(Even More) Language Modeling: Attention, and Building Blocks of Transformers

CMSC 473/673
Frank Ferraro

Five Broad Categories of Neural Networks

("single input" ~= single, flat vector, e.g., BoW of a document)
(single output ~= one prediction only)

Single Input, Single Output

Single Input, Multiple Outputs

Multiple Inputs, Single Output

Multiple Inputs, Multiple Outputs ("sequence prediction": no time delay)

Multiple Inputs, Multiple Outputs ("sequence-to-sequence": with time delay)

	Heads			
Name	Input	Output	Tasks	Ex. Datasets
Language Modeling	$x_{1:n-1}$	$x_n \in \mathcal{V}$	Generation	WikiText-103
Sequence Classification	$x_{1:N}$	$y \in \mathcal{C}$	Classification, Sentiment Analysis	GLUE, SST, MNLI
Question Answering	$x_{1:M}, x_{M:N}$	y span $[1:N]$	QA, Reading Comprehension	SQuAD, Natural Questions
Token Classification	$x_{1:N}$	$y_{1:N} \in \mathcal{C}^N$	NER, Tagging	OntoNotes, WNUT
Multiple Choice	$x_{1:N}, \mathcal{X}$	$y \in \mathcal{X}$	Text Selection	SWAG, ARC
Masked LM	$x_{1:N \setminus n}$	$x_n \in \mathcal{V}$	Pretraining	Wikitext, C4
Conditional Generation	$x_{1:N}$	$y_{1:M} \in \mathcal{V}^M$	Translation, Summarization	WMT, IWSLT, CNN/DM, XSum

Fig. 2 (Wolf et al., 2020: https://arxiv.org/pdf/1910.03771.pdf)

Five Broad Categories of Neural Networks

("single input" ~= single, flat vector, e.g., BoW of a document)
(single output ~= one prediction only)

Single Input, Single Output

Single Input, Multiple Outputs

Multiple Inputs, Single Output

Multiple Inputs, Multiple Outputs ("sequence prediction": no time delay)

Multiple Inputs, Multiple Outputs ("sequence-to-sequence": with time delay)

Name	Heads: Input	Heads: Output	Tasks	Ex. Datasets
Language Modeling	$x_{1:n-1}$	$x_n \in \mathcal{V}$	Generation	WikiText-103
Sequence Classification	$x_{1:N}$	$y \in \mathcal{C}$	Classification, Sentiment Analysis	GLUE, SST, MNLI
Question Answering	$x_{1:M}, x_{M:N}$	y span $[1:N]$	QA, Reading Comprehension	SQuAD, Natural Questions
Token Classification	$x_{1:N}$	$y_{1:N} \in \mathcal{C}^N$	NER, Tagging	OntoNotes, WNUT
Multiple Choice	$x_{1:N}, \mathcal{X}$	$y \in \mathcal{X}$	Text Selection	SWAG, ARC
Masked LM	$x_{1:N \setminus n}$	$x_n \in \mathcal{V}$	Pretraining	Wikitext, C4
Conditional Generation	$x_{1:N}$	$y_{1:M} \in \mathcal{V}^M$	Translation, Summarization	WMT, IWSLT, CNN/DM, XSum

Fig. 2 (Wolf et al., 2020: https://arxiv.org/pdf/1910.03771.pdf)

Five Broad Categories of Neural Networks

("single input" ~= single, flat vector, e.g., BoW of a document)
(single output ~= one prediction only)

Single Input, Single Output

Single Input, Multiple Outputs

Multiple Inputs, Single Output

Multiple Inputs, Multiple Outputs ("sequence prediction": no time delay)

Multiple Inputs, Multiple Outputs ("sequence-to-sequence": with time delay)

Name	Heads Input	Heads Output	Tasks	Ex. Datasets
Language Modeling	$x_{1:n-1}$	$x_n \in \mathcal{V}$	Generation	WikiText-103
Sequence Classification	$x_{1:N}$	$y \in \mathcal{C}$	Classification, Sentiment Analysis	GLUE, SST, MNLI
Question Answering	$x_{1:M}, x_{M:N}$	y span $[1:N]$	QA, Reading Comprehension	SQuAD, Natural Questions
Token Classification	$x_{1:N}$	$y_{1:N} \in \mathcal{C}^N$	NER, Tagging	OntoNotes, WNUT
Multiple Choice	$x_{1:N}, \mathcal{X}$	$y \in \mathcal{X}$	Text Selection	SWAG, ARC
Masked LM	$x_{1:N \setminus n}$	$x_n \in \mathcal{V}$	Pretraining	Wikitext, C4
Conditional Generation	$x_{1:N}$	$y_{1:M} \in \mathcal{V}^M$	Translation, Summarization	WMT, IWSLT, CNN/DM, XSum

Fig. 2 (Wolf et al., 2020: https://arxiv.org/pdf/1910.03771.pdf)

Five Broad Categories of Neural Networks

("single input" ~= single, flat vector, e.g., BoW of a document)
(single output ~= one prediction only)

Single Input, Single Output

Single Input, Multiple Outputs

Multiple Inputs, Single Output

Multiple Inputs, Multiple Outputs ("sequence prediction": no time delay)

Multiple Inputs, Multiple Outputs ("sequence-to-sequence": with time delay)

Name	Heads Input	Heads Output	Tasks	Ex. Datasets
Language Modeling	$x_{1:n-1}$	$x_n \in \mathcal{V}$	Generation	WikiText-103
Sequence Classification	$x_{1:N}$	$y \in \mathcal{C}$	Classification, Sentiment Analysis	GLUE, SST, MNLI
Question Answering	$x_{1:M}, x_{M:N}$	y span $[1:N]$	QA, Reading Comprehension	SQuAD, Natural Questions
Token Classification	$x_{1:N}$	$y_{1:N} \in \mathcal{C}^N$	NER, Tagging	OntoNotes, WNUT
Multiple Choice	$x_{1:N}, \mathcal{X}$	$y \in \mathcal{X}$	Text Selection	SWAG, ARC
Masked LM	$x_{1:N \setminus n}$	$x_n \in \mathcal{V}$	Pretraining	Wikitext, C4
Conditional Generation	$x_{1:N}$	$y_{1:M} \in \mathcal{V}^M$	Translation, Summarization	WMT, IWSLT, CNN/DM, XSum

Fig. 2 (Wolf et al., 2020: https://arxiv.org/pdf/1910.03771.pdf)

Five Broad Categories of Neural Networks

("single input" ~= single, flat vector, e.g., BoW of a document)
(single output ~= one prediction only)

Single Input, Single Output

Single Input, Multiple Outputs

Multiple Inputs, Single Output

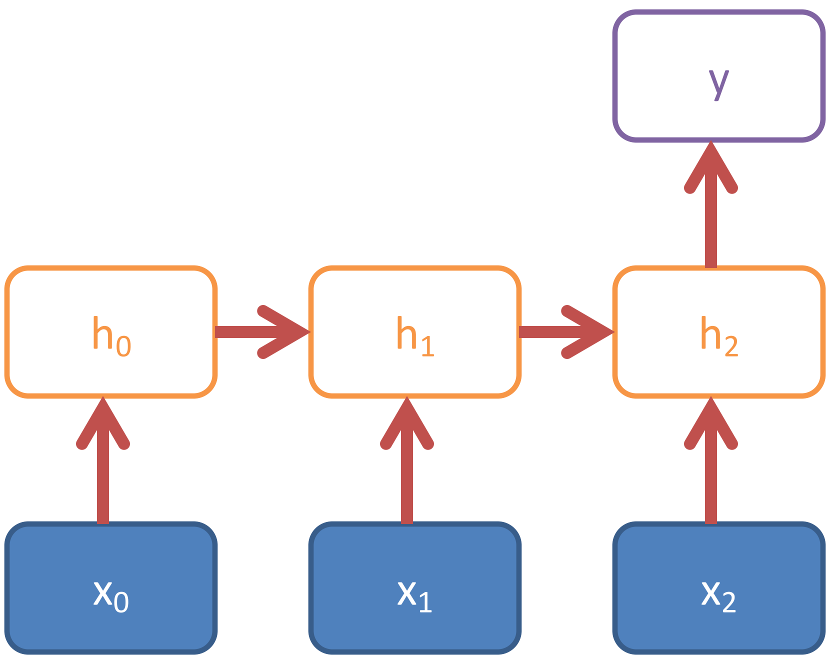
Multiple Inputs, Multiple Outputs ("sequence prediction": no time delay)

Multiple Inputs, Multiple Outputs ("sequence-to-sequence": with time delay)

	Heads			
Name	Input	Output	Tasks	Ex. Datasets
Language Modeling	$x_{1:n-1}$	$x_n \in \mathcal{V}$	Generation	WikiText-103
Sequence Classification	$x_{1:N}$	$y \in \mathcal{C}$	Classification, Sentiment Analysis	GLUE, SST, MNLI
Question Answering	$x_{1:M}, x_{M:N}$	y span $[1:N]$	QA, Reading Comprehension	SQuAD, Natural Questions
Token Classification	$x_{1:N}$	$y_{1:N} \in \mathcal{C}^N$	NER, Tagging	OntoNotes, WNUT
Multiple Choice	$x_{1:N}, \mathcal{X}$	$y \in \mathcal{X}$	Text Selection	SWAG, ARC
Masked LM	$x_{1:N \setminus n}$	$x_n \in \mathcal{V}$	Pretraining	Wikitext, C4
Conditional Generation	$x_{1:N}$	$y_{1:M} \in \mathcal{V}^M$	Translation, Summarization	WMT, IWSLT, CNN/DM, XSum

Fig. 2 (Wolf et al., 2020: https://arxiv.org/pdf/1910.03771.pdf)

