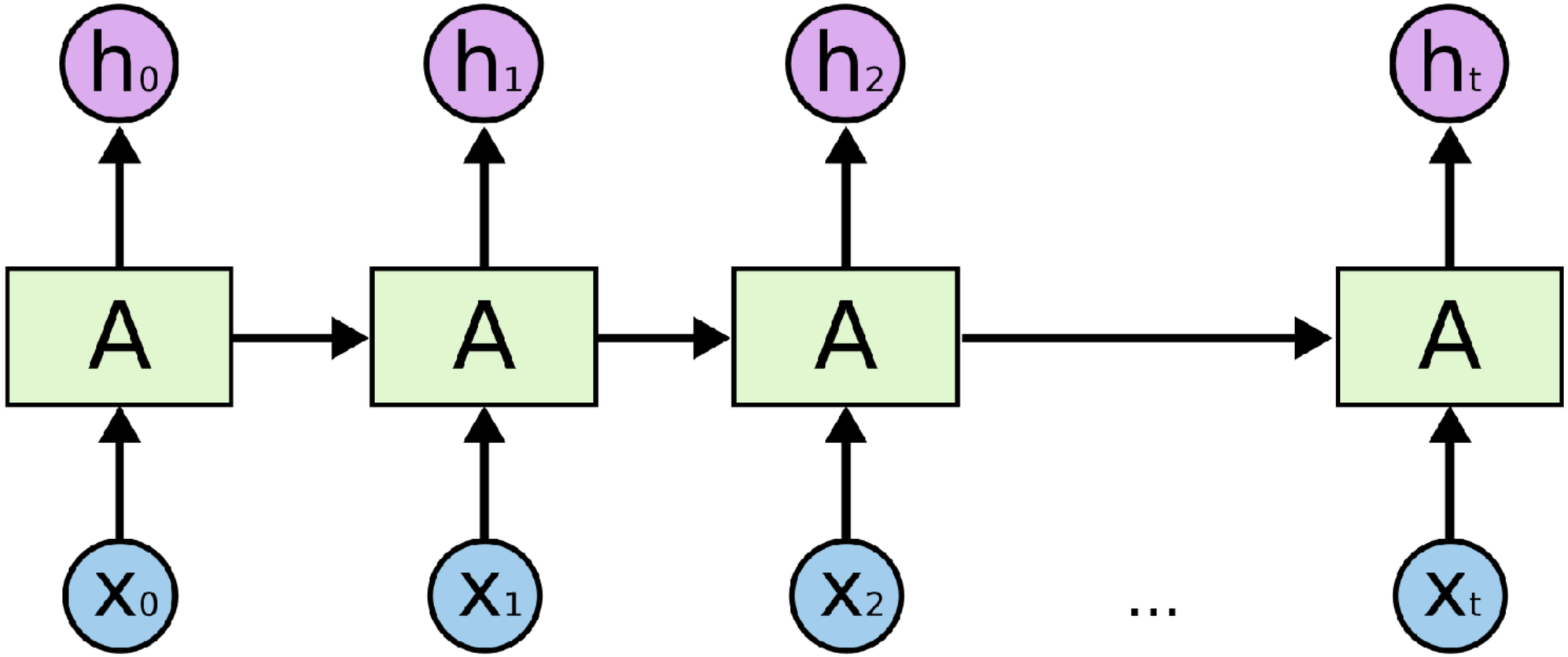
COMP0087 25/26
Lecture 5: Neural Language Models (cont'd)

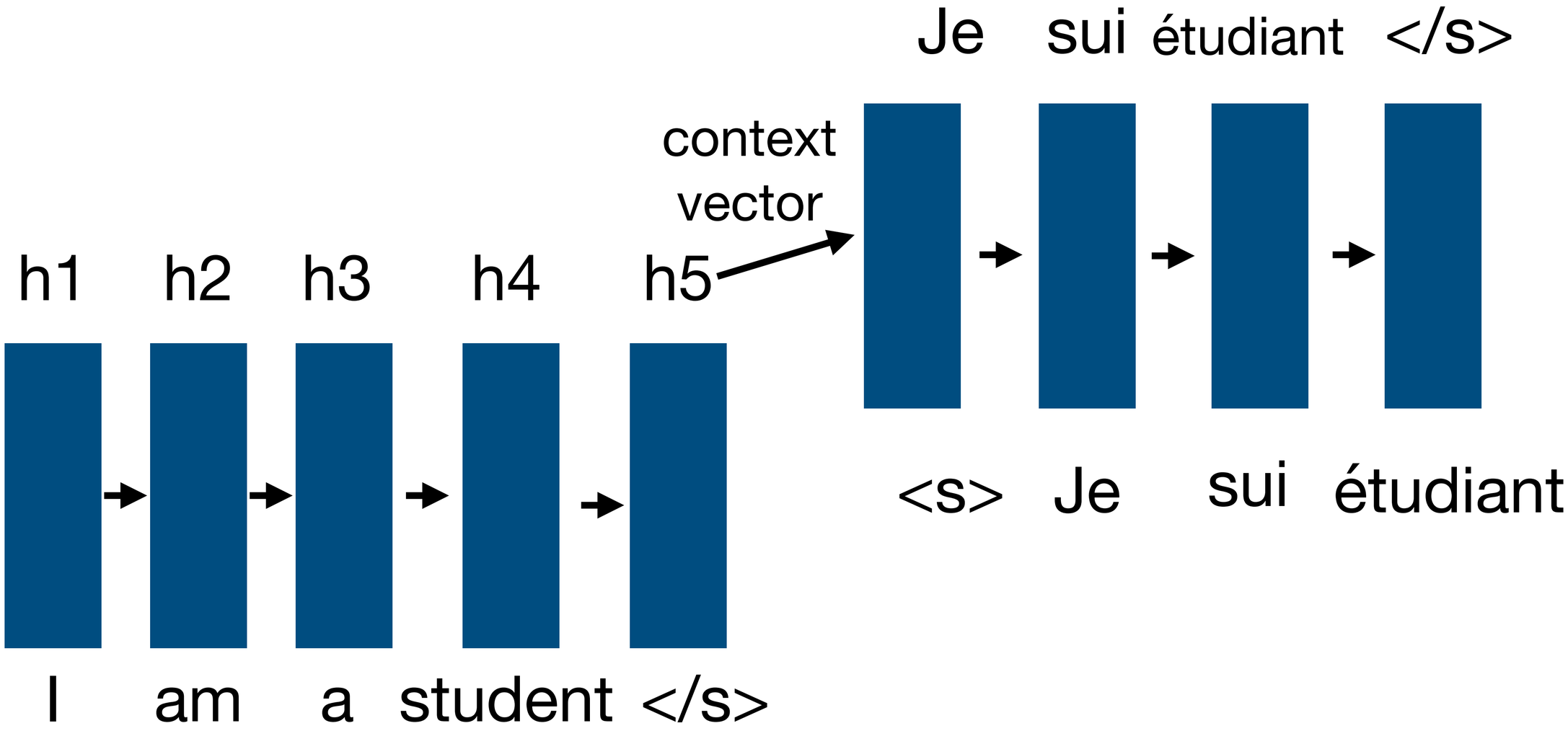
27/01/2026

comp0087.github.io

Recap: Neural Language Model



Sequence-to-sequence model



What can be wrong?

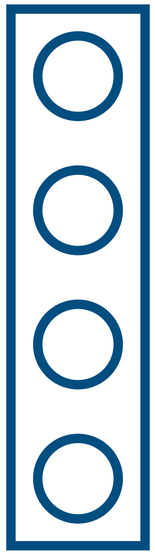
I am a student.

I am a student studying natural language processing at UCL

I am a student studying natural language processing at UCL where I also work part-time as a teaching assistant helping other students with their neural language modelling coursework.

What can be wrong?

I am a student.

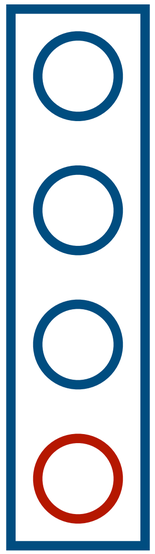


Je suis étudiant.

hidden_dim = 128

What can be wrong?

