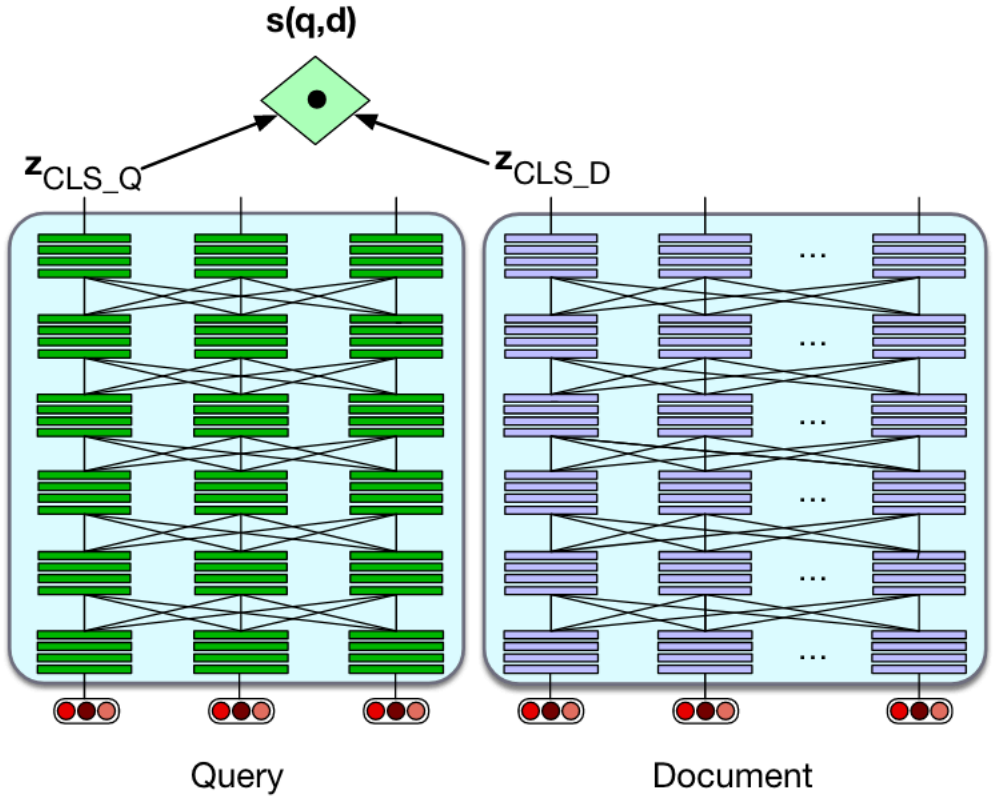
Information retrieval with pre-trained LMs

- **Bi-encoder**: encode queries and documents independently

$$\mathbf{z}_q = \text{BERT}_{\text{query}}(q)[\text{CLS}]$$

$$\mathbf{z}_d = \text{BERT}_{\text{doc}}(d)[\text{CLS}]$$

$$\text{score}(q, d) = \mathbf{z}_q \cdot \mathbf{z}_d$$



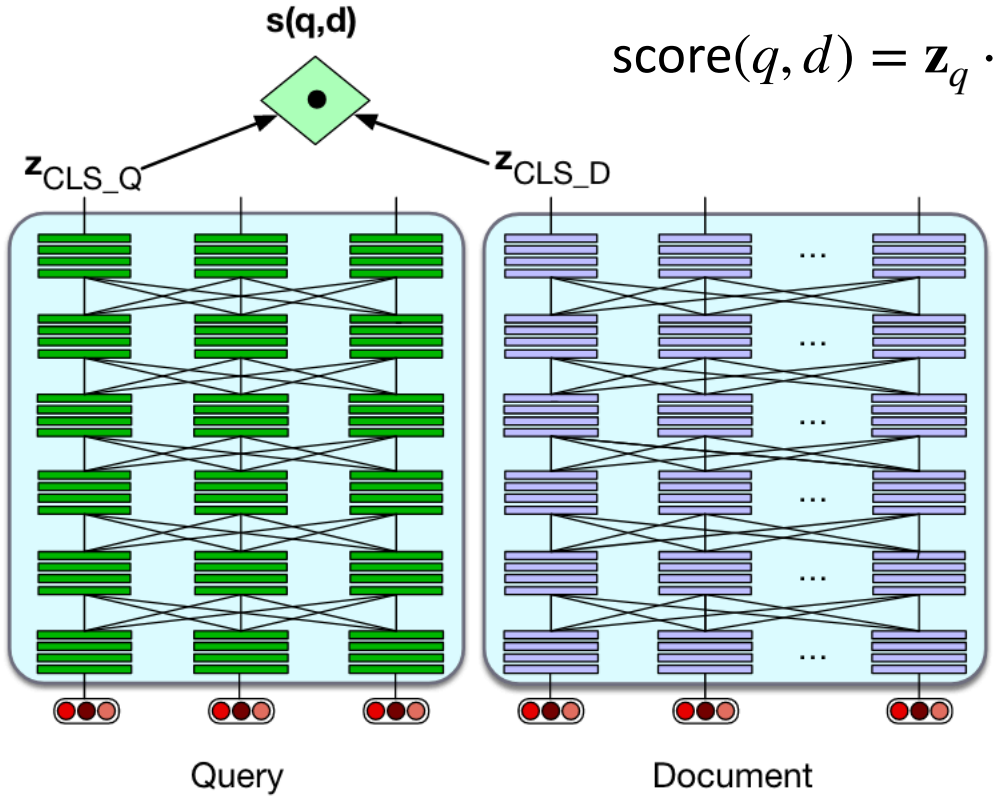
Information retrieval with pre-trained LMs

- Documents are encoded in advance
- When a new query comes in, we only have to encode that query
- Much more efficient but tends to give worse results
- Question: How to combine the two approaches?

$$\mathbf{z}_q = \text{BERT}_{\text{query}}(q)[\text{CLS}]$$

$$\mathbf{z}_d = \text{BERT}_{\text{doc}}(d)[\text{CLS}]$$

$$\text{score}(q, d) = \mathbf{z}_q \cdot \mathbf{z}_d$$



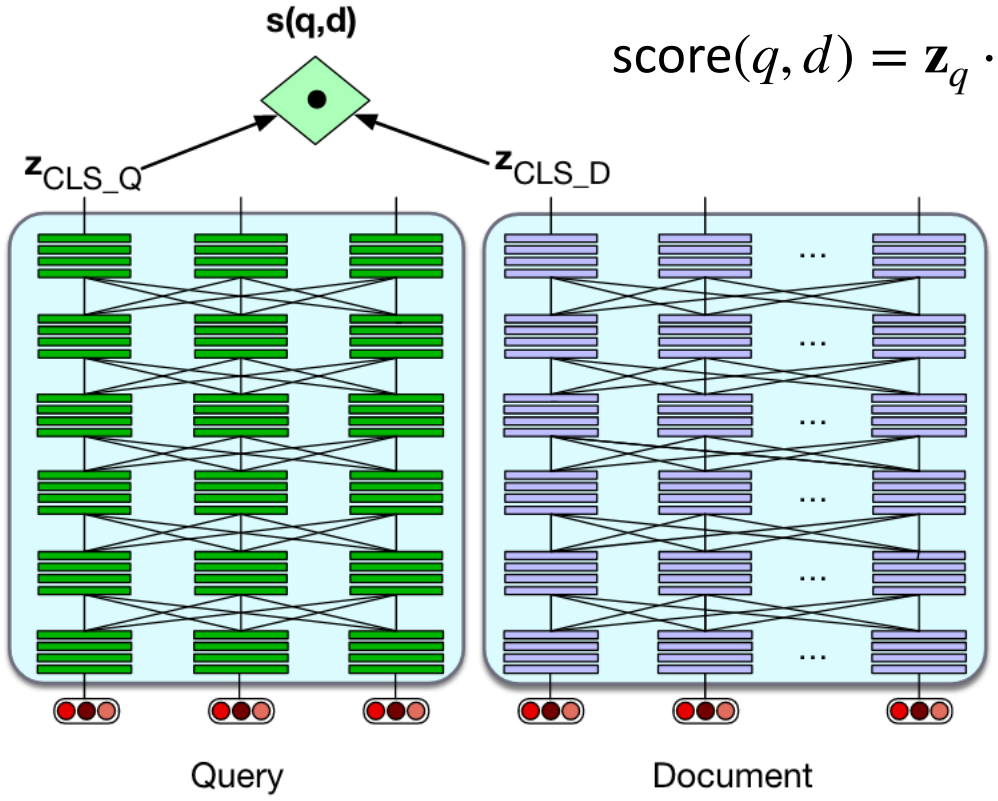
Information retrieval with pre-trained LMs