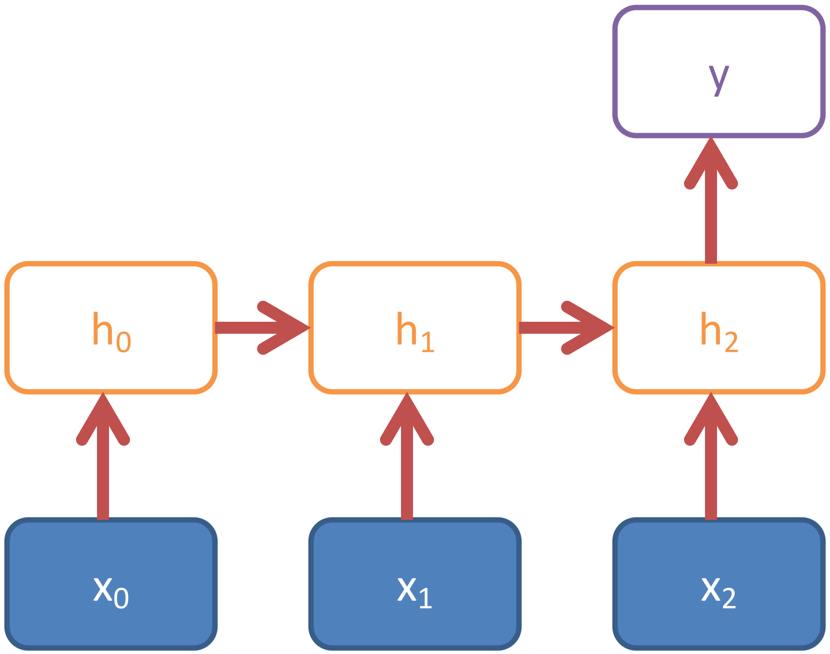
Sequence (Multiple) Input, Single Label Output



Recurrent: Sequence input, one output

Document classification
Action recognition in video (high-level)

Sequence (Multiple) Input, Single Label Output



Recurrent: Sequence input, one output

Document classification
Action recognition in video (high-level)

Think of this as generalizing using maxent models to build discriminatively trained classifiers

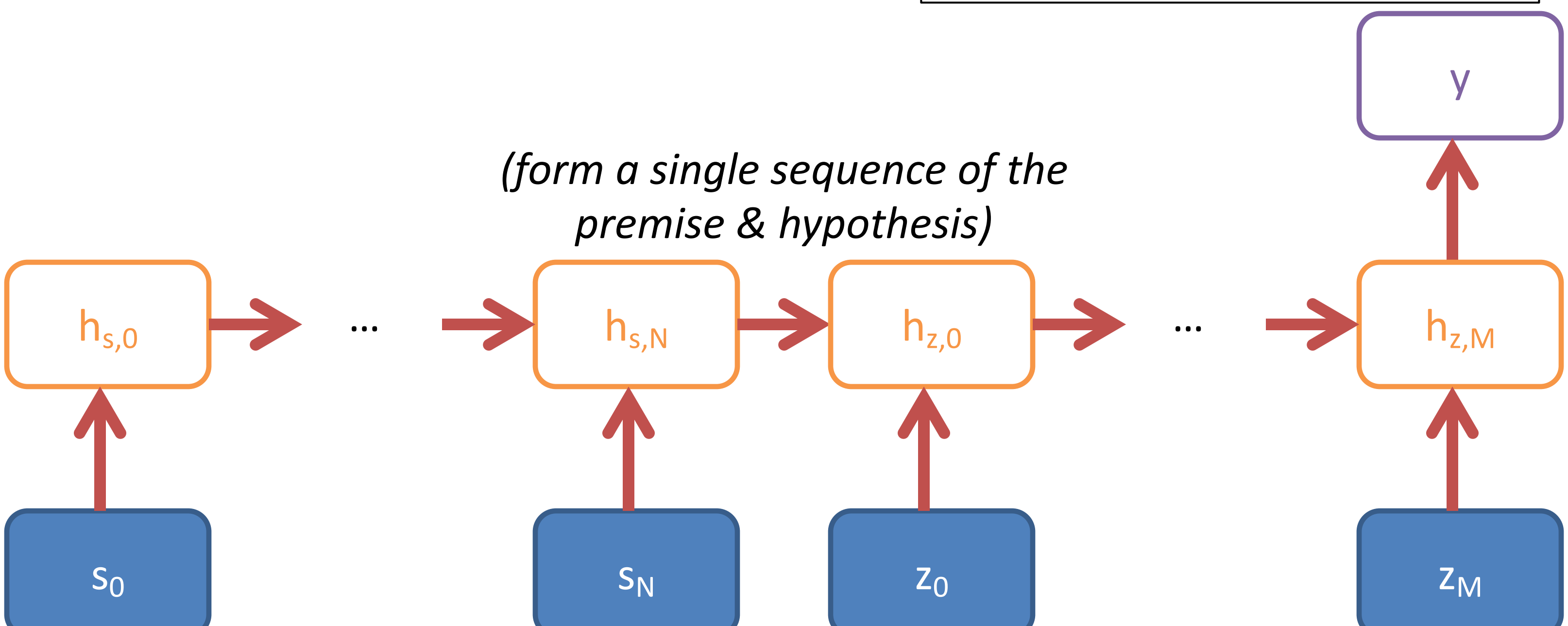
$$p(y \mid x) = \exp(\theta_y^T \text{ flat\_feats}(x))$$

$$\rightarrow$$

$$p(y \mid x) = \exp(\theta_y^T \text{ recurrent\_feats}(x))$$

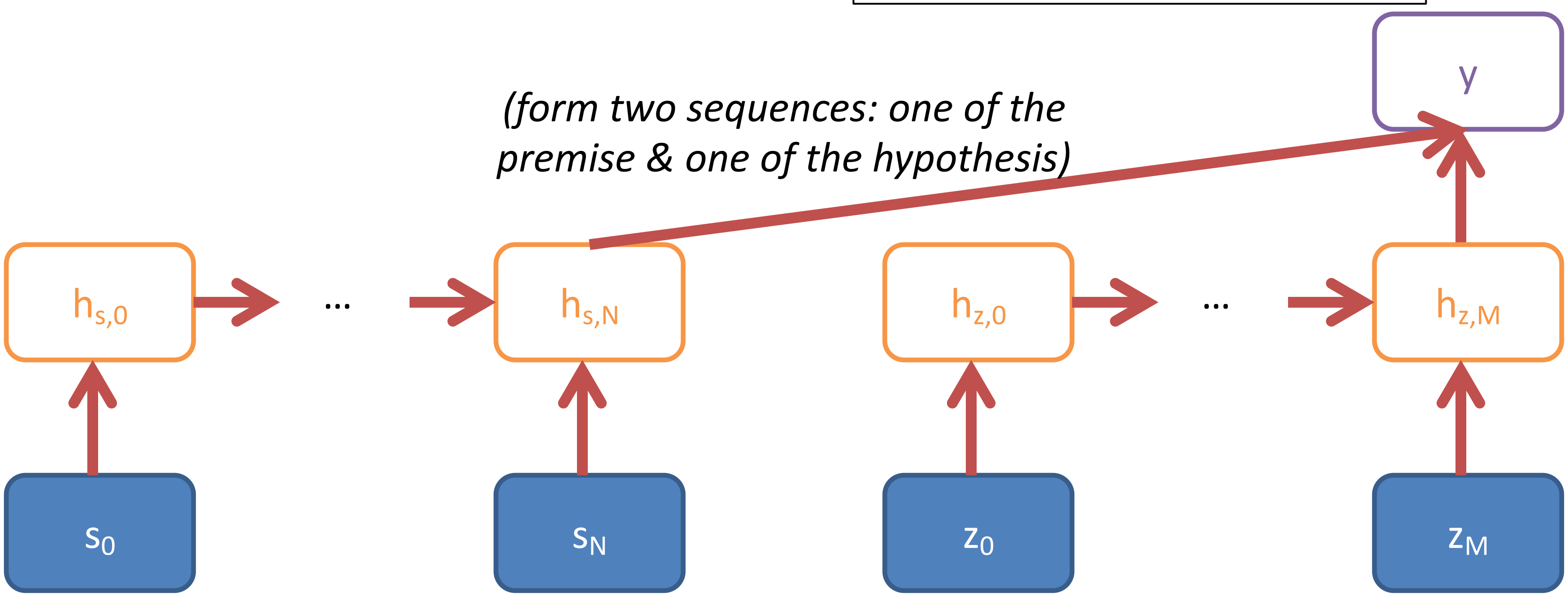
Example: RTE (many options)

p(ENTAILED | s: Michael Jordan, coach Phil Jackson and the star cast, including Scottie Pippen, took the Chicago Bulls to six National Basketball Association championships.
z: The Bulls basketball team is based in Chicago.)



Example: RTE (many options)

p(ENTAILED | s: Michael Jordan, coach Phil Jackson and the star cast, including Scottie Pippen, took the Chicago Bulls to six National Basketball Association championships.
z: The Bulls basketball team is based in Chicago.)



Reminder!

Many (but not all) of these tasks fall into the Sequence Input, Label Output

GLUE
https://gluebenchmark.com/

GLUE Tasks

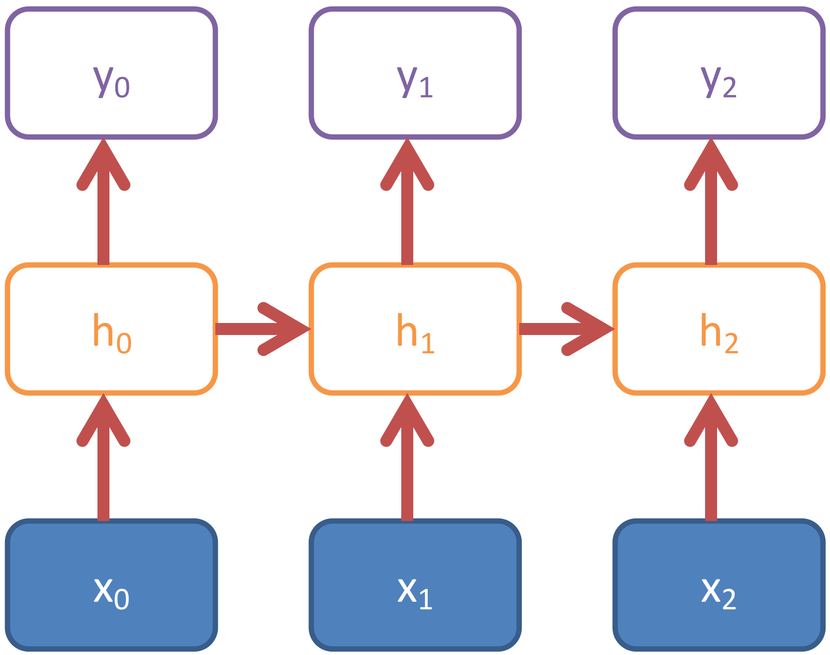
Name	Download
The Corpus of Linguistic Acceptability	
The Stanford Sentiment Treebank	
Microsoft Research Paraphrase Corpus	
Semantic Textual Similarity Benchmark	
Quora Question Pairs	
MultiNLI Matched	
MultiNLI Mismatched	
Question NLI	
Recognizing Textual Entailment	
Winograd NLI	
Diagnostics Main	