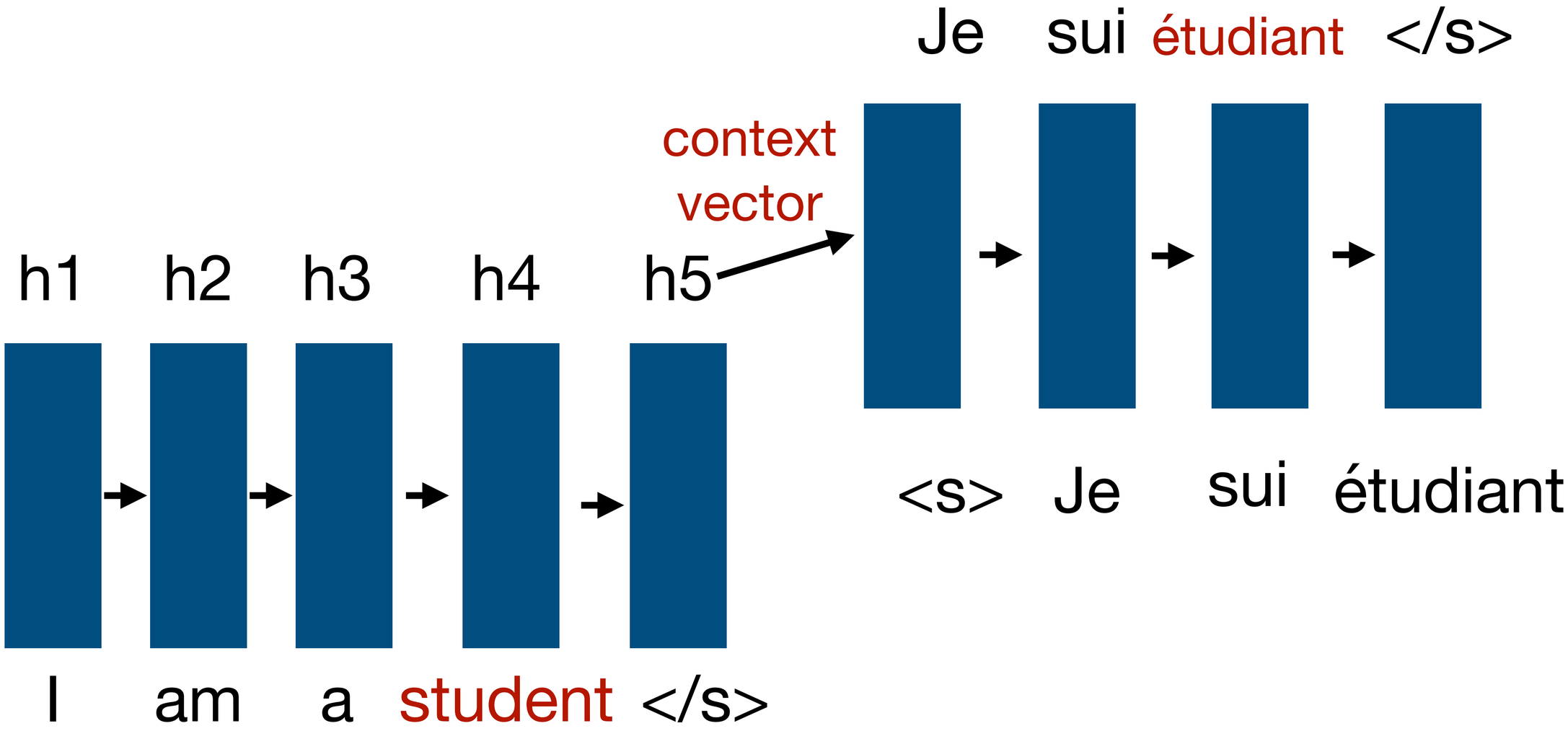
I am a student studying natural language processing at UCL where I also work part-time as a teaching assistant helping other students with their neural language modelling coursework.



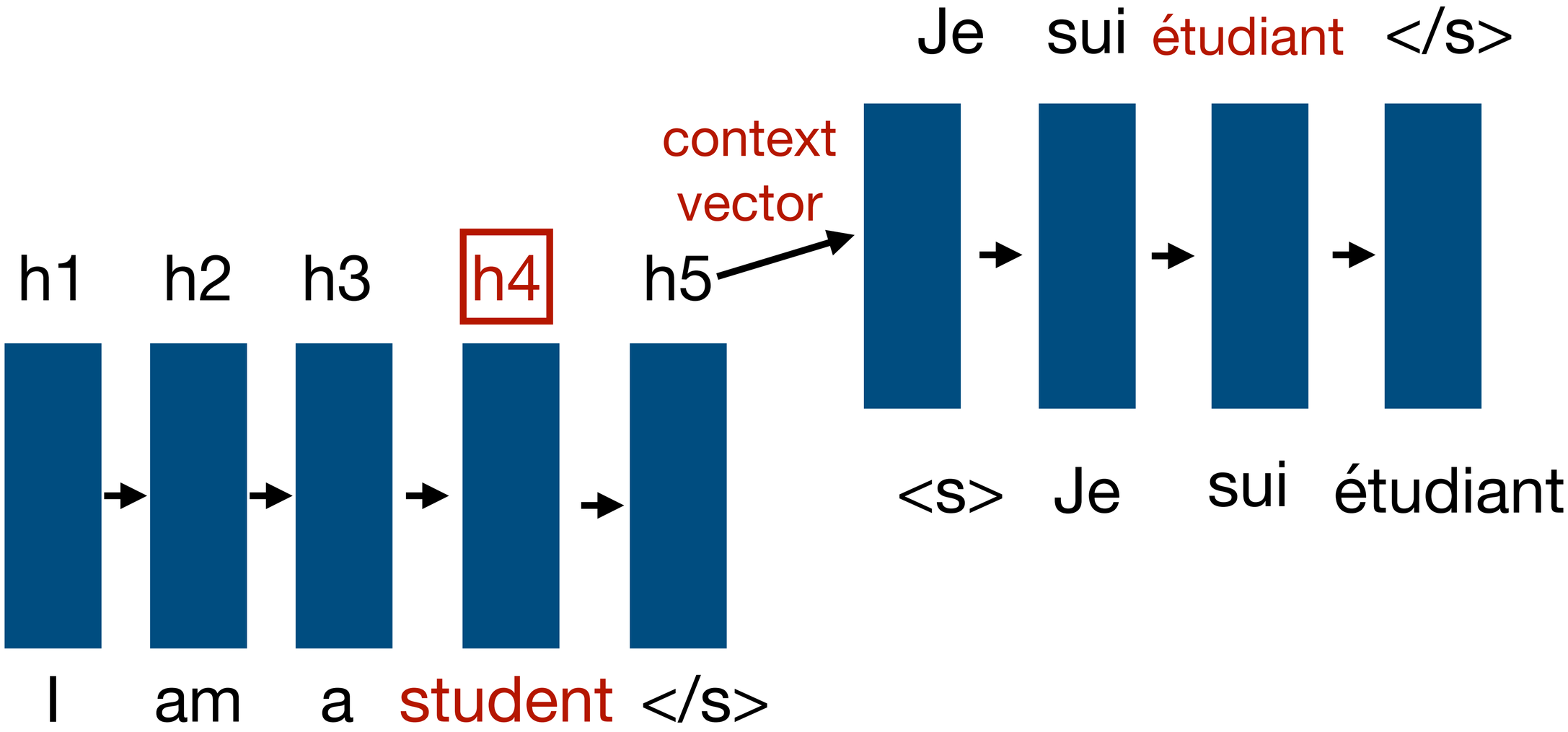
Je suis étudiant en traitement automatique du langage naturel à l'UCL, où je travaille également à temps partiel comme assistant d'enseignement, aidant d'autres étudiants dans leurs travaux …

hidden_dim = 128

We need perfect context vectors

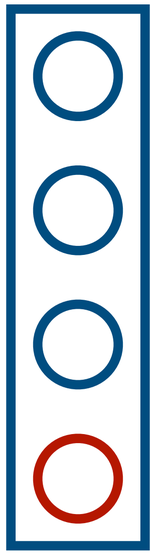


Let the decoder 'look back'



Let the decoder 'look back'

I am a student studying natural language processing at UCL where I also work part-time as a teaching assistant helping other students with their neural language modelling coursework.



Je suis étudiant en traitement automatique du langage naturel à l'UCL, où je travaille également à temps partiel comme assistant d'enseignement, aidant d'autres étudiants dans leurs travaux …

Human view

- We don't memorize the entire source sentence, then translate
- We look back at different parts as we write each word

Human view

- Source: "I am a student studying natural language processing at UCL"

- Target generation: Je suis un étudiant qui étudie le

Human view

- Source: "I am a student studying natural language processing at UCL"

- Target generation: Je suis un étudiant qui étudie le traitement

Human view

- Source: "I am a student studying natural language processing at UCL"

- Target generation: Je suis un étudiant qui étudie le traitement du

Human view

- Source: "I am a student studying natural language processing at UCL"

- Target generation: Je suis un étudiant qui étudie le traitement du langage

Machine View

- (le, natural language processing) -> (s7: h6, h7, h8)
- (traitement, processing) -> (s8: h8)
- (du, language) -> (s9: h7)
- (langage, language) -> (s10: h7)

Solution

- At each decoder step s_i, we can attend to relevant encoder states.
- But we need to define “relevant”

Task: find most relevant tokens

Input:

- Current decoder state s_i
- All encoder states $h_1, h_2, \ldots, h_n$

Output:

- which h_j is most relevant?