- Documents are encoded in advance
- When a new query comes in, we only have to encode that query
- Much more efficient but tends to give worse results
- Question: How to combine the two approaches?
  - **Re-ranking**

$$\mathbf{z}_q = \text{BERT}_{\text{query}}(q)[\text{CLS}]$$

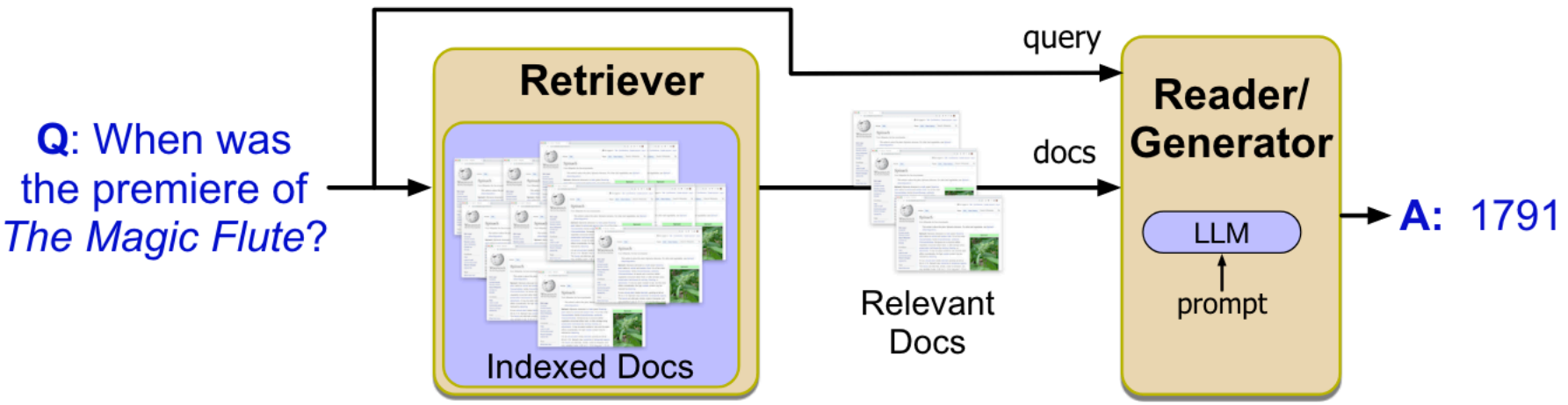
$$\mathbf{z}_d = \text{BERT}_{\text{doc}}(d)[\text{CLS}]$$

$$\text{score}(q, d) = \mathbf{z}_q \cdot \mathbf{z}_d$$



Information retrieval with pre-trained LMs

- Question Answering (QA) as a retrieval problem
- Find supporting documents for a query
- Extract or generate an answer based on the top-k docs



Information retrieval with pre-trained LMs

- Consider the document on the right
- There exist many queries for which this document might be relevant
  - When was HEC founded?
  - Can I do a PhD at HEC?
  - What's Viger Square?
  - Etc.
- Ideally, all of this needs to be encoded in the document representation

HEC Montréal (French: *Hautes études commerciales de Montréal*; English: *High Commercial Studies of Montreal*) is a bilingual public business school located in Montreal, Quebec, Canada. Founded in 1907, HEC Montréal is the graduate business school of the Université de Montréal and is the first established school of management in Canada.[2][3]

HEC Montréal offers undergraduate, graduate, and post-graduate programs, including Bachelor of Business Administration (BBA), Master of Science in Administration (MSc), Master of Management (MM), Master of Business Administration (MBA), and PhD in Administration, in addition to a joint Executive MBA program with McGill University.

HEC Montréal was founded in 1907 by the Board of Trade of Metropolitan Montreal. Its initial building in Viger Square is now called the Gilles Hocquart Building.[2]

In 1988, a group of HEC students established Jeux du Commerce, where more than 1300 students from 14 universities in Eastern Canada gather annually for academic, social, and sports events.[4][5] A similar competition has been established in Western Canada called JDC West.

As of 2021, the centenary of the HEC Montréal Alumni Association, the school had over 100,000 alumni.[6]

How to improve dense retrieval?

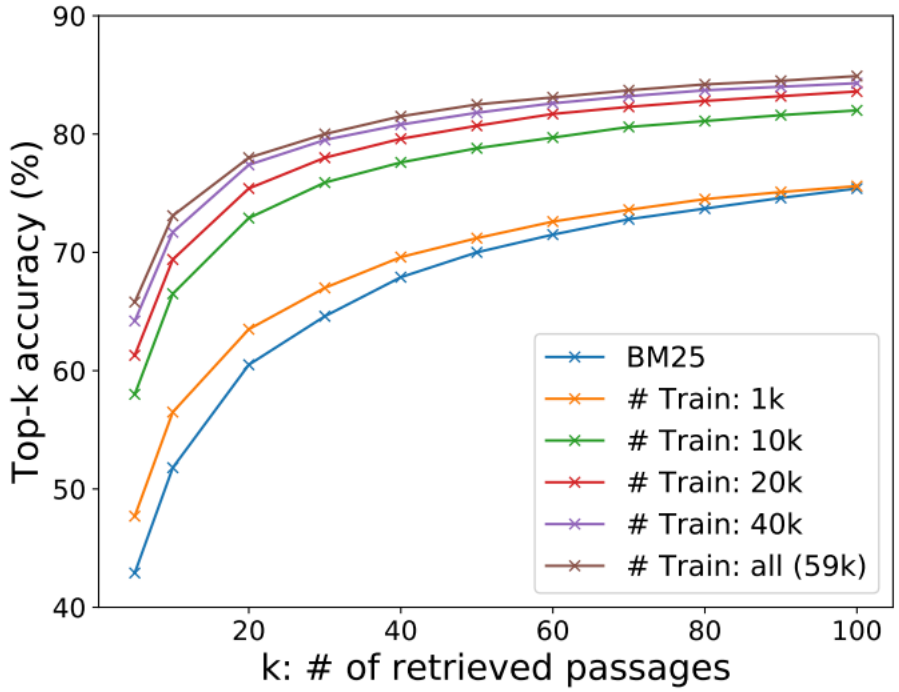
- Use a collection $\{(q_i, d_i^+, d_i^-)\}_{i=1}^n$ of queries paired with positive and negative documents
- Learn an implicit relevance definition based on this collection via **contrastive learning**
- Intuition: move positive documents close to query, push negative documents away from query

$$\mathcal{L}(q_i, d_i^+, \{d_{i,j}^-\}_{j=1}^k) = \frac{\exp(\mathbf{q} \cdot \mathbf{d}_i^+)}{\exp(\mathbf{q_i} \cdot \mathbf{d_i}^+) + \sum_{j=1}^k \exp(\mathbf{q}_i \cdot \mathbf{d}_{i,j}^-)}$$

Karpukhin et al. (2020) - Dense Passage Retrieval for Open-Domain Question Answering

Dense Passage Retrieval for Open-Domain QA

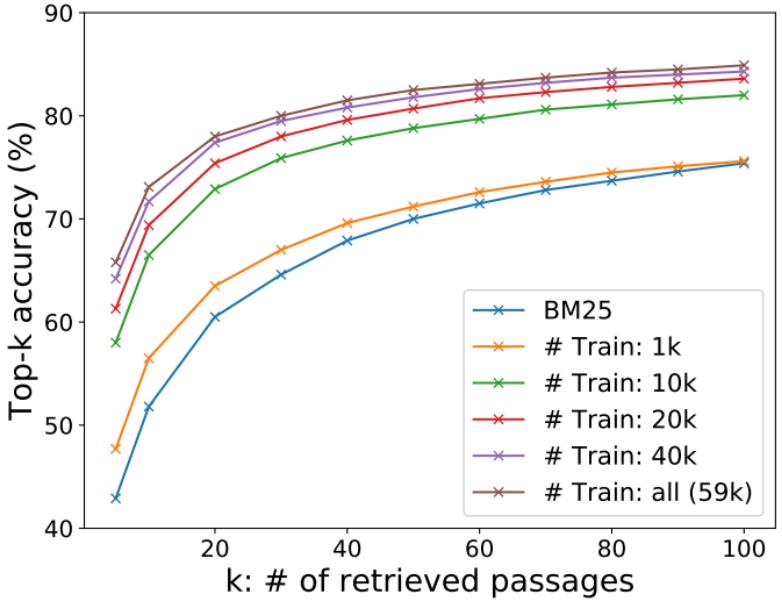
- How does it compare to BM25?