Name	Identifier
Broadcoverage Diagnostics	AX-b
CommitmentBank	CB
Choice of Plausible Alternatives	COPA
Multi-Sentence Reading Comprehension	MultiRC
Recognizing Textual Entailment	RTE
Words in Context	WiC
The Winograd Schema Challenge	WSC
BoolQ	BoolQ
Reading Comprehension with Commonsense Reasoning	ReCoRD
Winogender Schema Diagnostics	AX-g

SuperGLUE

https://super.gluebenchmark.com/

Sequence Input, Sequence Output
("sequence prediction": no time delay)



Recursive: Sequence input, Sequence output

Part of speech tagging
Named entity recognition

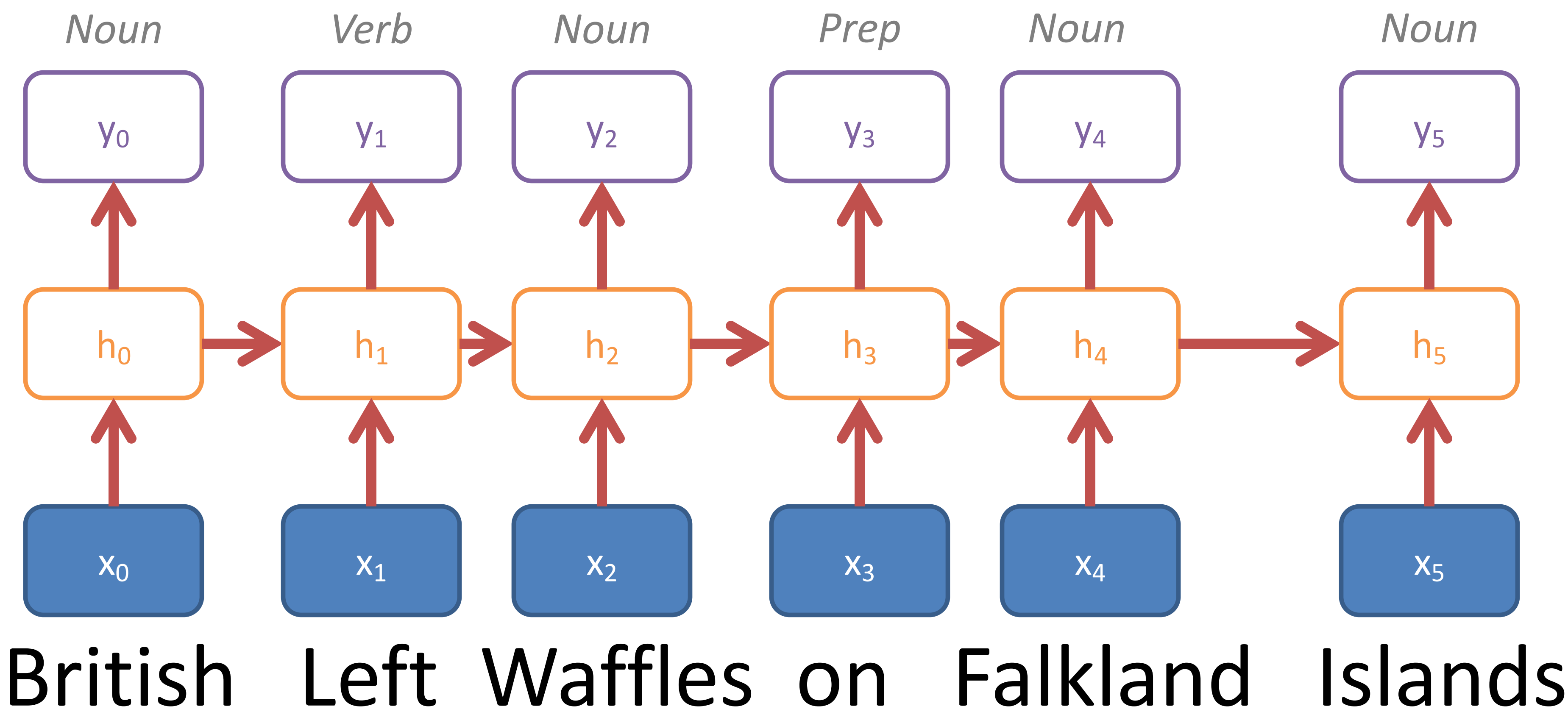
Heads				
Name	Input	Output	Tasks	Ex. Datasets
Language Modeling	$x_{1:n-1}$	$x_n \in \mathcal{V}$	Generation	WikiText-103
Sequence Classification	$x_{1:N}$	$y \in \mathcal{C}$	Classification, Sentiment Analysis	GLUE, SST, MNLI
Question Answering	$x_{1:M}, x_{M:N}$	y span $[1:N]$	QA, Reading Comprehension	SQuAD, Natural Questions
Token Classification	$x_{1:N}$	$y_{1:N} \in \mathcal{C}^N$	NER, Tagging	OntoNotes, WNUT
Multiple Choice	$x_{1:N}, \mathcal{X}$	$y \in \mathcal{X}$	Text Selection	SWAG, ARC
Masked LM	$x_{1:N \setminus n}$	$x_n \in \mathcal{V}$	Pretraining	Wikitext, C4
Conditional Generation	$x_{1:N}$	$y_{1:M} \in \mathcal{V}^M$	Translation, Summarization	WMT, IWSLT, CNN/DM, XSum

Example 1: Part of Speech Tagging

Noun	*Verb*	*Noun*	*Prep*	*Noun*	*Noun*
British	Left	Waffles	on	Falkland	Islands

Task: Predict a **part-of-speech** tag for each word in a provided sentence

Example 1: Part of Speech Tagging



Example 2: Named Entity Recognition

Task: Predict a **named entity** tag for each word in a provided sentence

British Left Waffles on Falkland Islands

Reminder!

What are Named Entities?

Named entity recognition (NER)

Identify proper names in texts, and classification into a set of predefined categories of interest

- Person names
- Organizations (companies, government organisations, committees, etc)
- Locations (cities, countries, rivers, etc)
- Date and time expressions
- Measures (percent, money, weight etc), email addresses, Web addresses, street addresses, etc.
- Domain-specific: names of drugs, medical conditions, names of ships, bibliographic references etc.

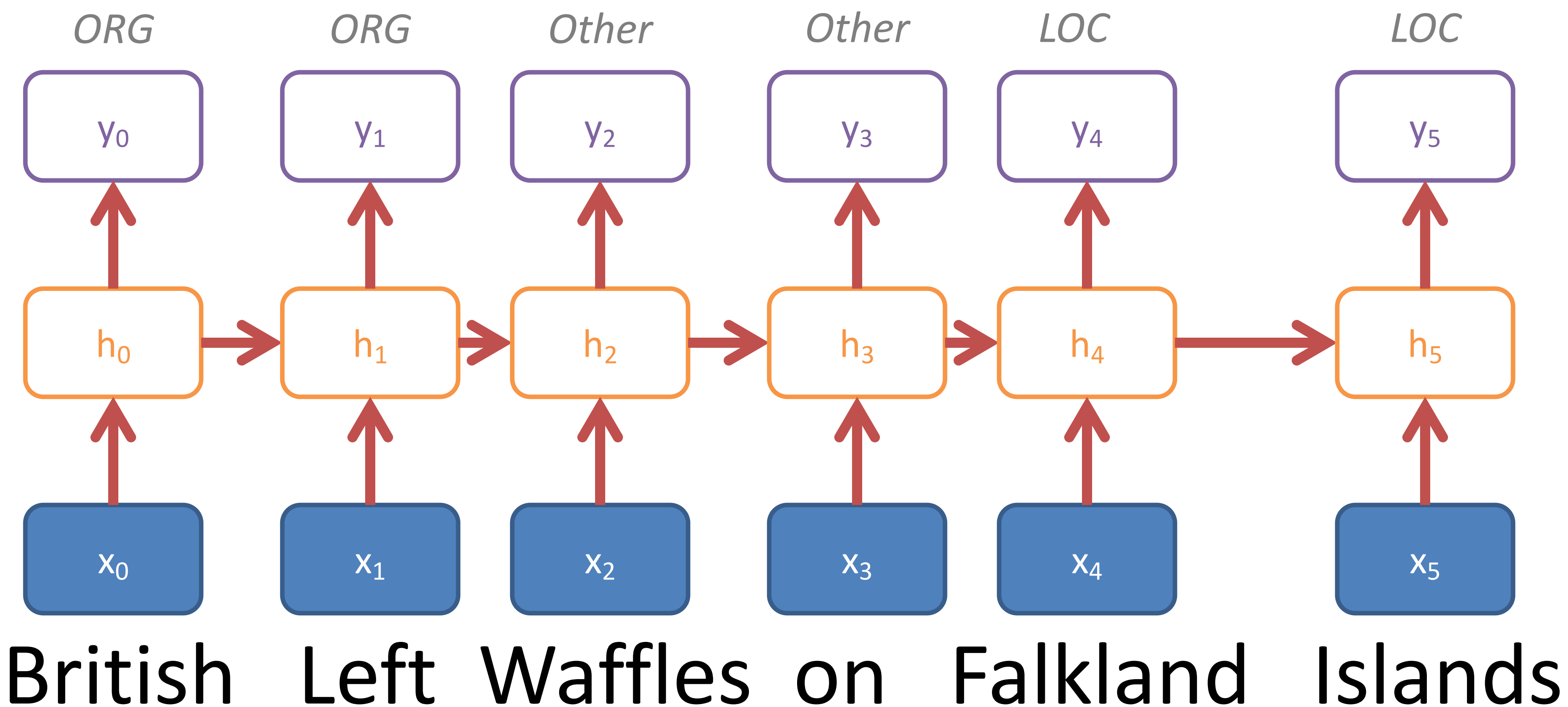
Cunningham and Bontcheva (2003, RANLP Tutorial)