Designing the Scoring Function

How relevant is encoder state h_j to decoder state s_i?

$$\text{score} = f(s_i, h_j)$$

Designing the Scoring Function

Dot product

$$f(s_i, h_j) = s_i^T \cdot h_j$$

- No additional parameters to learn
- Issue: s_i and h_j might be in different "semantic spaces"

Designing the Scoring Function

Bilinear (learned transformation)

$$f(s_i, h_j) = s_i^T \cdot W \cdot h_j$$

- W is a learned weight matrix.
- Maps encoder and decoder states to common space

Designing the Scoring Function

Neural network

$$f(s_i, h_j) = v^T \cdot \tanh(W_1 \cdot s_i + W_2 \cdot h_j + b)$$

- W_1: $d_{\text{hidden}} \times d_{\text{decoder}}$ matrix
- W_2: $d_{\text{hidden}} \times d_{\text{encoder}}$ matrix
- v: d_{hidden} dimensional vector
- Output: scalar score

Similarity between hidden states

$$e_{8,1} = s_8^T \cdot h_1 = 0.5 \quad \text{(I)}$$
$$e_{8,2} = s_8^T \cdot h_2 = 0.3 \quad \text{(am)}$$
$$e_{8,3} = s_8^T \cdot h_3 = 0.2 \quad \text{(a)}$$
$$e_{8,4} = s_8^T \cdot h_4 = 0.4 \quad \text{(student)}$$
$$e_{8,5} = s_8^T \cdot h_5 = 1.2 \quad \text{(studying)}$$
$$e_{8,6} = s_8^T \cdot h_6 = 1.8 \quad \text{(natural)}$$
$$e_{8,7} = s_8^T \cdot h_7 = 2.1 \quad \text{(language)}$$
$$e_{8,8} = s_8^T \cdot h_8 = 3.5 \quad \text{(processing)}$$
$$e_{8,9} = s_8^T \cdot h_9 = 0.6 \quad \text{(at)}$$
$$e_{8,10} = s_8^T \cdot h_{10} = 0.4 \quad \text{(UCL)}$$

The step to predict traitement

I am a student studying natural language processing at UCL

Similarity between hidden states

Apply softmax

$$\alpha_{i,j} = \frac{\exp(e_{i,j})}{\sum_{k=1}^{N} \exp(e_{i,k})}$$

$\alpha_{8,1} = 0.020$ (I)

$\alpha_{8,2} = 0.016$ (am)

$\alpha_{8,3} = 0.015$ (a)

$\alpha_{8,4} = 0.018$ (student)

$\alpha_{8,5} = 0.040$ (studying)

$\alpha_{8,6} = 0.073$ (natural)

$\alpha_{8,7} = 0.098$ (language)

$\alpha_{8,8} = 0.396$ (processing)

$\alpha_{8,9} = 0.022$ (at)

$\alpha_{8,10} = 0.018$ (UCL)

Attention overview

Given: decoder state s_i and encoder states $h_1, h_2, \ldots, h_n$

Step 1: Compute alignment scores

$$e_{i,j} = s_i^T \cdot h_j \quad \text{for } j = 1,2,\ldots,n$$

Attention overview

Given: decoder state s_i and encoder states $h_1, h_2, \ldots, h_n$

Step 2: Normalize to attention weights

$$\alpha_{i,j} = \frac{\exp(e_{i,j})}{\sum_{k=1}^{n} \exp(e_{i,k})} \quad \text{for } j = 1,2,\ldots,n$$

Attention overview

Given: decoder state s_i and encoder states $h_1, h_2, \ldots, h_n$

Step 3: Compute context vector

$$c_i = \sum_{j=1}^{n} \alpha_{i,j} \cdot h_j$$

Attention overview

Given: decoder state s_i and encoder states $h_1, h_2, \ldots, h_n$

Step 4: Generate output

$$\tilde{s}_i = \tanh(W_c[c_i; s_i])$$

$$y_i = \text{softmax}(W_y \tilde{s}_i)$$

The philosophy of attention

Supervised view

- (le, natural language processing) -> (s7: h6, h7, h8)
- (traitement, processing) -> (s8: h8)
- (du, language) -> (s9: h7)
- (langage, language) -> (s10: h7)

Scaling is hard

The philosophy of attention

What we have

Parallel sentences: (source, target) pairs

Example: ("I am a student", "Je suis étudiant”)

No labeled attention weight

Learn attention jointly with tasks

Bilinear or neural network-based attention

- The attention weights $\alpha_{i,j}$ are intermediate, latent variables
- Not directly supervised
- Learned as part of the overall translation objective

Trainable parameters θ :

encoder/decoder parameters + attention parameters

Learn attention jointly with tasks

Dot product attention

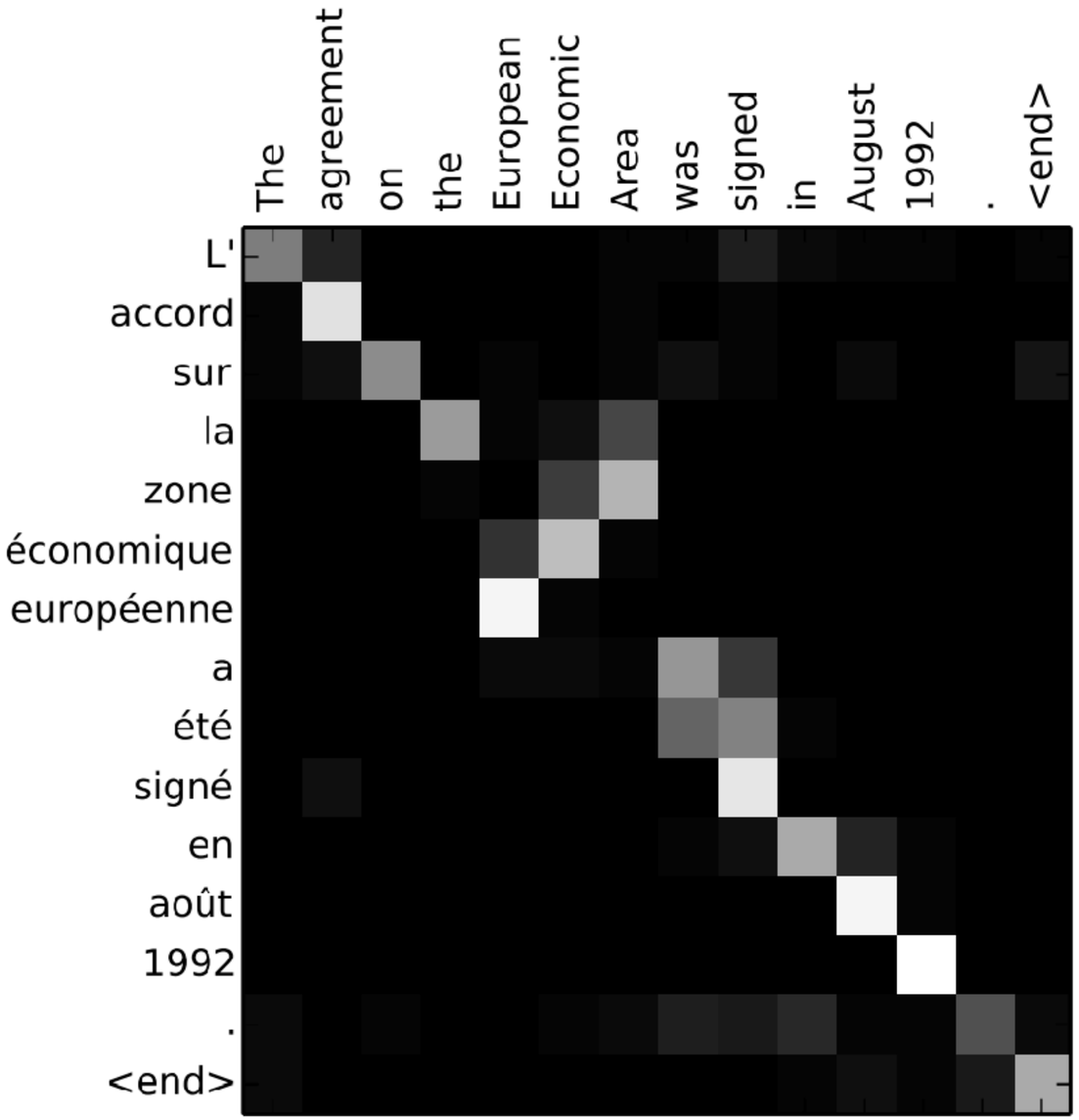
- Parameter-free attention
- Model learns implicitly through the encoder/decoder representations and the output generation parameters