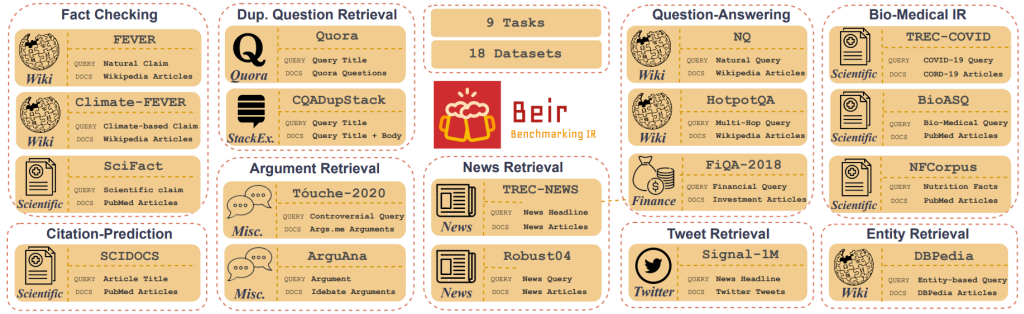


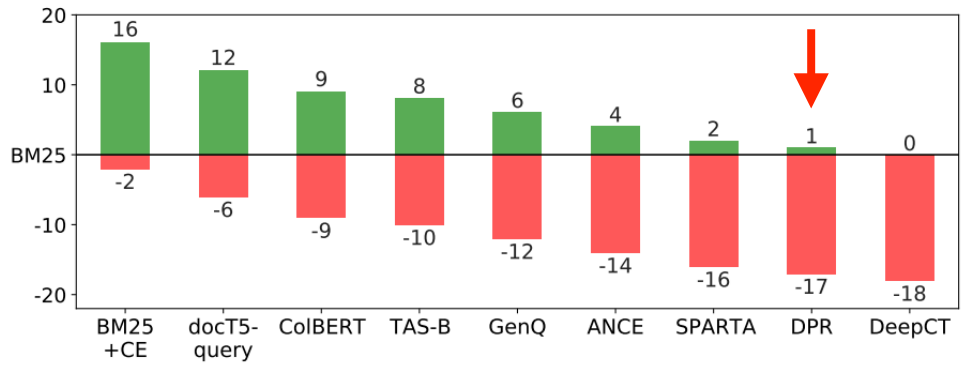
Karpukhin et al. (2020) - Dense Passage Retrieval for Open-Domain Question Answering

Dense Passage Retrieval for Open-Domain QA

- What happens outside of the training distribution?
- Question: what could be the reason?







Thakur et al. (2021) - BEIR: A Heterogenous Benchmark for Zero-shot Evaluation of Information Retrieval Models

Encoder-decoder Transformers

- Now we will see a different class of Transformer-based models that have been / are widely used

The paper that started it all

Recall from Week 8

Transformer: a specific kind of network architecture, like a fancier feedforward network, but based on attention

Attention Is All You Need

Ashish Vaswani[*]
Google Brain
avaswani@google.com

Noam Shazeer[*]
Google Brain
noam@google.com

Niki Parmar[*]
Google Research
nikip@google.com

Jakob Uszkoreit[*]
Google Research
usz@google.com

Llion Jones[*]
Google Research
llion@google.com

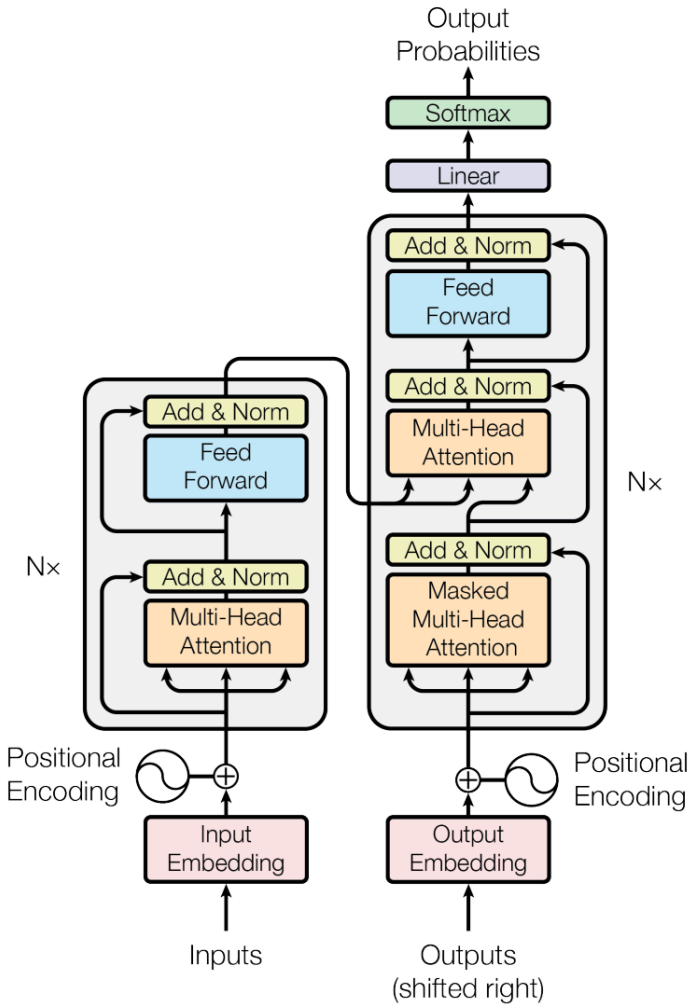
Aidan N. Gomez[*] [†]
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser[*]
Google Brain
lukaszkaiser@google.com

Illia Polosukhin[*] [‡]
illia.polosukhin@gmail.com

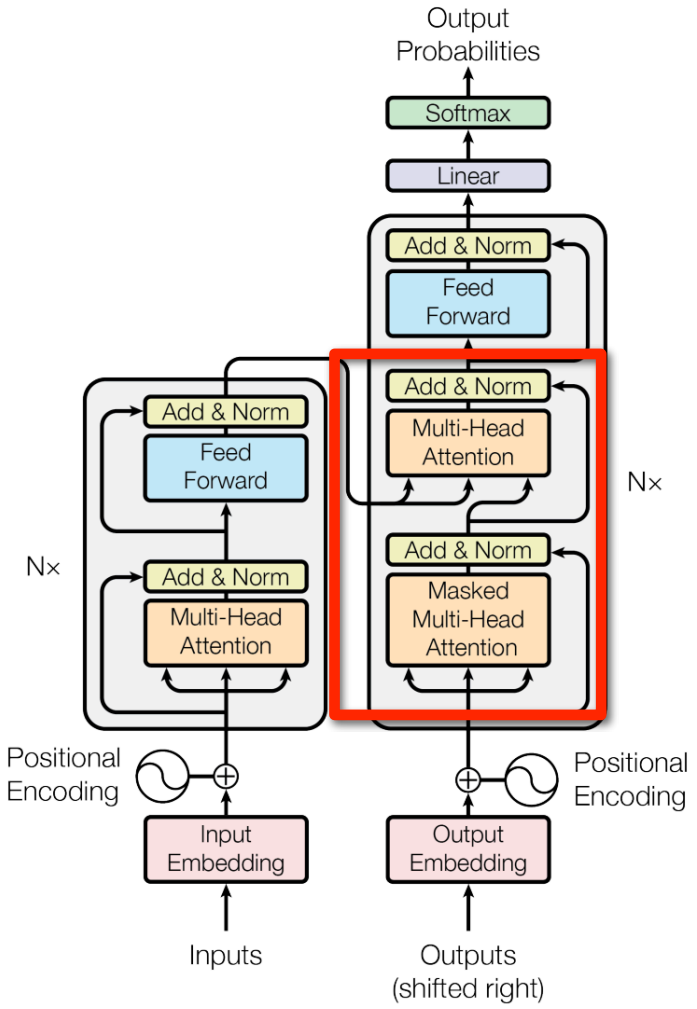
The original transformers was an enc-dec model

- **Encoder**
  - Bidirectional self-attention
- **Decoder**
  - Causal self-attention
- How to encoder and decoder ``communicate" with each other?
  - **Cross-attention**