Example 2: Named Entity Recognition

ORG	*ORG*	*Other*	*Other*	*LOC*	*LOC*
British	Left	Waffles	on	Falkland	Islands

Task: Predict a **named entity** tag for each word in a provided sentence

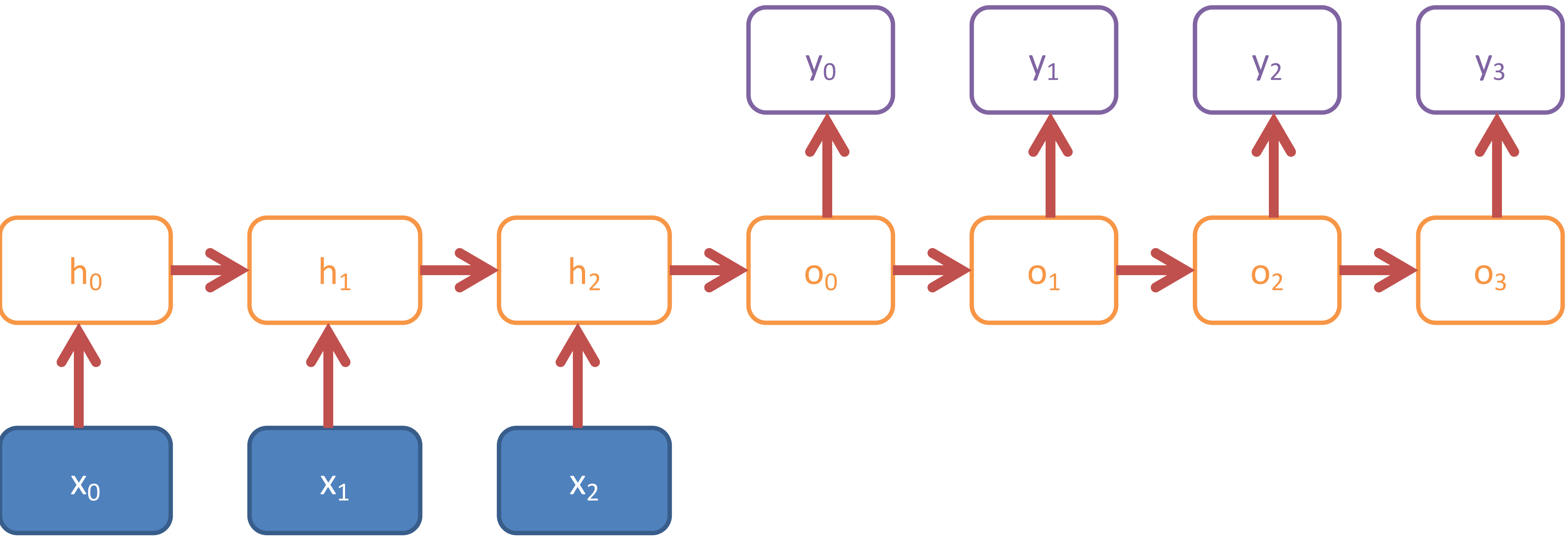
Example: Named Entity Recognition



How to evaluate sequence prediction w/o no time delay (*ForTokenClassification)

- Treat it as a standard prediction task
  - e.g., Accuracy, Precision, Recall, F1
- Most common: metric per token prediction
  - For every token, did you make the correct prediction?
- Less common but still helpful: metric per sequence
  - For every sequence, was the entire sequence correct?

Sequence Input, Sequence Output ("sequence-to-sequence": time delay)



Recursive: Sequence input, Sequence output (time delay)

Machine translation
Sequential description
Summarization

Name	Input	Output	Tasks	Ex. Datasets
		Heads		
Language Modeling	$x_{1:n-1}$	$x_n \in \mathcal{V}$	Generation	WikiText-103
Sequence Classification	$x_{1:N}$	$y \in \mathcal{C}$	Classification, Sentiment Analysis	GLUE, SST, MNLI
Question Answering	$x_{1:M}, x_{M:N}$	y span $[1:N]$	QA, Reading Comprehension	SQuAD, Natural Questions
Token Classification	$x_{1:N}$	$y_{1:N} \in \mathcal{C}^N$	NER, Tagging	OntoNotes, WNUT
Multiple Choice	$x_{1:N}, \mathcal{X}$	$y \in \mathcal{X}$	Text Selection	SWAG, ARC
Masked LM	$x_{1:N \setminus n}$	$x_n \in \mathcal{V}$	Pretraining	Wikitext, C4
Conditional Generation	$x_{1:N}$	$y_{1:M} \in \mathcal{V}^M$	Translation, Summarization	WMT, IWSLT, CNN/DM, XSum

Example: Translation

Translate English (observed) into French:

The cat is on the chair.

variable # of input words

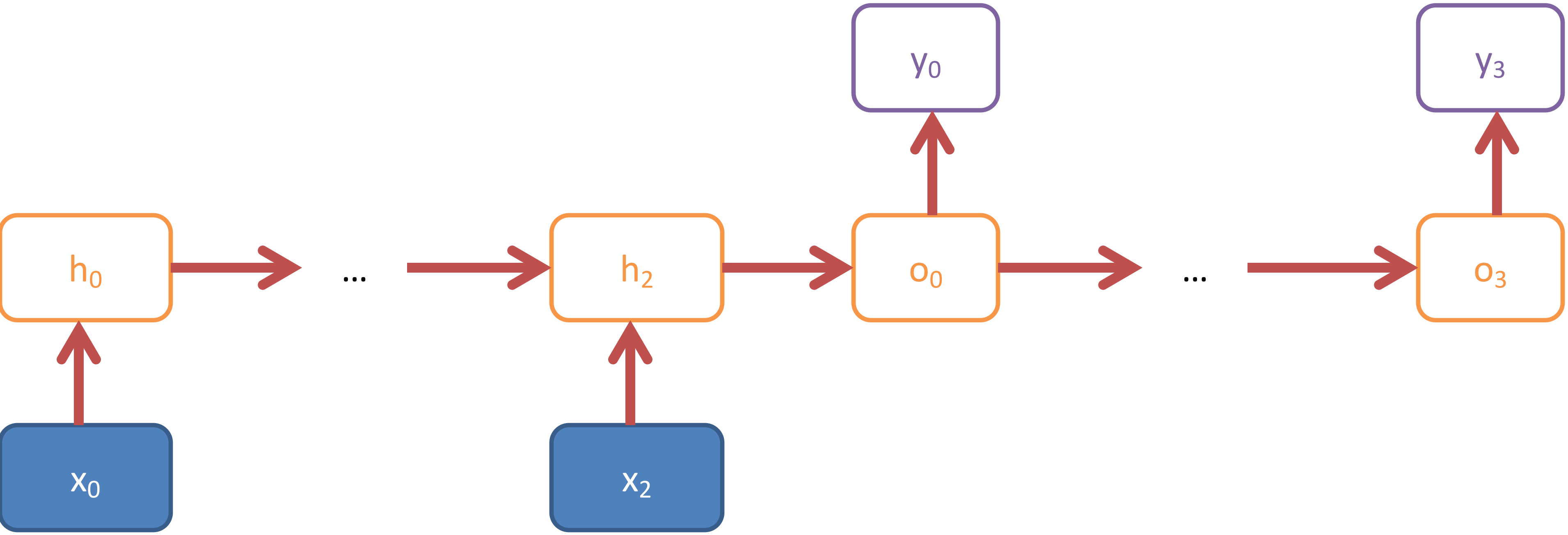
Le chat est sur la chaise.

variable # of output words

Example: Translation

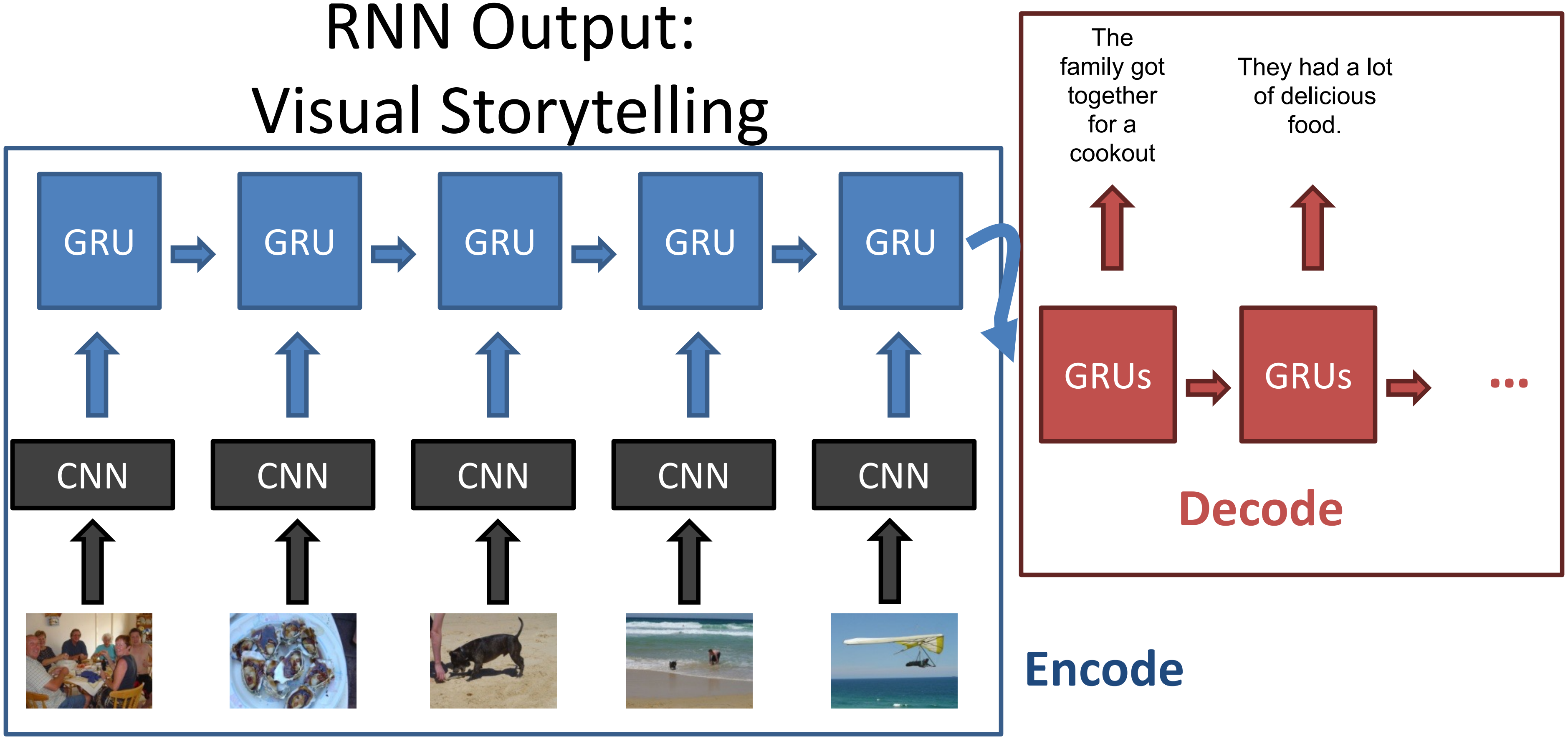
Translate English (observed) into French:

Le chat est sur la chaise.