Attention visualisation

Attention matrix: $\alpha_{i,j}$

- Rows: target (French) words
- Columns: source (English) words
- Color intensity: attention weight (darker = higher attention)

Attention visualisation



Long sequence modelling with attention

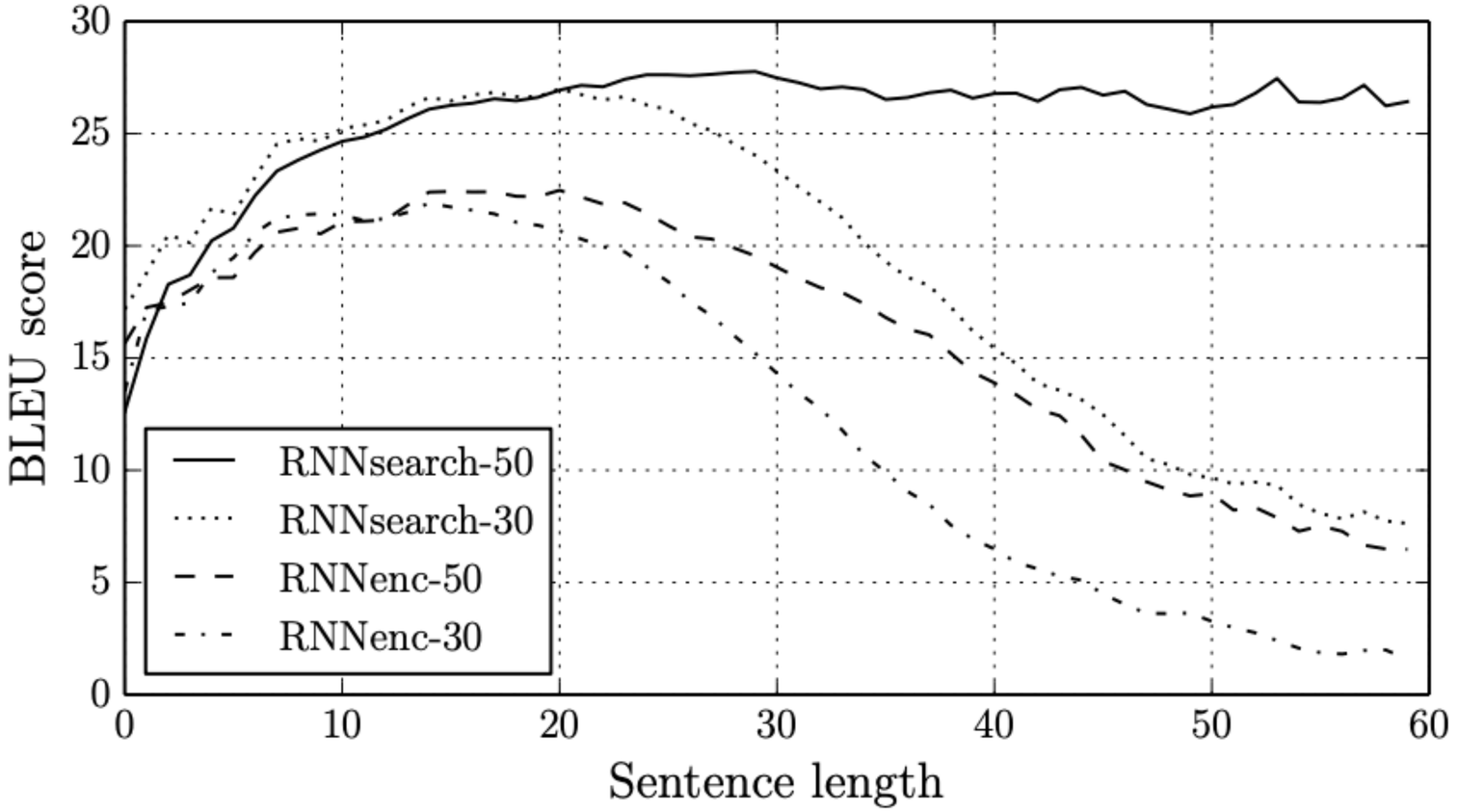


Figure 2: The BLEU scores of the generated translations on the test set with respect to the lengths of the sentences. The results are on the full test set which includes sentences having unknown words to the models.

Attention as a Shortcut

Vanilla seq2seq:

- Encoder: $x_j \rightarrow h_j \rightarrow ... \rightarrow h_n$ (many RNN steps)
- Context: h_n compressed into single vector

Seq2seq with attention

Information from any encoder position h_j can flow directly to any decoder step i

Text generation

Given a context, generate the next token

$$P(x_t \mid x_1, x_2, \ldots, x_{t-1})$$

How do we choose which word to generate?

Greedy Decoding

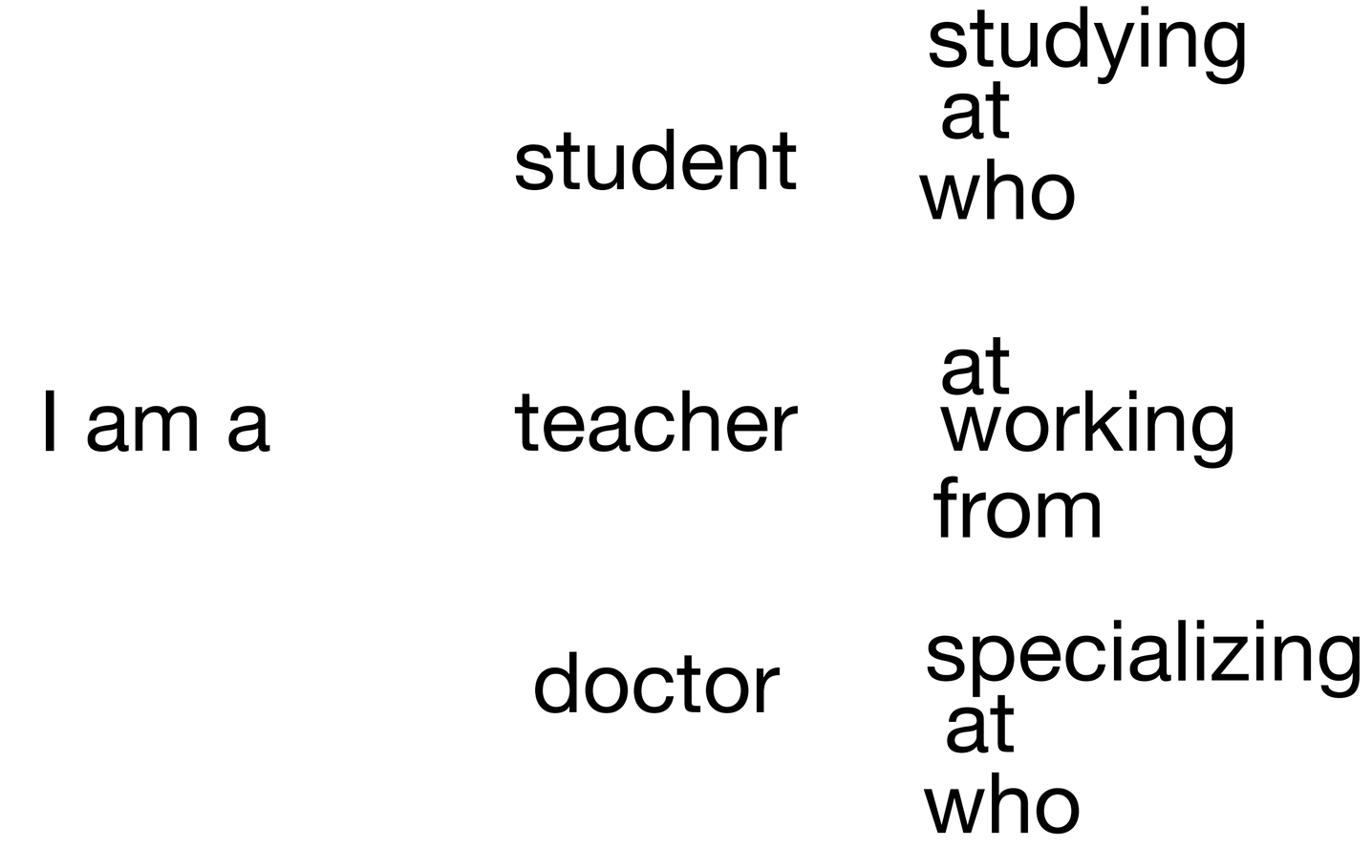
Always pick the most probable token:

$$x_t = \arg\max P(x_t \mid x_{<t})$$

- Pros: Simple, fast, deterministic
- Cons: Repetitive generation, lack of diversity

Beam Search

Keep track of top-k most probable sequences:



Beam Search

Keep track of top-k most probable sequences:

I am a student studying

I am a student at

I am a teacher at

Random sampling

Sample from the probability distribution:

$$x_t \sim P(x_t \mid x_{<t})$$

Random sampling with temperature scaling

Sample from the probability distribution:

$$P_{\text{temp}}(x_t) = \frac{\exp(\text{logit}_t / T)}{\sum_i \exp(\text{logit}_i / T)}$$

Random sampling with temperature scaling

I am a

student: z = 2.0

teacher: z = 1.5

doctor: z = 1.0

	T=1	T=0.01	T=10
student	50.6	1.00	33.5
teacher	30.7	1E-22	33.3
doctor	18.6	1E-44	33.1

Top-k sampling

P(student) = 0.35 P(teacher) = 0.20 P(doctor) = 0.15
P(person) = 0.10 P(elephant) = 0.001

Step 1: Select top-K tokens

Step2: Renormalize to sum to 1

Step 3: Sample from these 3 options only

Top-p sampling

P(student) = 0.35 P(teacher) = 0.20 P(doctor) = 0.15
P(person) = 0.10 P(elephant) = 0.001

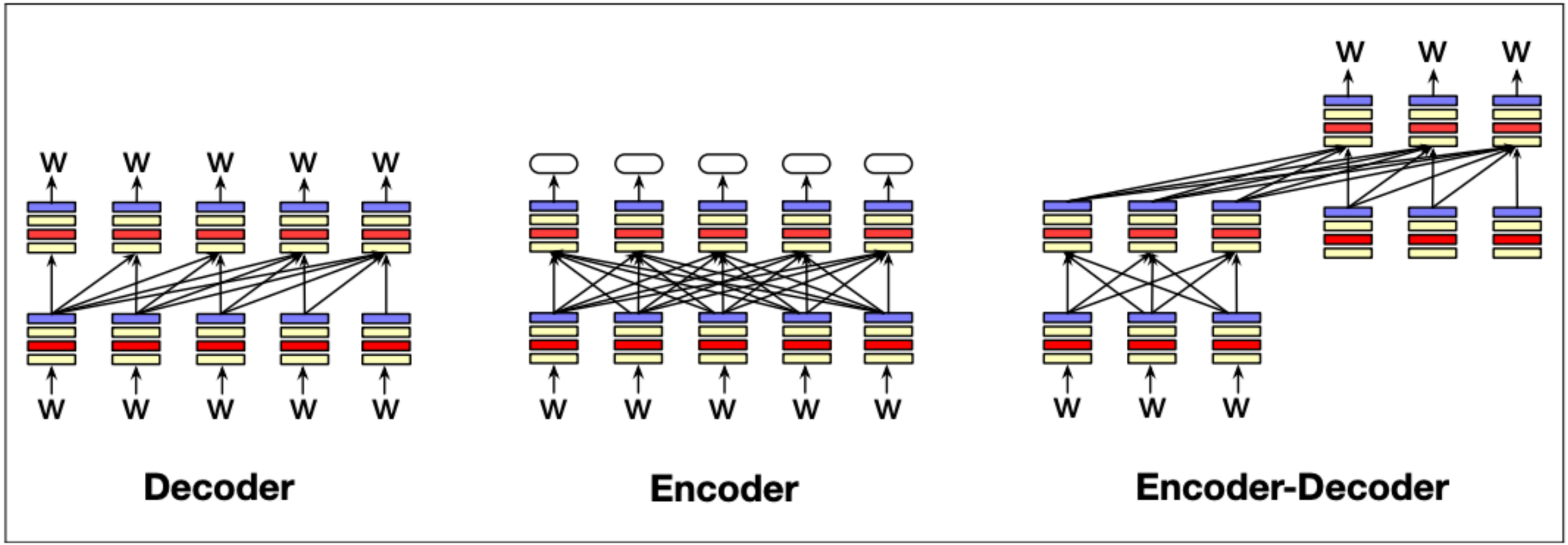
Choose smallest set of tokens whose cumulative probability ≥ p

Peaked distribution → fewer tokens (more focused)

Flat distribution → more tokens (more diverse)

Let's move on to a new topic.

Three model architectures



Source: https://web.stanford.edu/~jurafsky/slp3/

Decoder-only models

Example: RNN Language Model

- Modern example: GPT-2

RNN-LM and GPT-2

- Decoder-only architectures
- Autoregressive: generate one word at a time, left-to-right
- Language models: learn $P(w_t \mid w_1, \ldots, w_{t-1})$
- Trained on the same objective: predict next word

RNN-LM and GPT-2

- RNN-LM: uses recurrent connections
- GPT-2: uses a self-attention architecture

GPT is just a scaled-up, more efficient version of RNN language models.
The fundamental idea hasn't changed.

GPT-2: Language Modeling at Scale

- Architecture: Decoder-only (like RNN-LM, but scaled up)
- 1.5 billion parameters
- Can handle much longer context (1024 tokens)
- Training objective: langauge modelling

GPT-2 release

The Guardian

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

The creators of a revolutionary AI system that can write news stories and works of fiction – dubbed “deepfakes for text” – have taken the unusual step of not...

14 Feb 2019

Open **Release The Full Model!** #16

Franck-Dernoncourt on Feb 15, 2019

https://blog.openai.com/better-language-models/:

> We will further publicly discuss this [model release] strategy in six months. If you’d like to discuss large language models and their implications, please email us at: languagequestions@openai.com.

1

GPT-2 training data

WebText dataset (never released)

- 40GB of text crawled from internet
- 8 million documents from Reddit outbound links

How Does LM Lead to Task Learning?

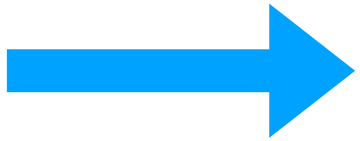
- Key idea: Natural text contains tasks implicitly

How Does LM Lead to Task Learning?

FineWeb Pretrain Data: Translation Pairs

Old English : bacan = ”baker”
Proto-Germanic : bakanan = ”bake”
Old Norse : baka = ”bake”
Middle Dutch : backen = ”bake”
Old High German : bahhan = ”bake”
German : backen = ”bake”
PIE : bheg- ”to warm, roast, bake”
Greek : phogein ”to roast”
Root : bhe- ”to warm”

Implicit task learning

I am a student  Je sui étudiant

Modern solution:

<s> English: I am a student. French: Je sui étudiant </s>

It works for any NLP tasks.

Encoder-only model

Different Goal: Not generation, but understanding

Key Tasks:

- Classify entire sentence (sentiment analysis, topic classification)
- Extract information (named entity recognition)
- Compare sentences (semantic similarity, paraphrase detection)

Limitation of decoder-only models

Unidirectional context

Key Tasks:

- GPT only sees left context: "I am a ___"
- For understanding, we want to see the full sentence in both directions

From Static to Contextualized Embeddings

Traditional Word Embeddings (Word2Vec, GloVe)

- Problem: One vector per word, regardless of context

ELMo (2018): Contextualized Word Representations

- Take embeddings from Language Models
- Key idea: Word representation depends on context

ELMo: Use bidirectional context

I am studying NLP at University College London

$h_{\text{right-to-left}}$

$$h_t = [h_{\text{right-to-left}}; h_{\text{right-to-left}}]$$

ELMo: Use bidirectional context

I went to the bank to deposit money

I sat by the river bank

- "bank" near "deposit" → financial meaning
- "bank" near "river" → geographical meaning
- Same word, different contexts → different representations

Better representation with language modelling

Task	Previous SOTA		Our baseline	ELMo + baseline	Increase (absolute/ relative)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17	0.7 / 5.8%
SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10	2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5	3.3 / 6.8%

Table 1: Test set comparison of ELMo enhanced neural models with state-of-the-art single model baselines across six benchmark NLP tasks. The performance metric varies across tasks – accuracy for SNLI and SST-5; F_1 for SQuAD, SRL and NER; average F_1 for Coref. Due to the small test sizes for NER and SST-5, we report the mean and standard deviation across five runs with different random seeds. The "increase" column lists both the absolute and relative improvements over our baseline.

ELMo: trained on 1B token for 10 epochs

Google: I have an idea.