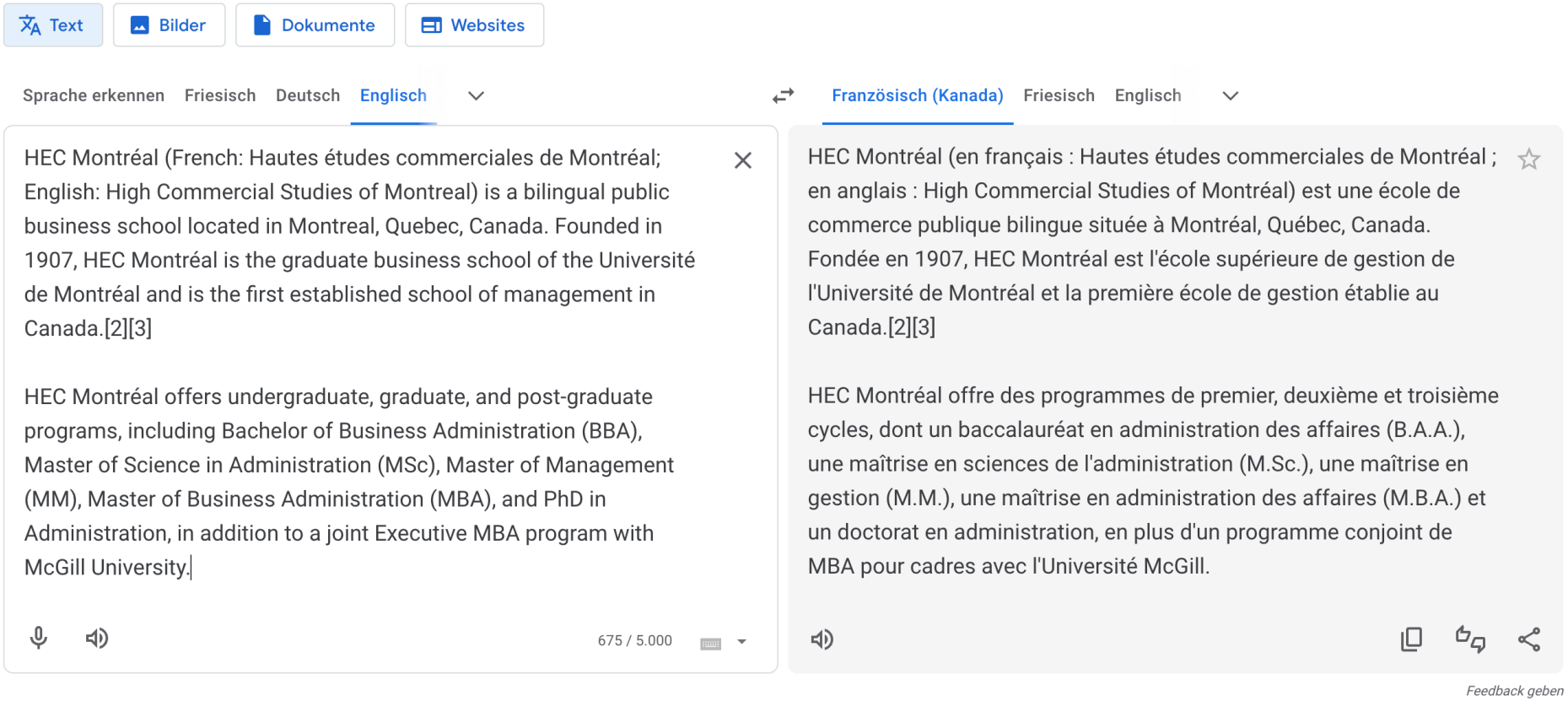


Cross attention

- The decoder has two self attention sub-layers instead of just one!
- Self-attention with $\mathbf{Q} = \mathbf{X}\mathbf{W}^Q$, $\mathbf{K} = \mathbf{X}\mathbf{W}^K$, and $\mathbf{V} = \mathbf{X}\mathbf{W}^V$
- Cross-attention with $\mathbf{Q} = \mathbf{X}\mathbf{W}^Q$, $\mathbf{K} = \mathbf{X}^L_{\text{enc}}\mathbf{W}^K$, and $\mathbf{V} = \mathbf{X}^L_{\text{enc}}\mathbf{W}^V$
- Keys and values are based on the representations from the last encoder layer!



Machine Translation via enc-dec models



Language diversity

- There are ~ 7000 languages in the world
- Some aspects of human language seem to be *universal*
  - Holding true for every language
  - *Statistically universal*: true for almost all languages
- E.g, every language seems to have nouns and verbs
- Yet, languages also **differ in many ways**
- This makes MT an interesting and challenging problem for language technology and research

Typology

- Sometimes differences between languages are systematic
- **Typology** studies these systematic cross-linguistic similarities and differences
- Example:
  - **Word order**
    - SVO
    - SOV
  - Who can think of example languages?

Typology

- Sometimes differences between languages are systematic
- **Typology** studies these systematic cross-linguistic similarities and differences
- Example:
  - **Word order**
    - SVO: French, English, German
    - SOV: Arabic
  - Who can think of example languages?

Typology

- Sometimes differences between languages are systematic
- **Typology** studies these systematic cross-linguistic similarities and differences
- Example:
  - **Word order**
    - SVO: French, English, German
    - SOV: Arabic
  - Who can think of example languages?

English: *He wrote a letter to a friend*
Japanese: *tomodachi ni tegami-o kaita*
friend to letter wrote
Arabic: *katabt risāla li ṣadq*
wrote letter to friend

Lexical divergences

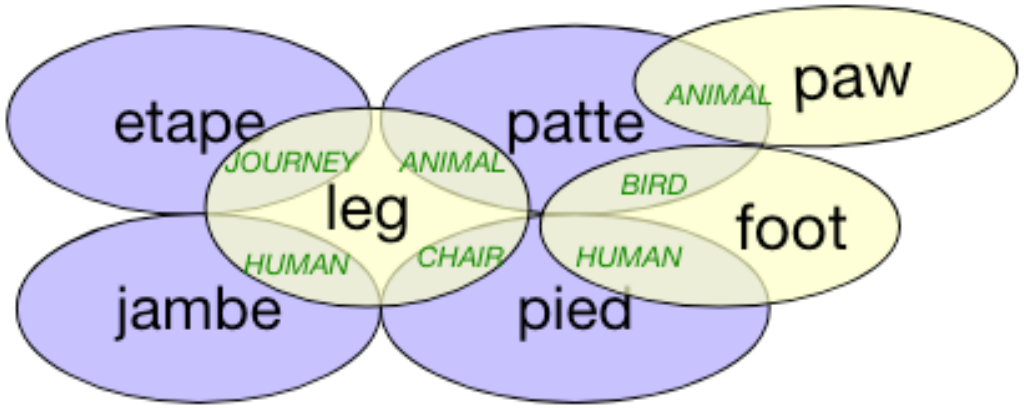
- When translating, we also need to map individual words from one the other
- But the appropriate word can vary depending on the *context*
- Example:
  - wall (English)
  - Mauer or Wand (German)
    - Wand is typically something inside

The Berlin Wall (Berliner Mauer)

Berliner Wand would sound really strange to any German

Lexical divergences

- How would you translate the English words *leg, paw, or foot* to French?



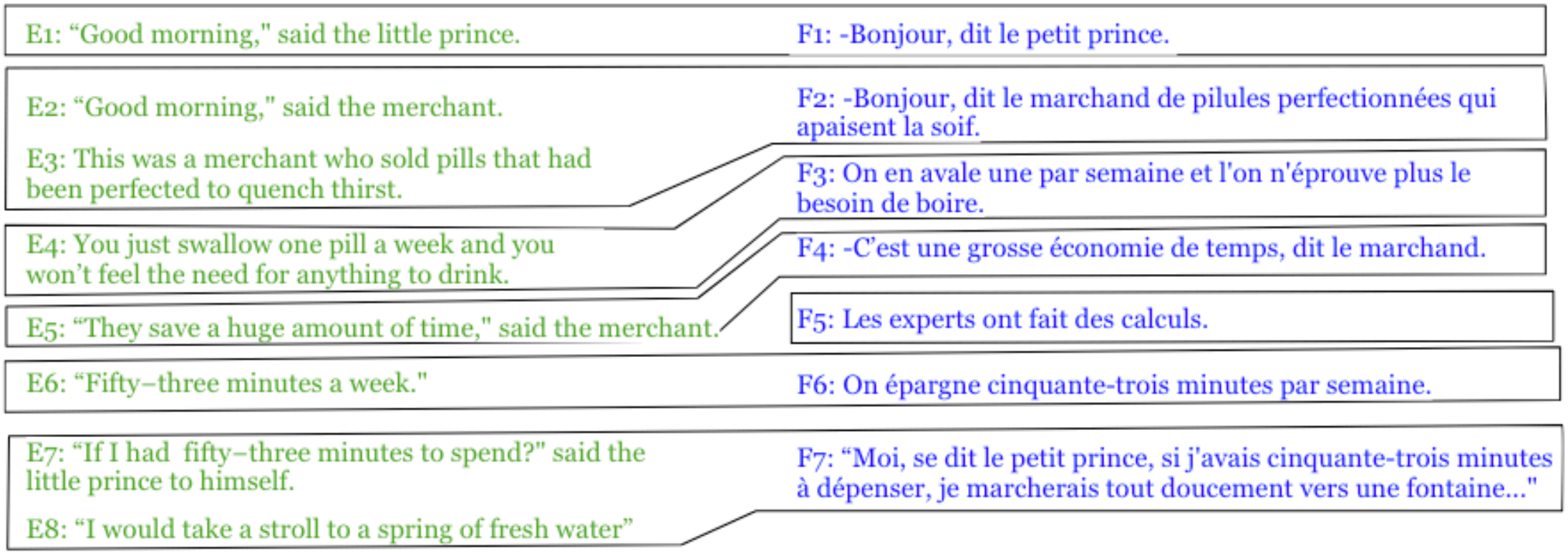
- Lexical gaps also exist
  - No word or phrase can express the exact meaning in another language

How should we tackle MT with enc-dec models?