The cat is on the chair.

RNN Output:
Visual Storytelling



Human Reference

The family has gathered around the dinner table to share a meal together. They all pitched in to help cook the seafood to perfection. Afterwards they took the family dog to the beach to get some exercise. The waves were cool and refreshing! The dog had so much fun in the water. One family member decided to get a better view of the waves!

Huang et al. (2016)

The family got together for a cookout. They had a lot of delicious food. The dog was happy to be there. They had a great time on the beach. They even had a swim in the water.

Teacher Forcing vs. No Teacher Forcing

(then negate, average)

log .2 + log .12 + log .2 + log .19 + log .3 + log .2 + log .2 + log .2

word	prob.
The	.2
gray	.01
blue	.001
fluffy	.0005
wet	.0005
…	…

word	prob.
black	.2
gray	.12
blue	.001
fluffy	.0005
wet	.0005
…	…

word	prob.
fluffy	.2
gray	.01
blue	.001
bald	.0005
wet	.0005
…	…

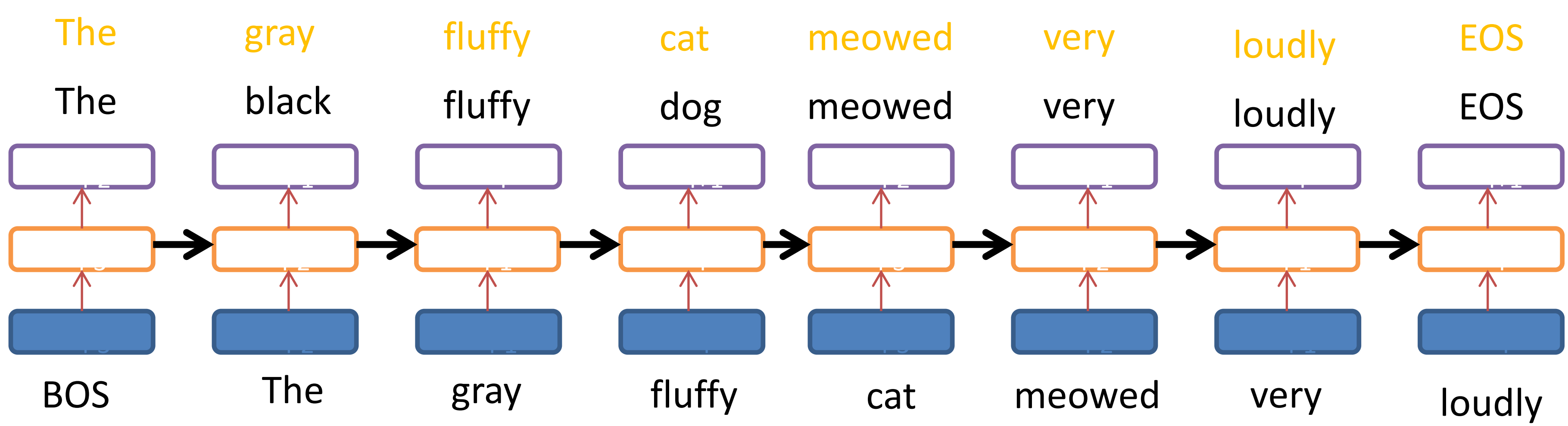
word	prob.
dog	.2
cat	.19
blue	.001
fluffy	.0005
wet	.0005
…	…

word	prob
meowed	.3
purred	.2
hissed	.1
fluffy	.001
wet	.001
…	…

word	prob.
very	.2
lots	.1
softly	. 1
fluffy	.0005
wet	.0005
…	…

word	prob
loudly	.2
softly	.01
quiet	.001
fluffy	.001
wet	.001
…	…

word	prob.
EOS	.3
and	.1
blue	.001
fluffy	.0005
wet	.0005
…	…



Teacher Forcing vs. No Teacher Forcing

(then negate, average)

$$\log .2 \; + \; \log .12 \; + \; \log .2 \; + \; \log .19 \; + \; \log .3 \; + \; \log .2 \; + \; \log .2 \; + \; \log .2$$

word	prob.
The	.2
gray	.01
blue	.001
fluffy	.0005
wet	.0005
…	…

word	prob.
black	.2
gray	.12
blue	.001
fluffy	.0005
wet	.0005
…	…

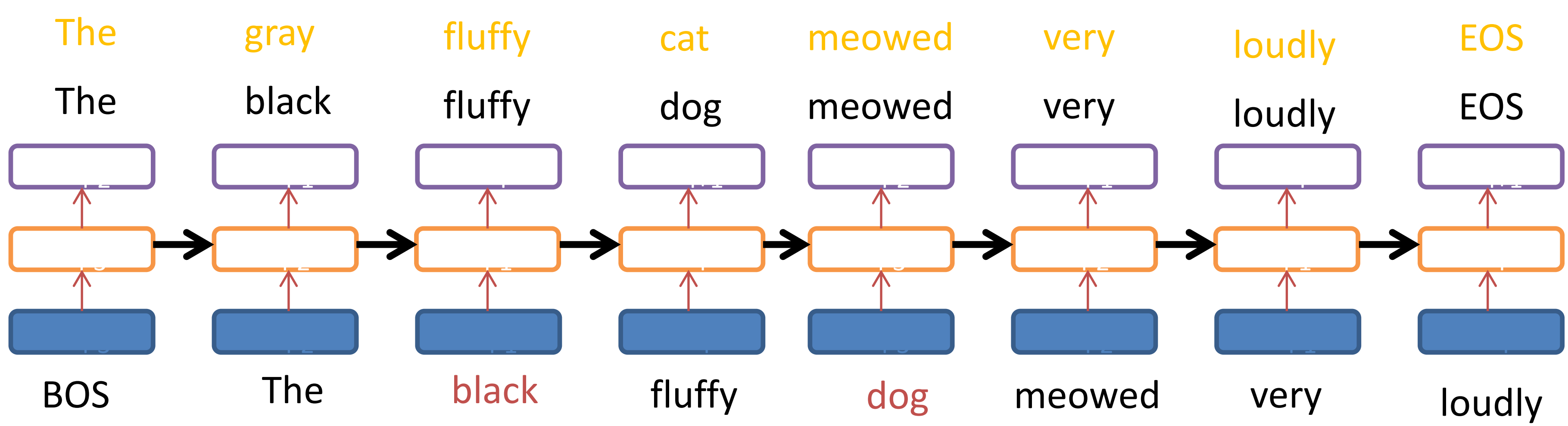
word	prob.
fluffy	.2
gray	.01
blue	.001
bald	.0005
wet	.0005
…	…

word	prob.	word	prob
dog	.2	meowed	.3
cat	.19	purred	.2
blue	.001	hissed	.1
fluffy	.0005	fluffy	.001
wet	.0005	wet	.001
…	…	…	…

word	prob.
very	.2
lots	.1
softly	. 1
fluffy	.0005
wet	.0005
…	…

word	prob
loudly	.2
softly	.01
quiet	.001
fluffy	.001
wet	.001
…	…

word	prob.
EOS	.3
and	.1
blue	.001
fluffy	.0005
wet	.0005
…	…



How to evaluate sequence prediction with no time delay (*ForConditionalGeneration)

Human Eval

- Get responses from your model
- Develop a questionnaire
- Show responses to human evaluators, getting "goodness"
  - Goodness can be: fluency, coherence, appropriateness, etc. Very task dependent.
- Single response vs. comparison

Automatic Eval

Human Eval: Single Response vs. Comparison

Single Response Example

For a translation task:

Original: The cat is on the chair.

Proposed translation: Le chat est sur la chaise.

Question: Is this a “good” translation?

Human Eval: Single Response vs. Comparison

Single Response Example

For a translation task:

Original: The cat is on the chair.

Proposed translation: Le chat est sur la chaise.

Question: Is this a “good” translation?

Comparison

For a translation task:

Original: The cat is on the chair.

Proposed translation 1: Le chat est sur la chaise.

Proposed translation 2: Le chat sont sur la chaise.

Question: Which translation do you “prefer?”

How to evaluate sequence prediction with no time delay (*ForConditionalGeneration)

Human Eval

- Get responses from your model
- Develop a questionnaire
- Show responses to human evaluators, getting "goodness"
  - Goodness can be: fluency, coherence, appropriateness, etc. Very task dependent.
- Single response vs. comparison

Automatic Eval

- "Accuracy"-based
  - Perplexity (maybe but can overfit; not always favored)
  - Word error rate
- "Precision"-based
  - BLEU (word n-gram overlap)
- "Recall"-based
  - ROUGE (word n-gram overlap)
- "F1"-based
  - METEOR (but only unigram)
- Embedding based
  - BERTScore (ICLR 2020; https://openreview.net/pdf?id=SkeHuCVFDr)

Huggingface evaluate library: https://huggingface.co/docs/evaluate/index

A note on {BLEU, ROUGE}

- Terminology
  - “Hypotheses”: predictions
  - “References”: targets / gold labels
- Just as there are macro and micro {precision, recall}, we have similar notions here
  - “corpus” {BLEU, ROUGE} → micro
  - “sentence” {BLEU, ROUGE} → macro

Key Highlights (1/3)

- While there are a number of different types of networks, it's helpful to think of them as *encoding* (learning to featurize) the input, and then making an appropriate prediction ("decode")