BERT: (Masked) Language Modeling at Scale

- Dataset size: Trained on 3.3B tokens (BooksCorpus + Wikipedia)
- Replace recurrence architecture with transformer
- Much larger model capacity
- Different training objectives

Masked Language Modelling

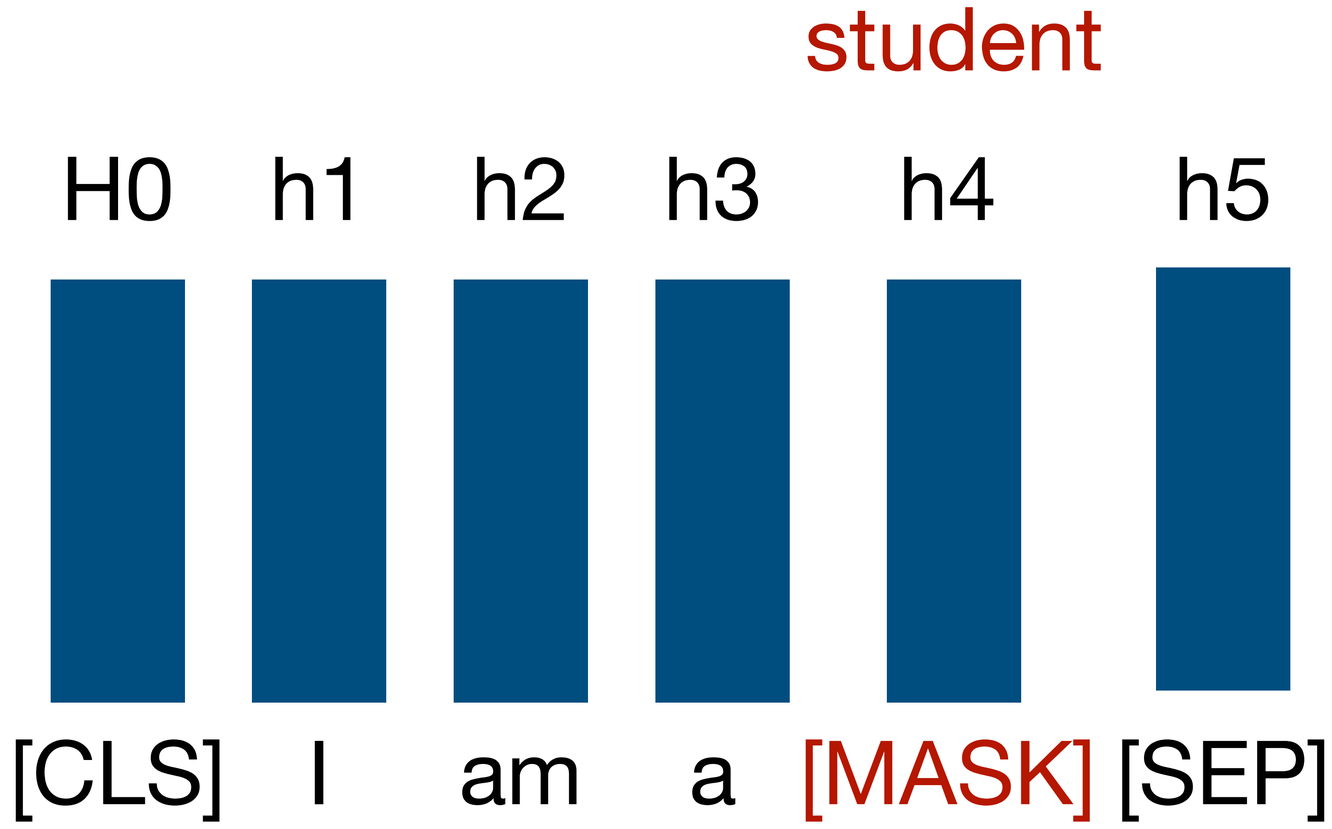
- Mask 15% of tokens, predict them using full bidirectional context

I am a student studying NLP at UCL

I am a [MASK] studying NLP at [MASK]

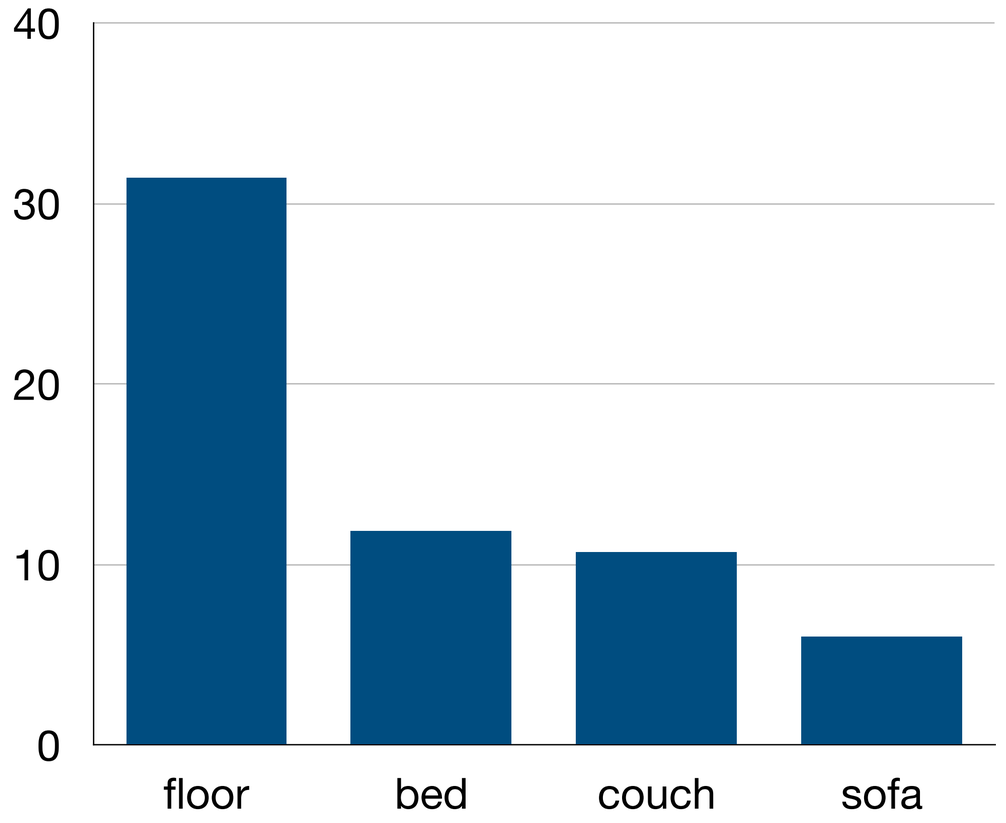
$P(\text{student} \mid \text{I am a [MASK] studying NLP at [MASK]})$
$P(\text{UCL} \mid \text{I am a [MASK] studying NLP at [MASK]})$