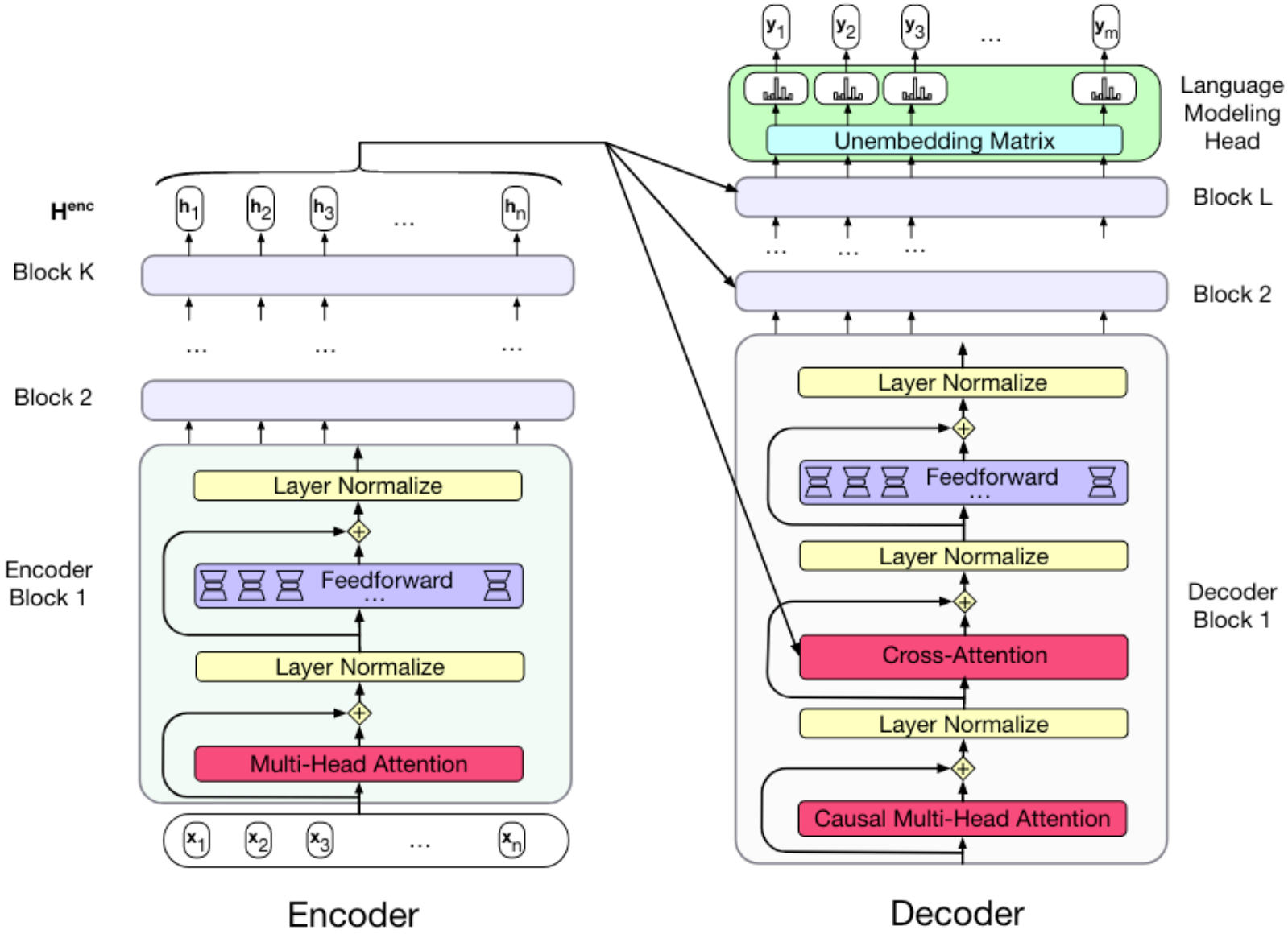
- Suggestions?
- We need to make some **assumptions** first
  - Often, MT systems assume that sentences are independent, i.e., we work with a corpus of **aligned** sentences in two (or more) languages
- What is your objective?
  - $P(y_1, \ldots, y_m \mid x_1, \ldots, x_n)$
  - Parameterized by an enc-dec neural network

$$\begin{aligned} \mathbf{h} &= \text{encoder}(x) \\ y_{t+1} &= \text{decoder}(\mathbf{h}, y_1, \ldots, y_t) \quad \forall t \in [1, \ldots, m] \end{aligned}$$

MT training data



The model architecture

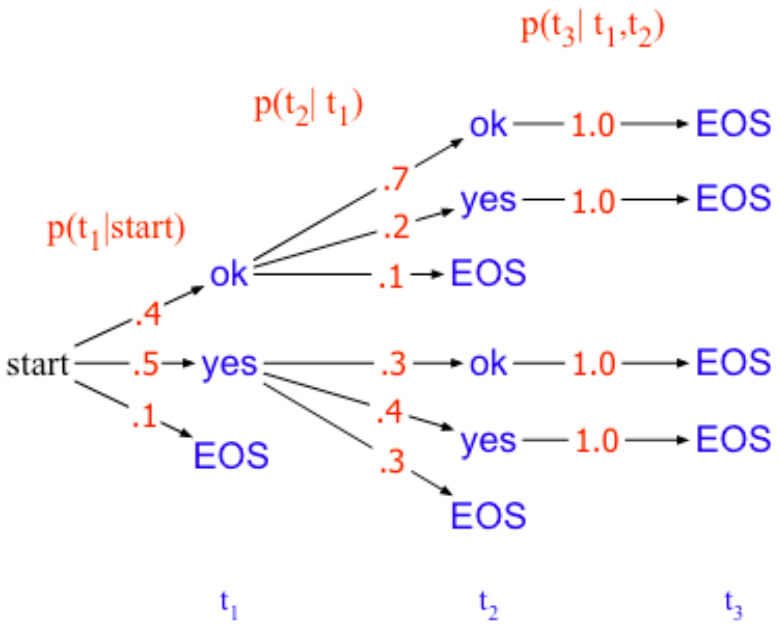


Decoding via beam-search

- So far, during decoding, we always picked the word with the highest probability (**greedy decoding**)
  - $\hat{w} = \text{argmax}_{w \in \mathcal{V}} P(w \mid \mathbf{w}_{<t})$
- Being greedy doesn't mean being optimal
- Beam search is an alternative method for decoding
- It considers multiple alternatives (hypotheses) in parallel

Decoding via beam-search

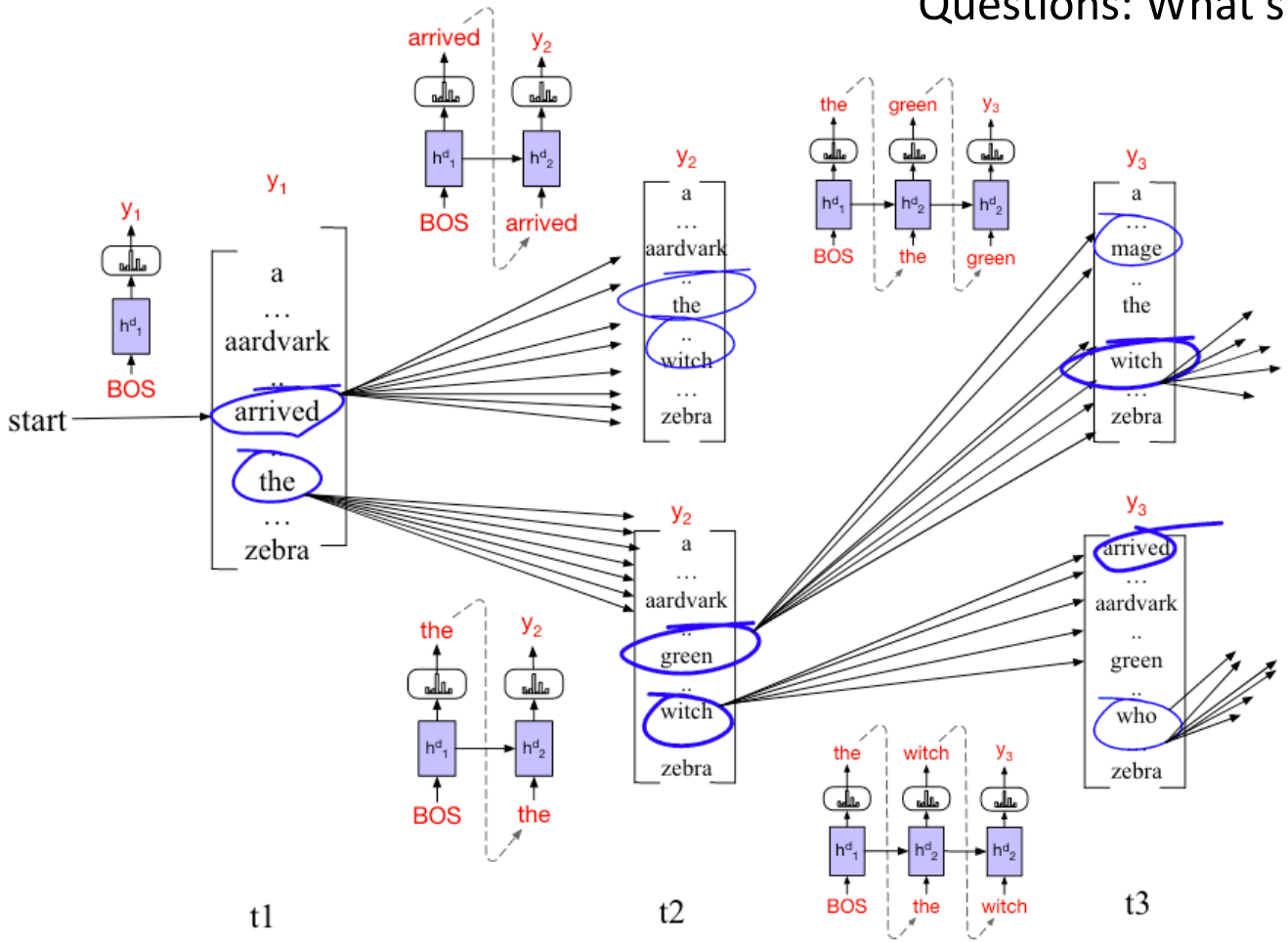
- In the toy problem on the right, the sequence with the largest **joint probability** is “ok ok EOS”
- Greedy-decoding is unable to decode that sequence
  - Can you see why?