Single Input, Single Output

Single Input, Multiple Outputs

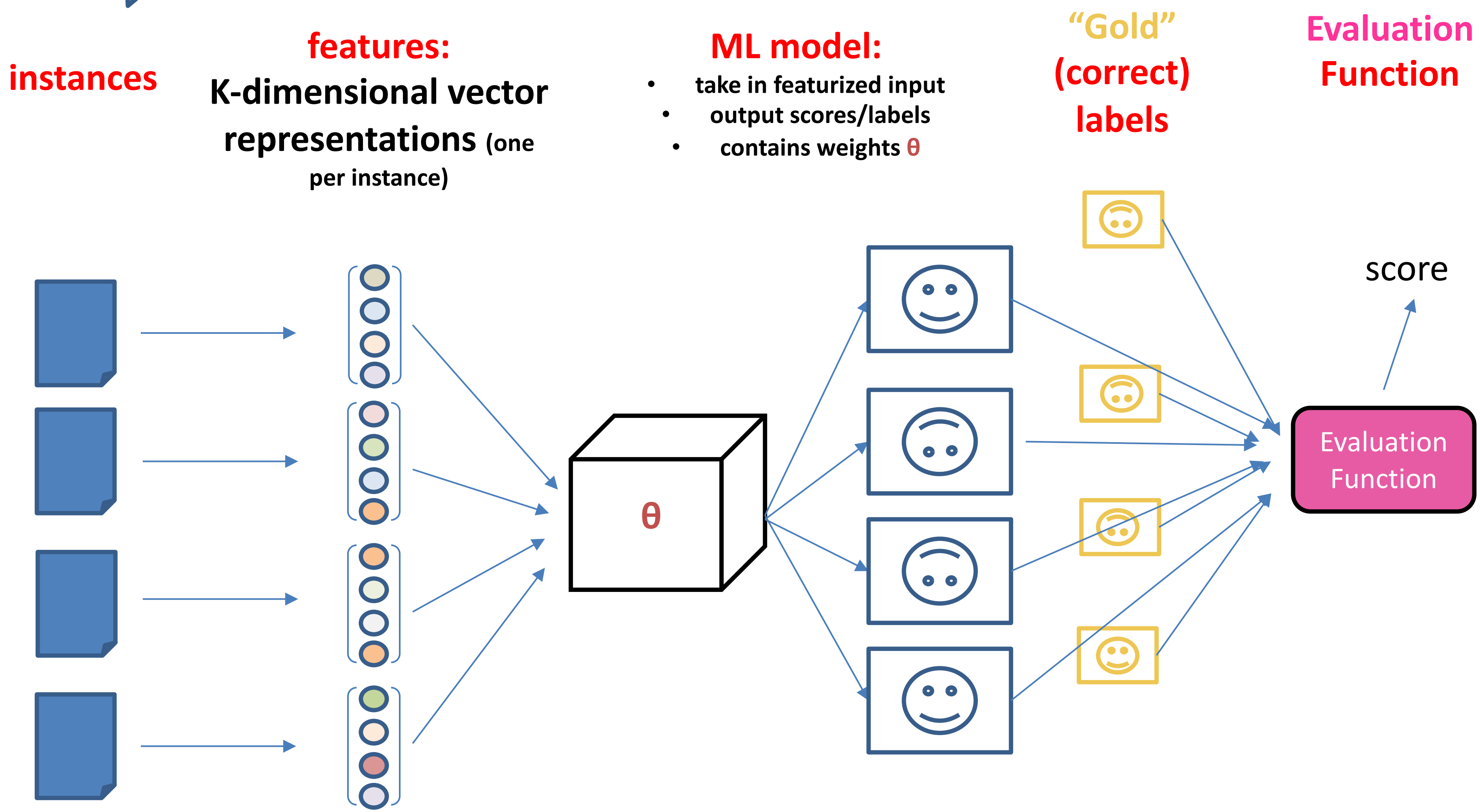
Multiple Inputs, Single Output

Multiple Inputs, Multiple Outputs ("sequence prediction": no time delay)

Multiple Inputs, Multiple Outputs ("sequence-to-sequence": with time delay); also called "encoder-decoder")

Name	Heads Input	Heads Output	Tasks	Ex. Datasets
Language Modeling	$x_{1:n-1}$	$x_n \in \mathcal{V}$	Generation	WikiText-103
Sequence Classification	$x_{1:N}$	$y \in \mathcal{C}$	Classification, Sentiment Analysis	GLUE, SST, MNLI
Question Answering	$x_{1:M}, x_{M:N}$	y span $[1:N]$	QA, Reading Comprehension	SQuAD, Natural Questions
Token Classification	$x_{1:N}$	$y_{1:N} \in \mathcal{C}^N$	NER, Tagging	OntoNotes, WNUT
Multiple Choice	$x_{1:N}, \mathcal{X}$	$y \in \mathcal{X}$	Text Selection	SWAG, ARC
Masked LM	$x_{1:N \setminus n}$	$x_n \in \mathcal{V}$	Pretraining	Wikitext, C4
Conditional Generation	$x_{1:N}$	$y_{1:M} \in \mathcal{V}^M$	Translation, Summarization	WMT, IWSLT, CNN/DM, XSum

ML/NLP Framework for Prediction



Reminder (from deck 3)!

Helpful ML Terminology Recap (3)

Learning:

the process of adjusting the model's weights to learn to make good predictions.

Inference / Prediction / Decoding / Classification:

the process of using a model's existing weights to make (hopefully!) good predictions

A Couple Notes on "Encoder-Decoder" Models

- *Many* people use the term "encoder-decoder" to describe the "sequence-to-sequence with time delay" model type. But…

- "Encoder-decoder" terminology is quite broad

Single Input, Single Output

Single Input, Multiple Outputs

Multiple Inputs, Single Output

Multiple Inputs, Multiple Outputs ("sequence prediction": no time delay)

Multiple Inputs, Multiple Outputs ("sequence-to-sequence": with time delay)

Key Highlights (2/3)

- While there are a number of different types of networks, it's helpful to think of them as *encoding* (learning to featurize) the input, and then making an appropriate prediction ("decode")
- This *encoding* is driven by learning what is effective for language modeling

Single Input, Single Output

Single Input, Multiple Outputs

Multiple Inputs, Single Output

Multiple Inputs, Multiple Outputs ("sequence prediction": no time delay)

Multiple Inputs, Multiple Outputs ("sequence-to-sequence": with time delay)

	Heads			
Name	Input	Output	Tasks	Ex. Datasets
Language Modeling	$x_{1:n-1}$	$x_n \in \mathcal{V}$	Generation	WikiText-103
Sequence Classification	$x_{1:N}$	$y \in \mathcal{C}$	Classification, Sentiment Analysis	GLUE, SST, MNLI
Question Answering	$x_{1:M}, x_{M:N}$	y span $[1:N]$	QA, Reading Comprehension	SQuAD, Natural Questions
Token Classification	$x_{1:N}$	$y_{1:N} \in \mathcal{C}^N$	NER, Tagging	OntoNotes, WNUT
Multiple Choice	$x_{1:N}, \mathcal{X}$	$y \in \mathcal{X}$	Text Selection	SWAG, ARC
Masked LM	$x_{1:N \setminus n}$	$x_n \in \mathcal{V}$	Pretraining	Wikitext, C4
Conditional Generation	$x_{1:N}$	$y_{1:M} \in \mathcal{V}^M$	Translation, Summarization	WMT, IWSLT, CNN/DM, XSum

Key Highlights (3/3)

- While there are a number of different types of networks, it's helpful to think of them as *encoding* (learning to featurize) the input, and then making an appropriate prediction ("decode")
- This *encoding* is driven by learning what is effective for language modeling
- This *decoding* can be "prediction" or "language modeling"

Single Input, Single Output

Single Input, Multiple Outputs

Multiple Inputs, Single Output

Multiple Inputs, Multiple Outputs ("sequence prediction": no time delay)

Multiple Inputs, Multiple Outputs ("sequence-to-sequence": with time delay)

	Heads			
Name	Input	Output	Tasks	Ex. Datasets
Language Modeling	$x_{1:n-1}$	$x_n \in \mathcal{V}$	Generation	WikiText-103
Sequence Classification	$x_{1:N}$	$y \in \mathcal{C}$	Classification, Sentiment Analysis	GLUE, SST, MNLI
Question Answering	$x_{1:M}, x_{M:N}$	y span $[1:N]$	QA, Reading Comprehension	SQuAD, Natural Questions
Token Classification	$x_{1:N}$	$y_{1:N} \in \mathcal{C}^N$	NER, Tagging	OntoNotes, WNUT
Multiple Choice	$x_{1:N}, \mathcal{X}$	$y \in \mathcal{X}$	Text Selection	SWAG, ARC
Masked LM	$x_{1:N \setminus n}$	$x_n \in \mathcal{V}$	Pretraining	Wikitext, C4
Conditional Generation	$x_{1:N}$	$y_{1:M} \in \mathcal{V}^M$	Translation, Summarization	WMT, IWSLT, CNN/DM, XSum

Some Consequences of these Key Highlights

- While there are a number of different types of networks, it's helpful to think of them as *encoding* (learning to featurize) the input, and then making an appropriate prediction ("decode")
- This *encoding* is driven by learning what is effective for language modeling
- This *decoding* can be "prediction" or "language modeling"

Encoding: use a modeling structure that is effective. This could be:

- a bag-of-words style, or
- an auto-regressive (left-to-right) encoder, or
- a bi-directional / auto-encoding encoder

Decoding: an *auto-regressive* (left-to-right) structure, e.g., process one item, then another, then another

Encoder vs. Decoder

Encoding: use a modeling structure that is effective. This could be:

- a bag-of-words style, or
- an auto-regressive (left-to-right) encoder, or
- a bi-directional / auto-encoding encoder

Decoding: an *auto-regressive* (left-to-right) structure, e.g., process one item, then another, then another

Transformers	
Masked $[x_{1:N \backslash n} \Rightarrow x_n]$	
BERT	(Devlin et al., 2018)
RoBERTa	(Liu et al., 2019a)
Autoregressive $[x_{1:n-1} \Rightarrow x_n]$	
GPT / GPT-2	(Radford et al., 2019)
Trans-XL	(Dai et al., 2019)
XLNet	(Yang et al., 2019)
Seq-to-Seq $[\sim x_{1:N} \Rightarrow x_{1:N}]$	
BART	(Lewis et al., 2019)
T5	(Raffel et al., 2019)
Marian	(J.-Dowmunt et al., 2018)

Fig. 2 (Wolf et al., 2020: https://arxiv.org/pdf/1910.03771.pdf)

Outline

Transformer Language Models as General Language Encoders

The Attention Mechanism

Two Well-Known (Recent) Instances of Learning from Language Models

GPT2 [Radford et al., 2018]

Language Models are Unsupervised Multitask Learners

Alec Radford [*1] Jeffrey Wu [*1] Rewon Child [1] David Luan [1] Dario Amodei [**1] Ilya Sutskever [**1]

Abstract

Natural language processing tasks, such as question answering, machine translation, reading comprehension, and summarization, are typically approached with supervised learning on task-specific datasets. We demonstrate that language models begin to learn these tasks without any explicit supervision when trained on a new dataset of millions of webpages called WebText. When conditioned on a document plus questions, the answers generated by the language model reach 55 F1 on the CoQA dataset - matching or exceeding the performance of 3 out of 4 baseline systems without using the 127,000+ training examples. The capacity of the language model is essential to the success of zero-shot task transfer and increasing it improves performance in a log-linear fashion across tasks. Our largest model, GPT-2, is a 1.5B parameter Transformer that achieves state of the art results on 7 out of 8 tested language modeling datasets in a zero-shot setting but still underfits WebText. Samples from the model reflect these improvements and contain coherent paragraphs of text. These findings suggest a promising path towards building language processing systems which learn to perform tasks from their naturally occurring demonstrations.

competent generalists. We would like to move towards more general systems which can perform many tasks – eventually without the need to manually create and label a training dataset for each one.