BERT training objectives



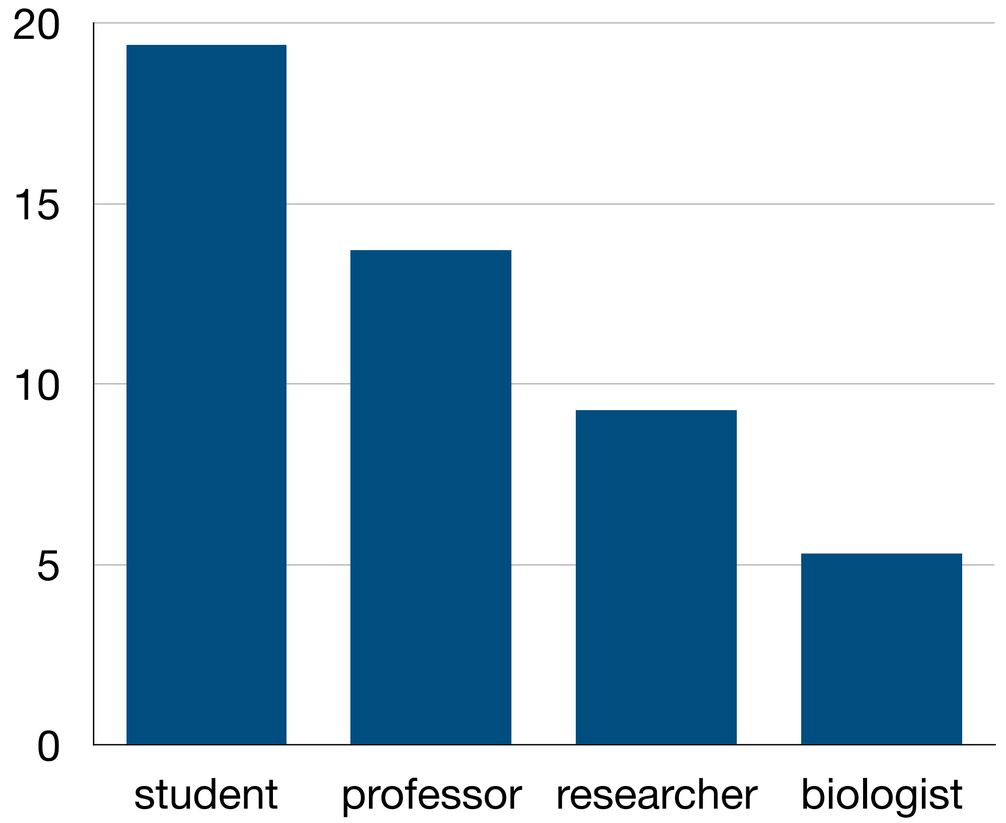
BERT Learns with Cloze-Style Prediction

The cat sat on the ___



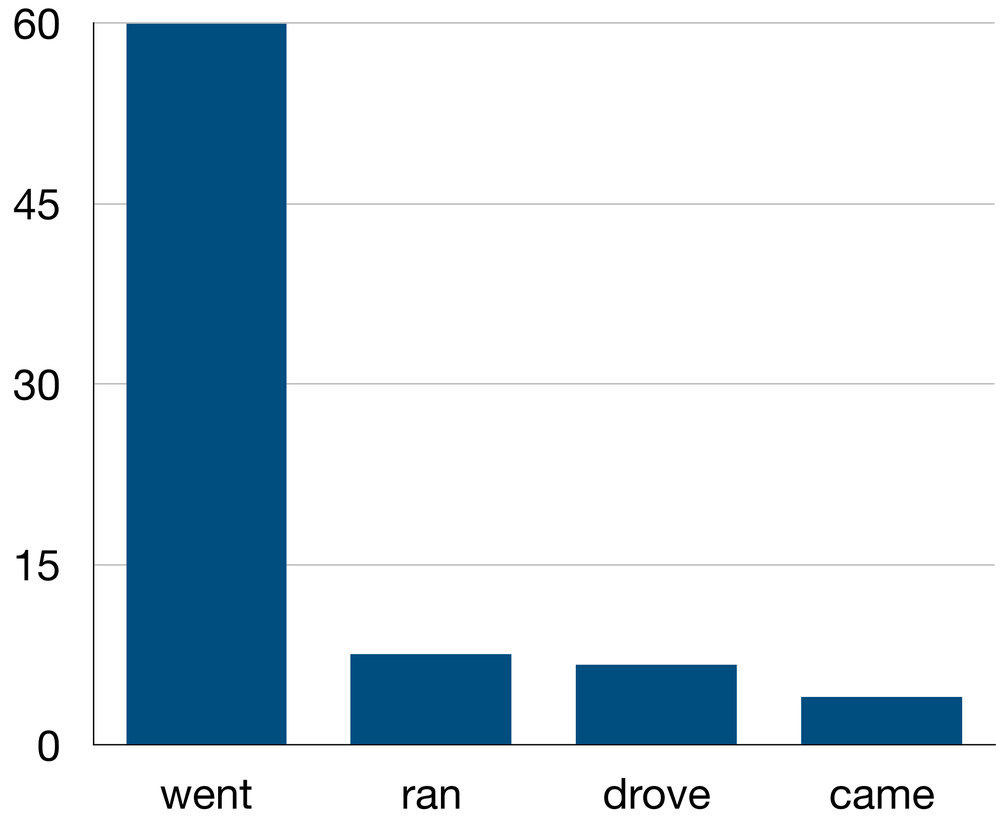
BERT Learns with Cloze-Style Prediction

I am a ___ studying NLP



BERT Learns with Cloze-Style Prediction

She ___ to the store yesterday



Language Models as Knowledge Bases?

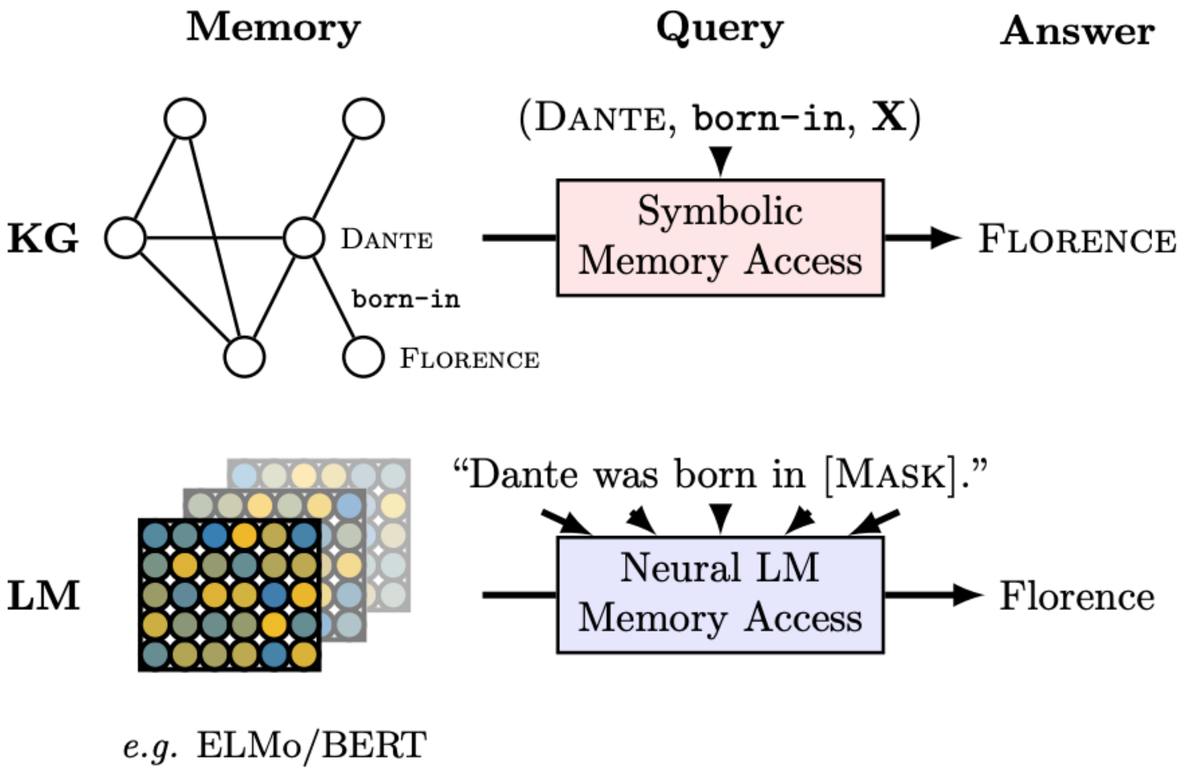


Figure 1: Querying knowledge bases (KB) and language models (LM) for factual knowledge.

How BERT is Actually Used

Add task-specific layers on top of BERT

Update BERT's weights for your specific task

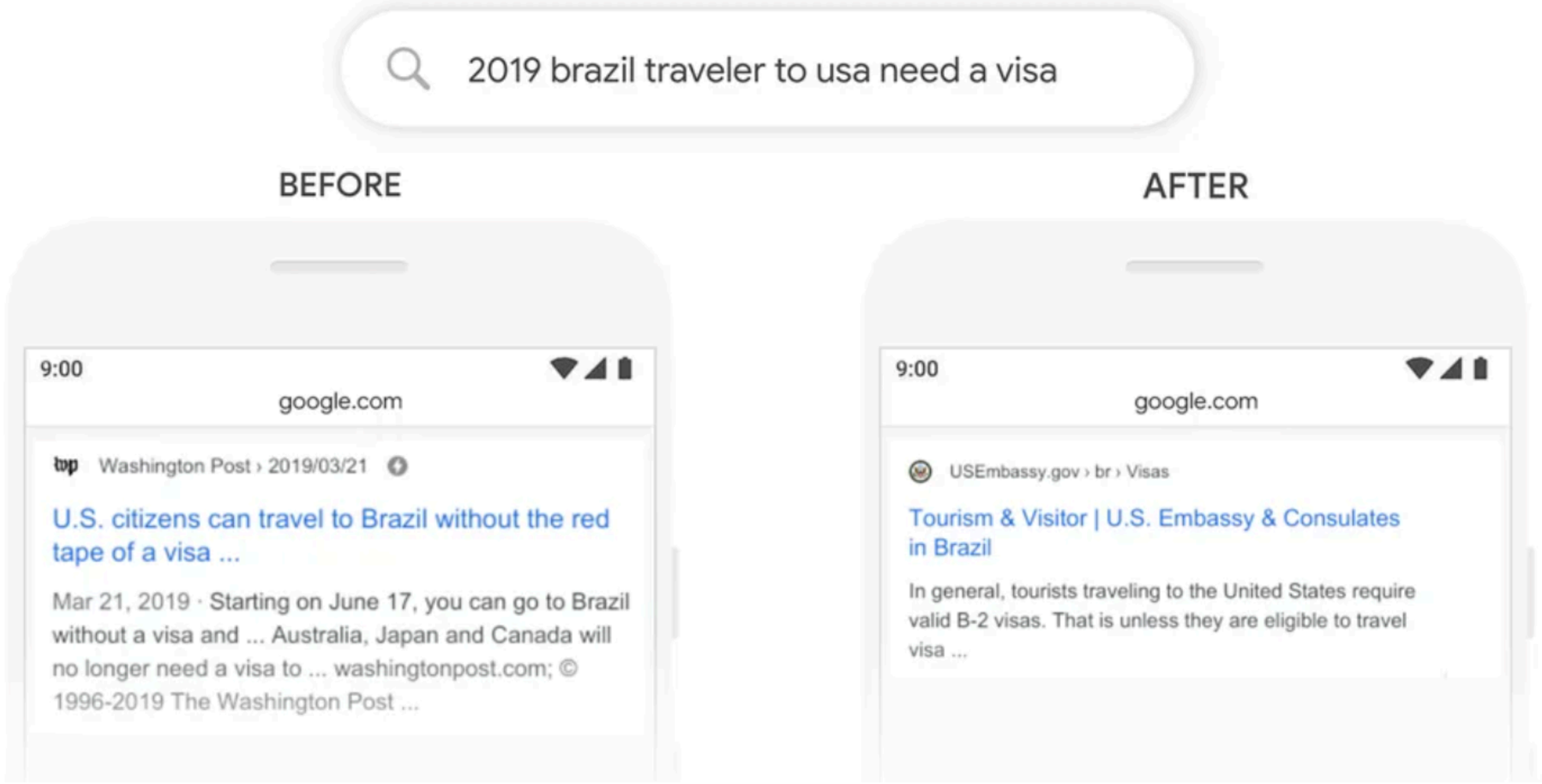
- Sentiment analysis: BERT → classifier layer
- NER: BERT → token classification layer
- Question answering: BERT → span prediction layer