Decoding via beam-search

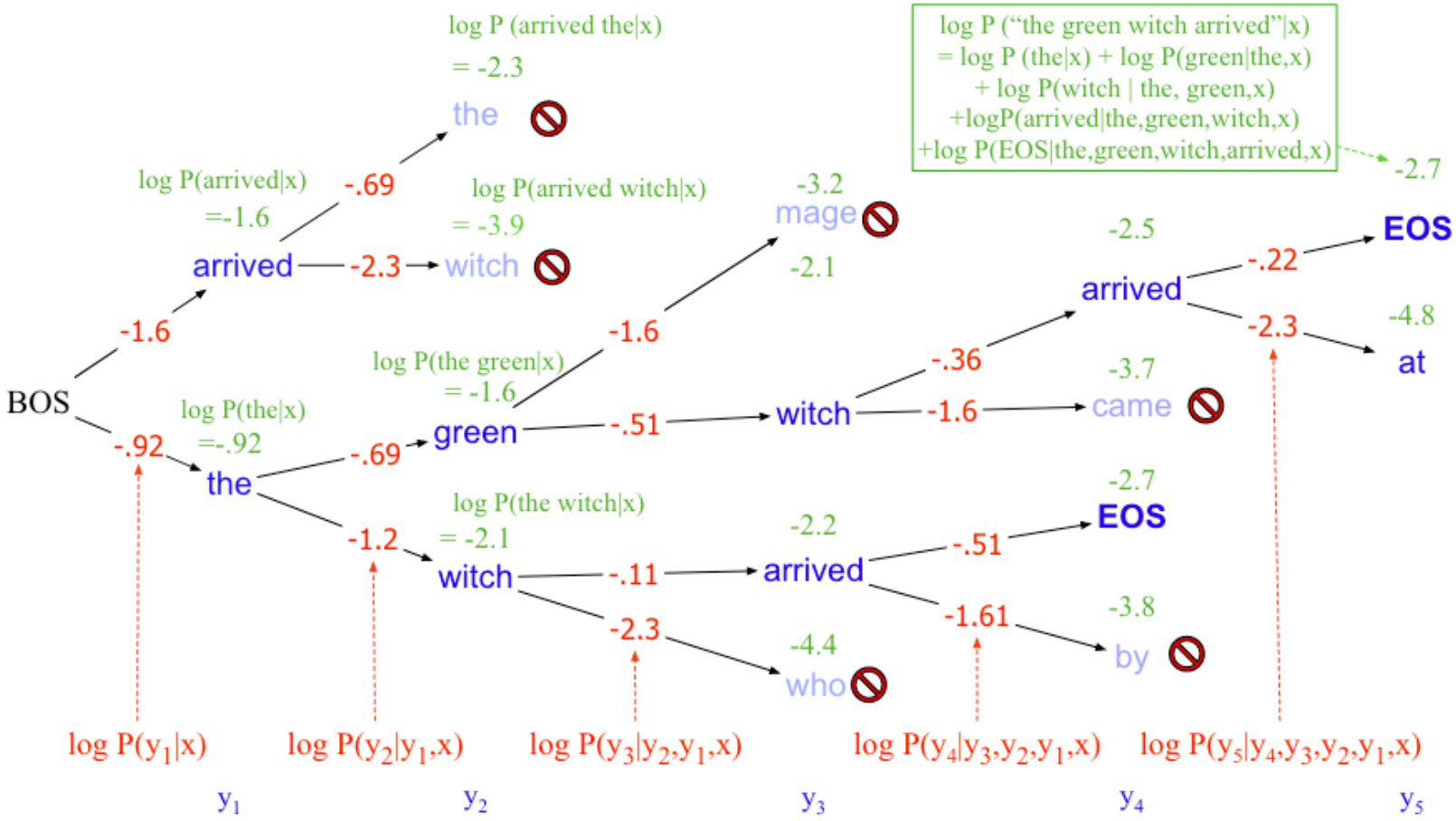
- At each step, consider k possible continuations (beam width)

Questions: What's the beam width here?



Scoring with Beam search

- We maintain the log probability at each step and sum them along the paths we expand



Evaluating MT systems

- What makes a good translation?
  - **Adequacy**
  - **Fluency**
- The gold standard ~> **Human evaluation**
  - Problem: can be expensive and slow
- We want a good **automatic evaluation**
  - Potentially less accurate but hopefully still provide a good signal to compare two systems

Evaluating MT systems

- **Character F-score (chrF)**
  - chrP: percentage of character 1-grams, 2-grams, …, k-grams in the hypothesis that also occur in the reference (averaged)
  - chrR: percentage of character 1-grams, 2-grams, …, k-grams in the reference that also occur in the hypothesis (averaged)
  - Compute an F-score by combining the two

$$\text{chrF}\beta = (1+\beta^2)\frac{\text{chrP}\cdot\text{chrR}}{\beta^2\cdot\text{chrP}+\text{chrR}} \qquad \text{chrF2} = \frac{5\cdot\text{chrP}\cdot\text{chrR}}{4\cdot\text{chrP}+\text{chrR}}$$

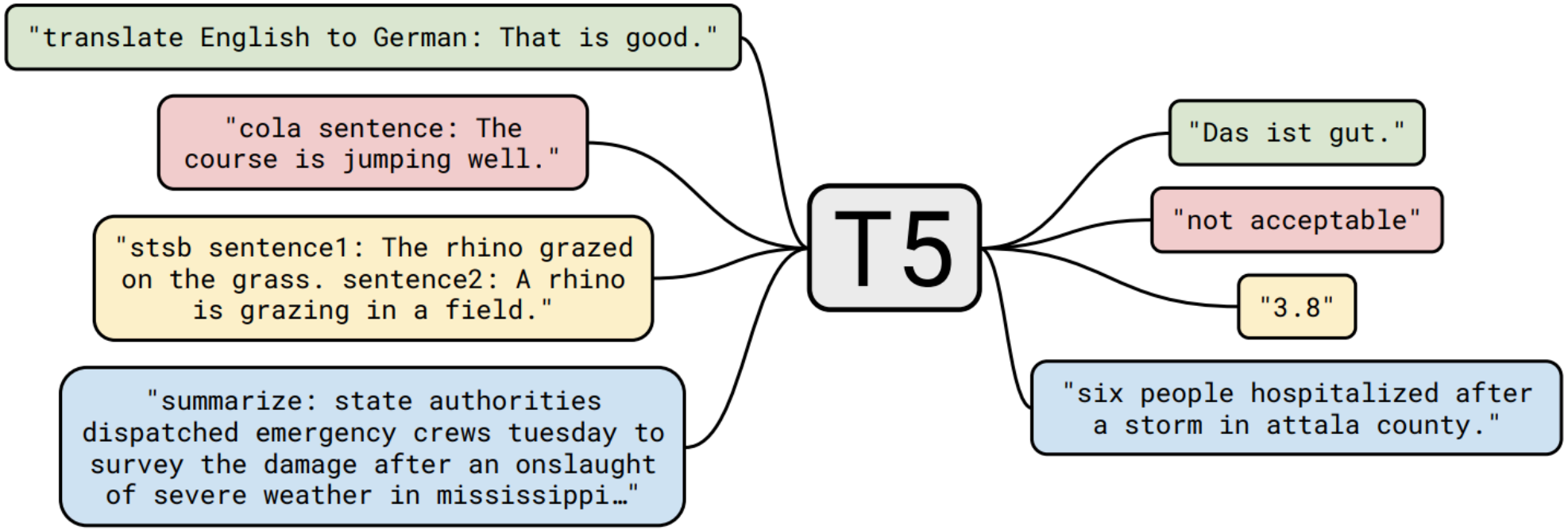
Evaluating MT systems

- **BLEU score**
  - Purely precision based
  - N-gram word precision
    - Consider a corpus $\mathscr{C}$ with a single sentence, i.e., $\mathscr{C} = \{s\}$ *(Sentence BLEU)*
    - Uni-gram precision: percentage of uni-grams in the candidate translation (hypothesis) than also occur in the corpus

Evaluating MT systems

- ***Sentence BLEU score***
  - Compute n-gram precision for uni-, bi-, tri-, and 4-grams and take the geometric mean
- Downsides
  - Word based, making it hard to compare systems with different tokenization
  - Doesn’t work well for languages with complex morphology
- Everyone in NLP knows that BLEU isn’t great but we still use it…