The dominant approach to creating ML systems is to collect a dataset of training examples demonstrating correct behavior for a desired task, train a system to imitate these behaviors, and then test its performance on independent and identically distributed (IID) held-out examples. This has served well to make progress on narrow experts. But the often erratic behavior of captioning models (Lake et al., 2017), reading comprehension systems (Jia & Liang, 2017), and image classifiers (Alcorn et al., 2018) on the diversity and variety of possible inputs highlights some of the shortcomings of this approach.

Our suspicion is that the prevalence of single task training on single domain datasets is a major contributor to the lack of generalization observed in current systems. Progress towards robust systems with current architectures is likely to require training and measuring performance on a wide range of domains and tasks. Recently, several benchmarks have been proposed such as GLUE (Wang et al., 2018) and decaNLP (McCann et al., 2018) to begin studying this.

Multitask learning (Caruana, 1997) is a promising framework for improving general performance. However, multitask training in NLP is still nascent. Recent work reports modest performance improvements (Yogatama et al.,

BERT [Devlin et al., 2019 NAACL)

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

Abstract

We introduce a new language representation model called **BERT**, which stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.

BERT is conceptually simple and empirically powerful. It obtains new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE score to

There are two existing strategies for applying pre-trained language representations to downstream tasks: *feature-based* and *fine-tuning*. The feature-based approach, such as ELMo (Peters et al., 2018a), uses task-specific architectures that include the pre-trained representations as additional features. The fine-tuning approach, such as the Generative Pre-trained Transformer (OpenAI GPT) (Radford et al., 2018), introduces minimal task-specific parameters, and is trained on the downstream tasks by simply fine-tuning *all* pre-trained parameters. The two approaches share the same objective function during pre-training, where they use unidirectional language models to learn general language representations.

We argue that current techniques restrict the power of the pre-trained representations, especially for the fine-tuning approaches. The major limitation is that standard language models are

GPT-2 & BERT (and others) In Practice

- Use pytorch code

  → The huggingface transformers package is very popular

  Some hooks for tensorflow code
- Read the documentation!
  - (but it may be dense): open a REPL/Colab notebook and play around!

GPT-2 & BERT (and others) In Practice

- Use pytorch code

  → The huggingface transformers package is very popular

  Some hooks for tensorflow code
- Read the documentation!
  - (but it may be dense): open a REPL/Colab notebook and play around!
- Exception: GPT-3, GPT-4
  - Model not publicly downloadable, must access through a completely separate API
  - Quota-based

GPT-2 Take-Away

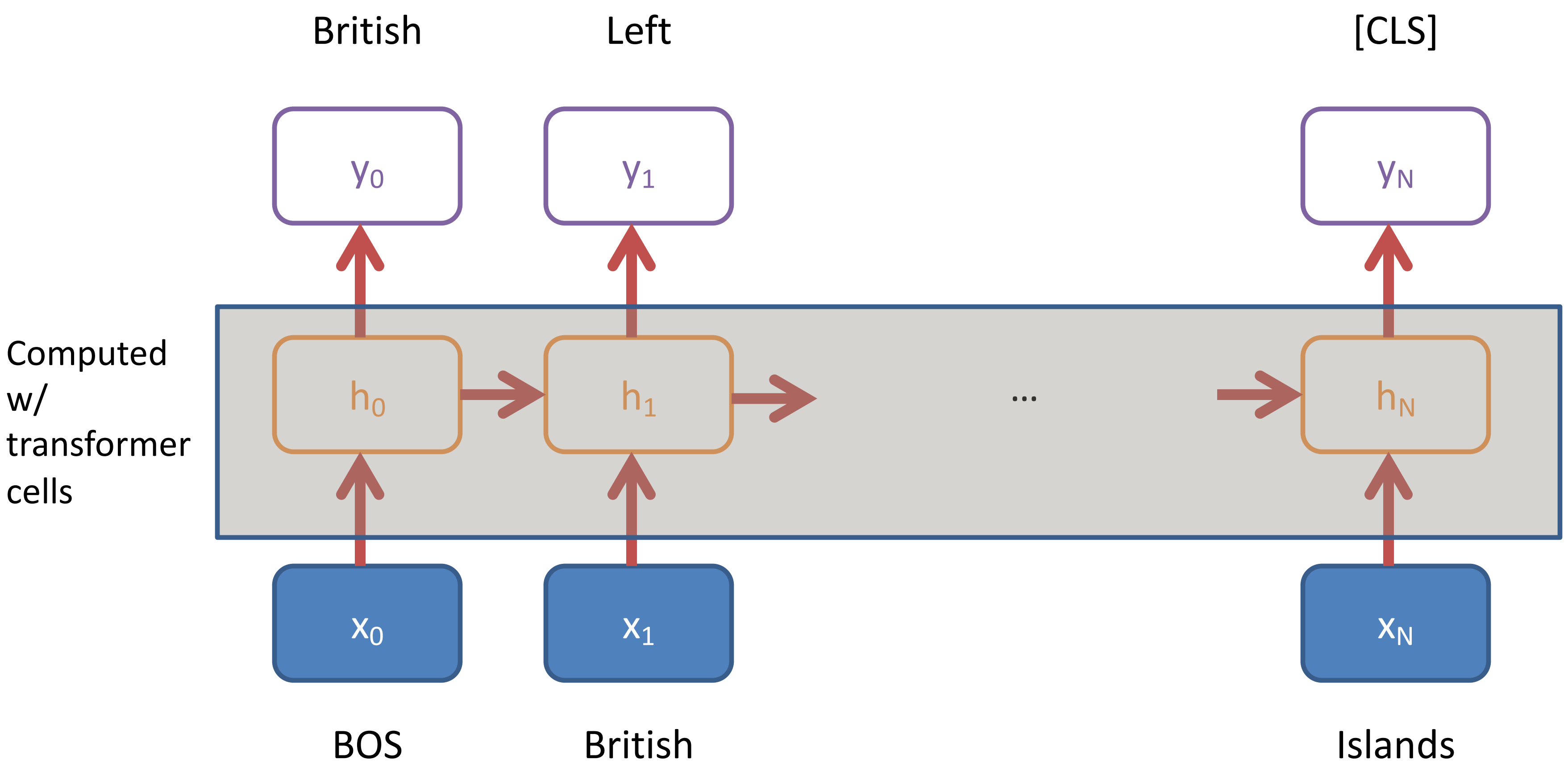
Language models can provide an effective way of learning embeddings that are useful for downstream tasks

- Auto-regressive model that uses a transformer cell

$$p(w_1 \dots w_N) = \prod_i p(w_i | w_1, \dots, w_{i-1})$$

https://openai.com/blog/gpt-2-1-5b-release/
https://github.com/openai/gpt-2

GPT-2 Model & Representation



BERT Take-Aways

1. Demonstration of bidirectional transformer for language understanding

2. Clean separation of “pre-training” and “fine-tuning” tasks

3. Clear demonstration that language model “pre-training” can yield useful embeddings

Pretraining vs. Fine-tuning

Pre-training

Learning an **encoder** to produce effective embeddings through "general" training objectives that are end-task agnostic

Pretraining vs. Fine-tuning

Pre-training

Learning an **encoder** to produce effective embeddings through "general" training objectives that are end-task agnostic

1. Next-sentence prediction [NSP]
2. Masked Language Modeling [MLM]

Pre-training: NSP

- Given two sentences s_1 and s_2, predict whether s_2 follows s_1 in "natural" text

Pretraining vs. Fine-tuning

Pre-training

Learning an **encoder** to produce effective embeddings through "general" training objectives that are end-task agnostic

1. Next-sentence prediction [NSP]
2. Masked Language Modeling [MLM]

Pre-training: MLM

- Given a sentence $s = w_1 \ldots w_N$, mask out (remove) a word w_i and predict what that word should be

"The cat chased the mouse" ➔ "The cat [MASK] the mouse"

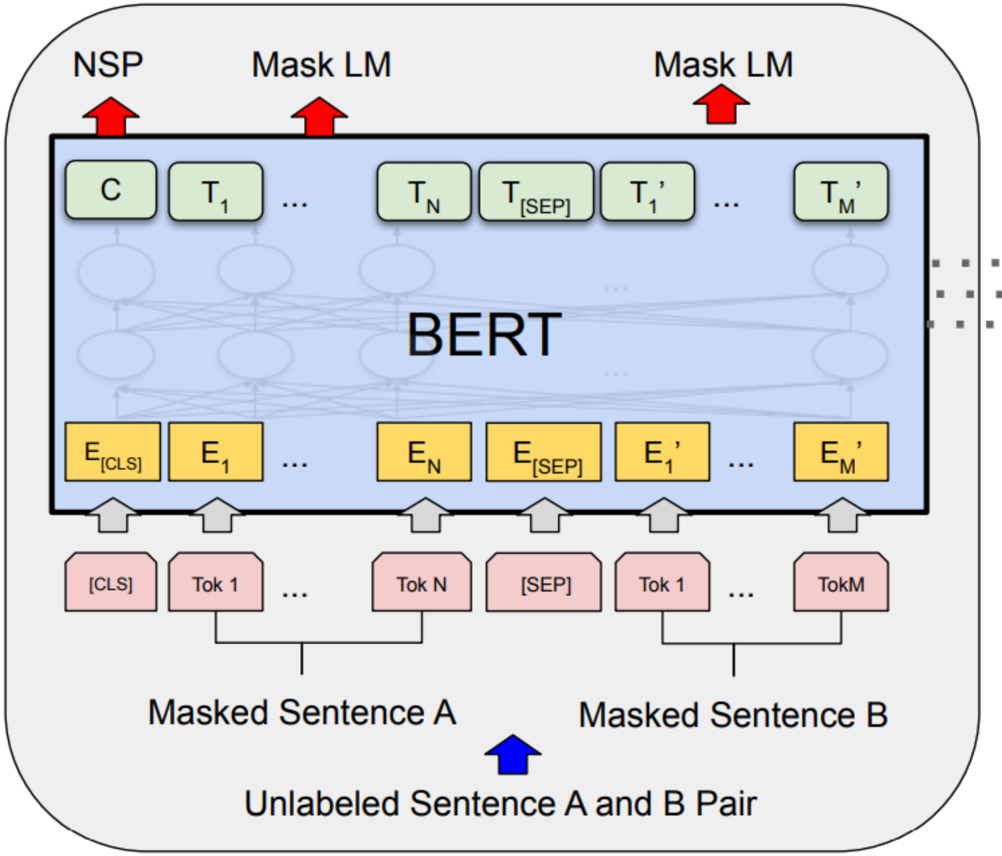
$$p(w \mid \text{The cat [MASK]the mouse})$$