How BERT is Actually Used

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
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**

Table 1: GLUE Test results, scored by the evaluation server (https://gluebenchmark.com/leaderboard). The number below each task denotes the number of training examples. The "Average" column is slightly different than the official GLUE score, since we exclude the problematic WNLI set.[8] BERT and OpenAI GPT are single-model, single task. F1 scores are reported for QQP and MRPC, Spearman correlations are reported for STS-B, and accuracy scores are reported for the other tasks. We exclude entries that use BERT as one of their components.

Real-world impact: Google Search (2019)



Encoder-Decoder modelling

Example: RNN Seq2seq Model

- Task: Transform input sequence → output sequence
- Example: "I am a student" → "Je suis un étudiant"
- Architecture: Encoder compresses input, Decoder generates output
- Modern example: T5, BART

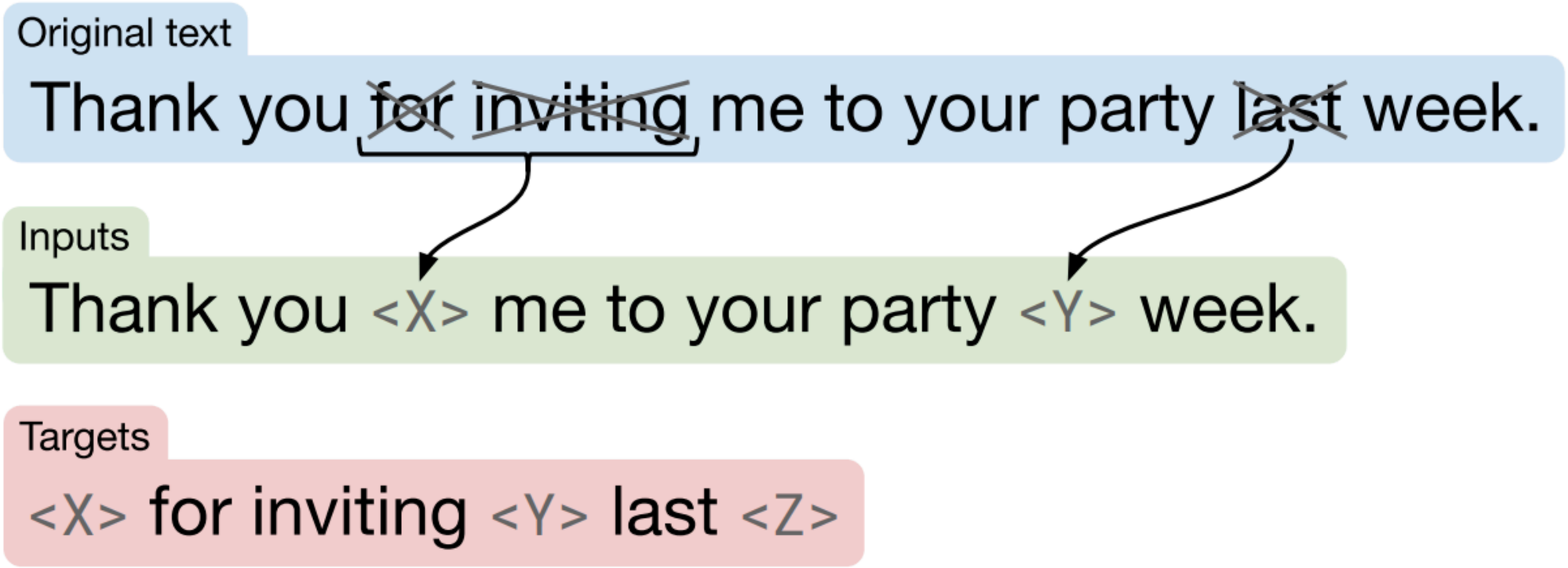
T5: Text-to-Text Transfer Transformer

- Encoder: Let’s replace RNN with BERT
- Decoder: Let’s replace RNN with GPT

T5: Text-to-Text Transfer Transformer

- Encoder: Bidirectional, reads full input (like BERT)
- Decoder: Autoregressive, generates output (like GPT)
- Frame ALL NLP tasks as text-to-text

T5: Text-to-Text Transfer Transformer



T5: Text-to-Text Transfer Transformer

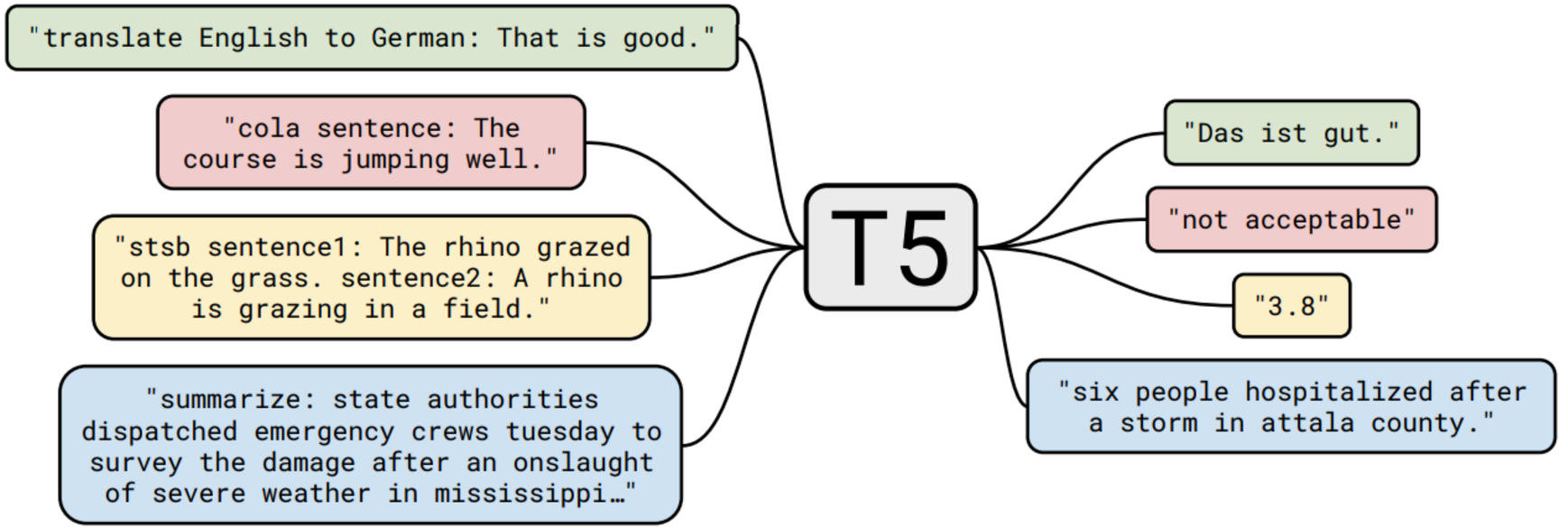


Figure 1: A diagram of our text-to-text framework. Every task we consider—including translation, question answering, and classification—is cast as feeding our model text as input and training it to generate some target text. This allows us to use the same model, loss function, hyperparameters, etc. across our diverse set of tasks. It also provides a standard testbed for the methods included in our empirical survey. "T5" refers to our model, which we dub the "**T**ext-**t**o-**T**ext **T**ransfer **T**ransformer".

T5 pretraining data

- C4 dataset: widely used by later pretraining efforts
- 750GB open data release

The history of pretraining

- Pretraining: Elmo (encoder-only)
- Scaling: BERT (encoder-only)
- Different training objectives: T5 (encoder-decoder), BERT, GPT2
- Training data: Elmo(1B), BERT (30B), GPT2 (10B), T5 (156B)
- Inference usage: finetuning, prompting (BERT)

Thank you!