T5: seq-to-seq everywhere



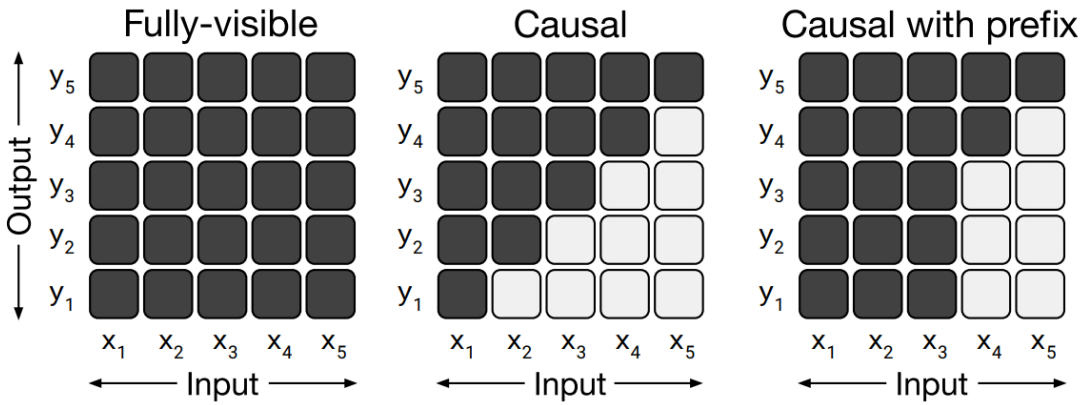
There are a few famous figures in our field, this is one of them :)

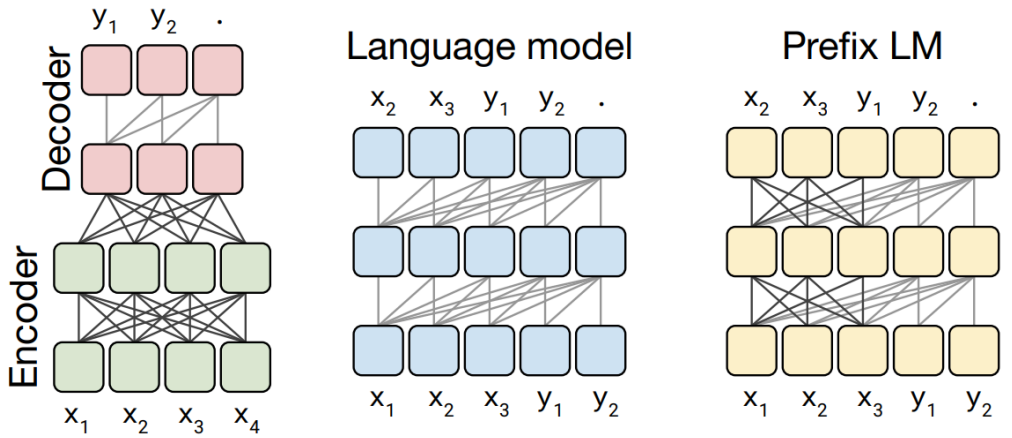
T5: seq-to-seq everywhere

- General **text-to-text** format
- Machine translation
  - Translate this text to French: Montréal is great. Montréal est génial
- Summarization
  - Summarize this news article: [Some article] [Summary of the article]
- What other tasks can you think of?
- Can you think of a NLP task that **can't** be formatted as text-to-text?

What's the best possible model for text-to-text leaning?

- **Model structure**
  - Which attention should we use?
    - Bi-directional
    - Causal
    - A mix of the two
- **Model architecture**
  - Encoder-decoder
  - Decoder-only
  - A mix of the two





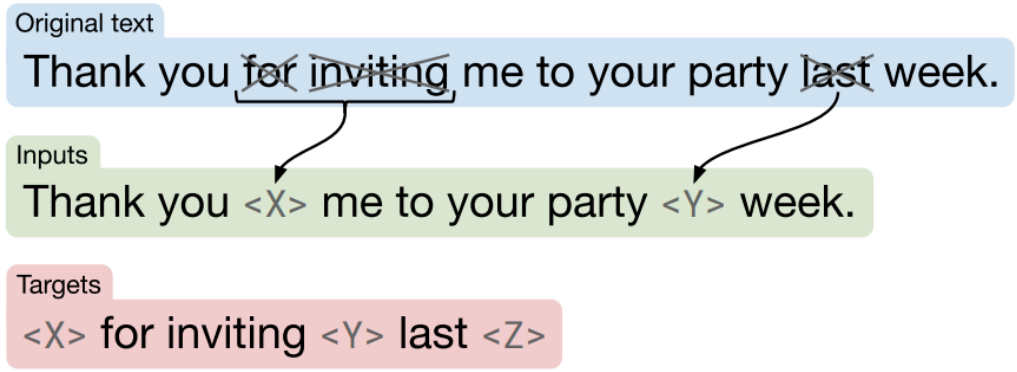
What's the best possible model for text-to-text leaning?

Architecture	Objective	Params	Cost	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Encoder-decoder	Denoising	$2P$	M	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Enc-dec, shared	Denoising	P	M	82.81	18.78	**80.63**	**70.73**	26.72	39.03	**27.46**
Enc-dec, 6 layers	Denoising	P	$M/2$	80.88	18.97	77.59	68.42	26.38	38.40	26.95
Language model	Denoising	P	M	74.70	17.93	61.14	55.02	25.09	35.28	25.86
Prefix LM	Denoising	P	M	81.82	18.61	78.94	68.11	26.43	37.98	27.39
Encoder-decoder	LM	$2P$	M	79.56	18.59	76.02	64.29	26.27	39.17	26.86
Enc-dec, shared	LM	P	M	79.60	18.13	76.35	63.50	26.62	39.17	27.05
Enc-dec, 6 layers	LM	P	$M/2$	78.67	18.26	75.32	64.06	26.13	38.42	26.89
Language model	LM	P	M	73.78	17.54	53.81	56.51	25.23	34.31	25.38
Prefix LM	LM	P	M	79.68	17.84	76.87	64.86	26.28	37.51	26.76

- Some combinations are better than others
- Denoising seems to be almost always better than LMing

What's the best possible model for text-to-text leaning?

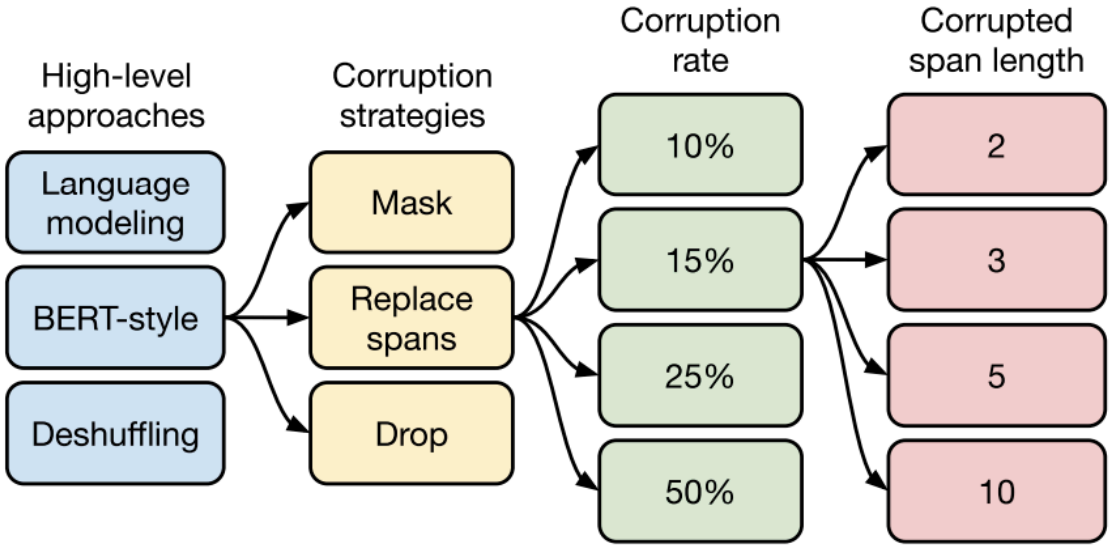
- Training objective
- What's the best way to encode useful information about and decode from a sequence?



Objective	Inputs	Targets
Prefix language modeling	Thank you for inviting	me to your party last week .
BERT-style Devlin et al. (2018)	Thank you **<M>** **<M>** me to your party apple week .	*(original text)*
Deshuffling	party me for your to . last fun you inviting week Thank	*(original text)*
MASS-style Song et al. (2019)	Thank you **<M>** **<M>** me to your party **<M>** week .	*(original text)*
I.i.d. noise, replace spans	Thank you **<X>** me to your party **<Y>** week .	**<X>** for inviting **<Y>** last **<Z>**
I.i.d. noise, drop tokens	Thank you me to your party week .	for inviting last
Random spans	Thank you **<X>** to **<Y>** week .	**<X>** for inviting me **<Y>** your party last **<Z>**

What's the best possible model for text-to-text leaning?

- Simply try all possible combinations



What's the best possible model for text-to-text leaning?

Objective	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
BERT-style (Devlin et al., 2018)	82.96	19.17	**80.65**	69.85	26.78	**40.03**	27.41
MASS-style (Song et al., 2019)	82.32	19.16	80.10	69.28	26.79	**39.89**	27.55
★ Replace corrupted spans	83.28	**19.24**	**80.88**	**71.36**	**26.98**	39.82	**27.65**
Drop corrupted tokens	**84.44**	**19.31**	**80.52**	68.67	**27.07**	39.76	**27.82**

Corruption rate	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
10%	**82.82**	19.00	**80.38**	69.55	**26.87**	39.28	**27.44**
★ 15%	**83.28**	19.24	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
25%	**83.00**	**19.54**	**80.96**	70.48	**27.04**	**39.83**	**27.47**
50%	81.27	19.32	79.80	70.33	**27.01**	**39.90**	**27.49**

Span length	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Baseline (i.i.d.)	**83.28**	19.24	80.88	71.36	**26.98**	**39.82**	**27.65**
2	**83.54**	19.39	**82.09**	**72.20**	**26.76**	**39.99**	**27.63**
3	**83.49**	**19.62**	**81.84**	**72.53**	**26.86**	39.65	**27.62**
5	**83.40**	19.24	**82.05**	**72.23**	**26.88**	39.40	**27.53**
10	82.85	19.33	**81.84**	70.44	**26.79**	39.49	**27.69**

- Each of these matters to some extent

What’s the best possible model for text-to-text leaning?

- One important ingredient is missing.
- Can you guess which one?

What's the best possible model for text-to-text leaning?

- One important ingredient is missing
- Can you guess which one?

Data

What's the best possible model for text-to-text leaning?

- Data 
  - C4: Colossal Clean Crawled Corpus

Data set	Size	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ C4	745GB	83.28	**19.24**	80.88	71.36	**26.98**	**39.82**	**27.65**
C4, unfiltered	6.1TB	81.46	19.14	78.78	68.04	26.55	39.34	27.21
RealNews-like	35GB	**83.83**	**19.23**	80.39	72.38	**26.75**	**39.90**	**27.48**
WebText-like	17GB	**84.03**	**19.31**	**81.42**	71.40	**26.80**	**39.74**	**27.59**
Wikipedia	16GB	81.85	**19.31**	81.29	68.01	**26.94**	39.69	**27.67**
Wikipedia + TBC	20GB	83.65	**19.28**	**82.08**	**73.24**	**26.77**	39.63	**27.57**

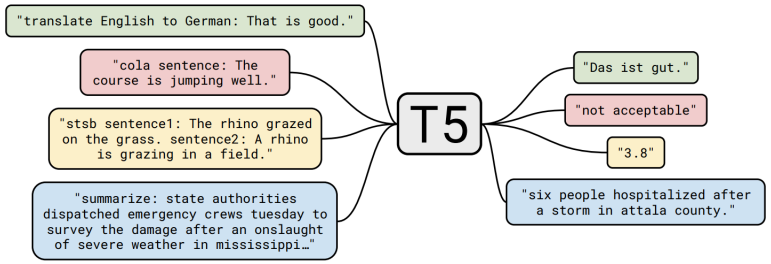
Number of tokens	Repeats	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ Full data set	0	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
2^{29}	64	**82.87**	**19.19**	**80.97**	**72.03**	**26.83**	**39.74**	**27.63**
2^{27}	256	82.62	**19.20**	79.78	69.97	**27.02**	**39.71**	27.33
2^{25}	1,024	79.55	18.57	76.27	64.76	26.38	39.56	26.80
2^{23}	4,096	76.34	18.33	70.92	59.29	26.37	38.84	25.81

What's the best possible model for text-to-text leaning?

- **Fine-tuning**
- Recall from last week:
  - You can choose which weights to update
  - Updating all weights seems to works better for T5

Fine-tuning method	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ All parameters	**83.28**	**19.24**	**80.88**	**71.36**	**26.98**	**39.82**	**27.65**
Adapter layers, $d = 32$	80.52	15.08	79.32	60.40	13.84	17.88	15.54
Adapter layers, $d = 128$	81.51	16.62	79.47	63.03	19.83	27.50	22.63
Adapter layers, $d = 512$	81.54	17.78	79.18	64.30	23.45	33.98	25.81
Adapter layers, $d = 2048$	81.51	16.62	79.47	63.03	19.83	27.50	22.63
Gradual unfreezing	82.50	18.95	79.17	**70.79**	26.71	39.02	26.93

Overall performance

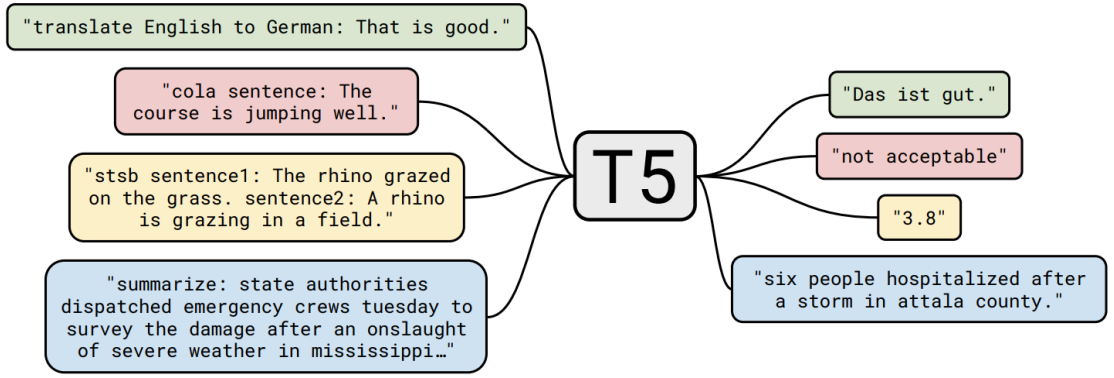


Model	GLUE Average	CoLA Matthew's	SST-2 Accuracy	MRPC F1	MRPC Accuracy	STS-B Pearson	STS-B Spearman
Previous best	89.4[a]	69.2[b]	97.1[a]	**93.6**[b]	**91.5**[b]	92.7[b]	92.3[b]
T5-Small	77.4	41.0	91.8	89.7	86.6	85.6	85.0
T5-Base	82.7	51.1	95.2	90.7	87.5	89.4	88.6
T5-Large	86.4	61.2	96.3	92.4	89.9	89.9	89.2
T5-3B	88.5	67.1	97.4	92.5	90.0	90.6	89.8
T5-11B	**90.3**	**71.6**	**97.5**	92.8	90.4	**93.1**	**92.8**

Model	QQP F1	QQP Accuracy	MNLI-m Accuracy	MNLI-mm Accuracy	QNLI Accuracy	RTE Accuracy	WNLI Accuracy
Previous best	74.8[c]	**90.7**[b]	91.3[a]	91.0[a]	**99.2**[a]	89.2[a]	91.8[a]
T5-Small	70.0	88.0	82.4	82.3	90.3	69.9	69.2
T5-Base	72.6	89.4	87.1	86.2	93.7	80.1	78.8
T5-Large	73.9	89.9	89.9	89.6	94.8	87.2	85.6
T5-3B	74.4	89.7	91.4	91.2	96.3	91.1	89.7
T5-11B	**75.1**	90.6	**92.2**	**91.9**	96.9	**92.8**	**94.5**

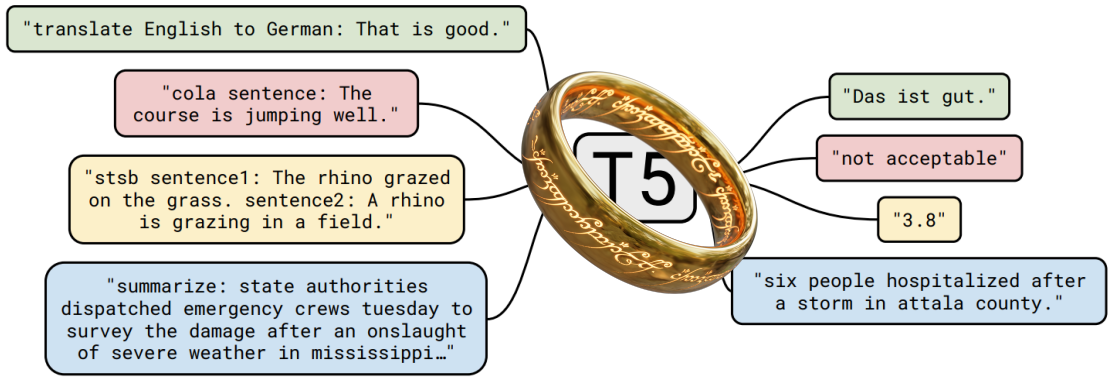
- Really good across the board. New SOTA on several tasks.
- More results in the paper

T5: seq-to-seq everywhere



- Exhaustive search over the space of possible seq-to-seq models
- Very strong enc-dec model for NLP downstream tasks

T5: seq-to-seq everywhere



- Paradigm-shift: **One model to rule them all?**
- Preview for next week:
  - ~> This can be simplified and pushed even further …

[15 minute break]