Pretraining vs. Fine-tuning

Pre-training

Learning an **encoder** to produce effective embeddings through "general" training objectives that are end-task agnostic

1. Next-sentence prediction [NSP]
2. Masked Language Modeling [MLM]



Pre-training

Fig. 1

Pretraining vs. Fine-tuning

Pre-training

Learning an **encoder** to produce effective embeddings through "general" training objectives that are end-task agnostic

1. Next-sentence prediction [NSP]
2. Masked Language Modeling [MLM]

Fine-Tuning

Learning task-specific **decoders** using the embeddings produced from the pre-training, e.g.,

- RTE
- Question-answering
- <Your task here>

Pretraining vs. Fine-tuning

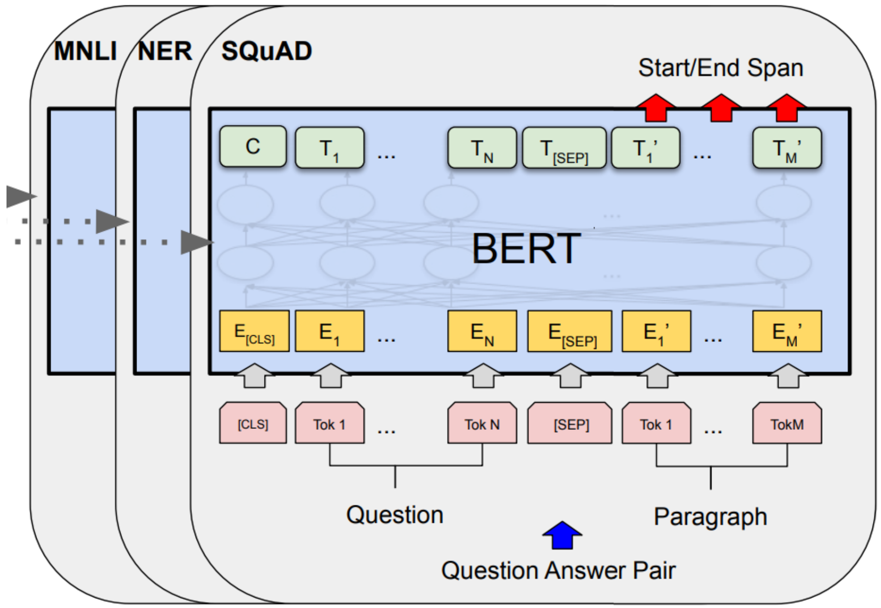


Fig. 1 Fine-Tuning

Fine-Tuning

Learning task-specific **decoders** using the embeddings produced from the pre-training, e.g.,

- RTE
- Question-answering
- <Your task here>

Pre-training *then* Fine-tuning

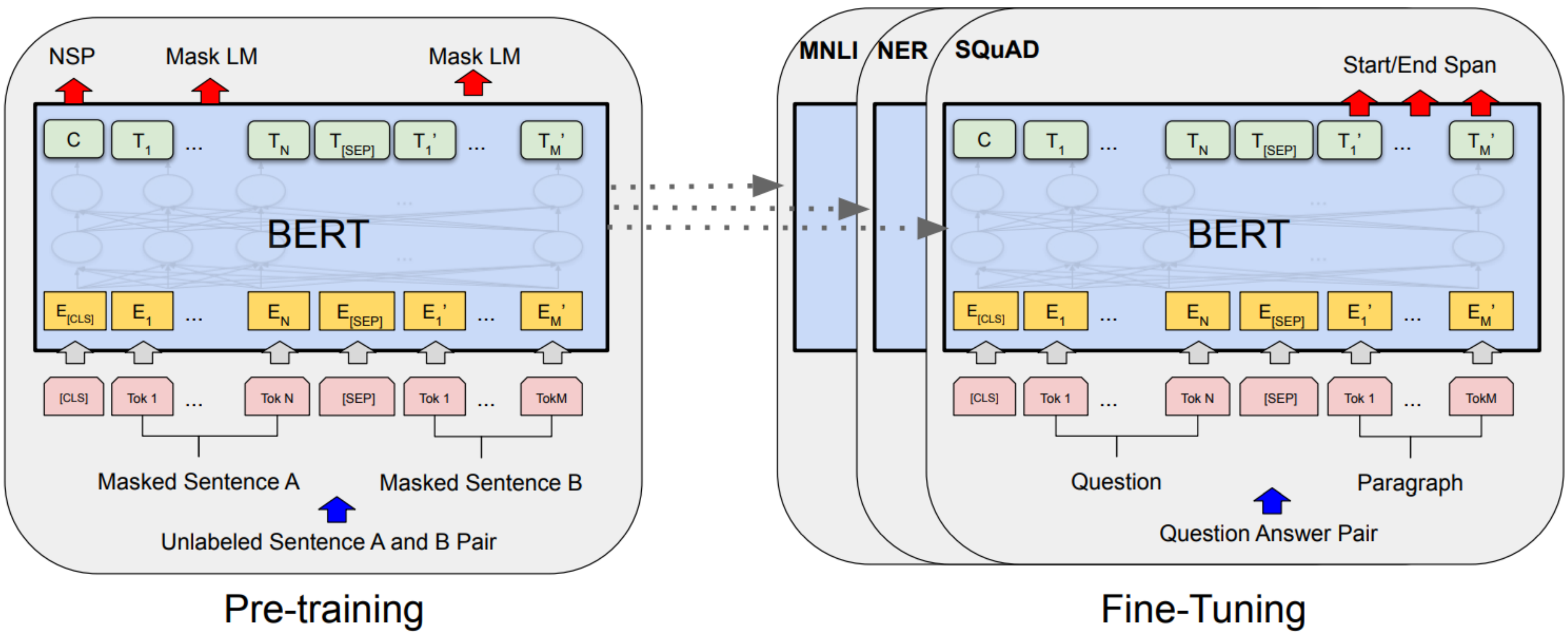
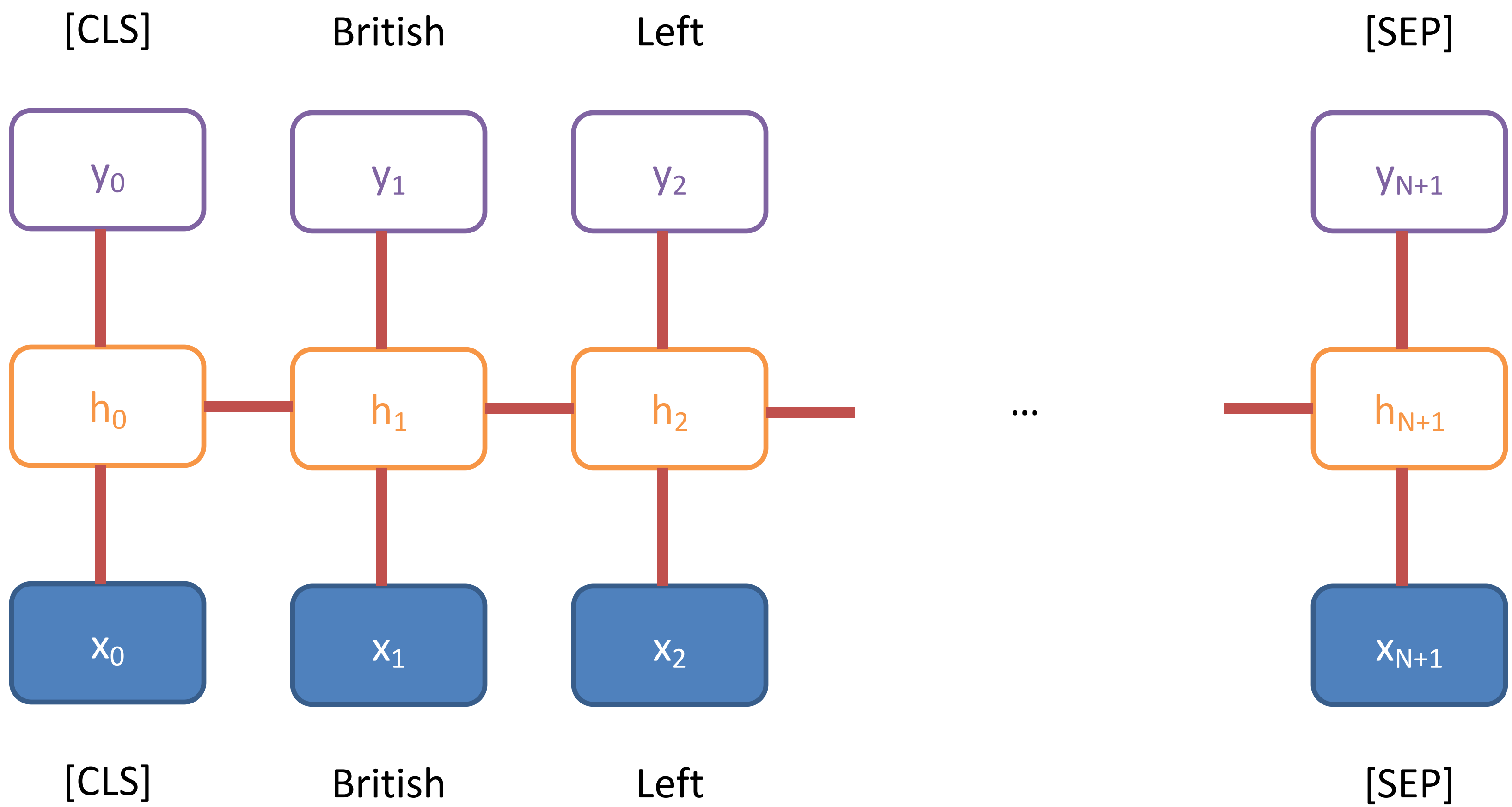


Fig. 1

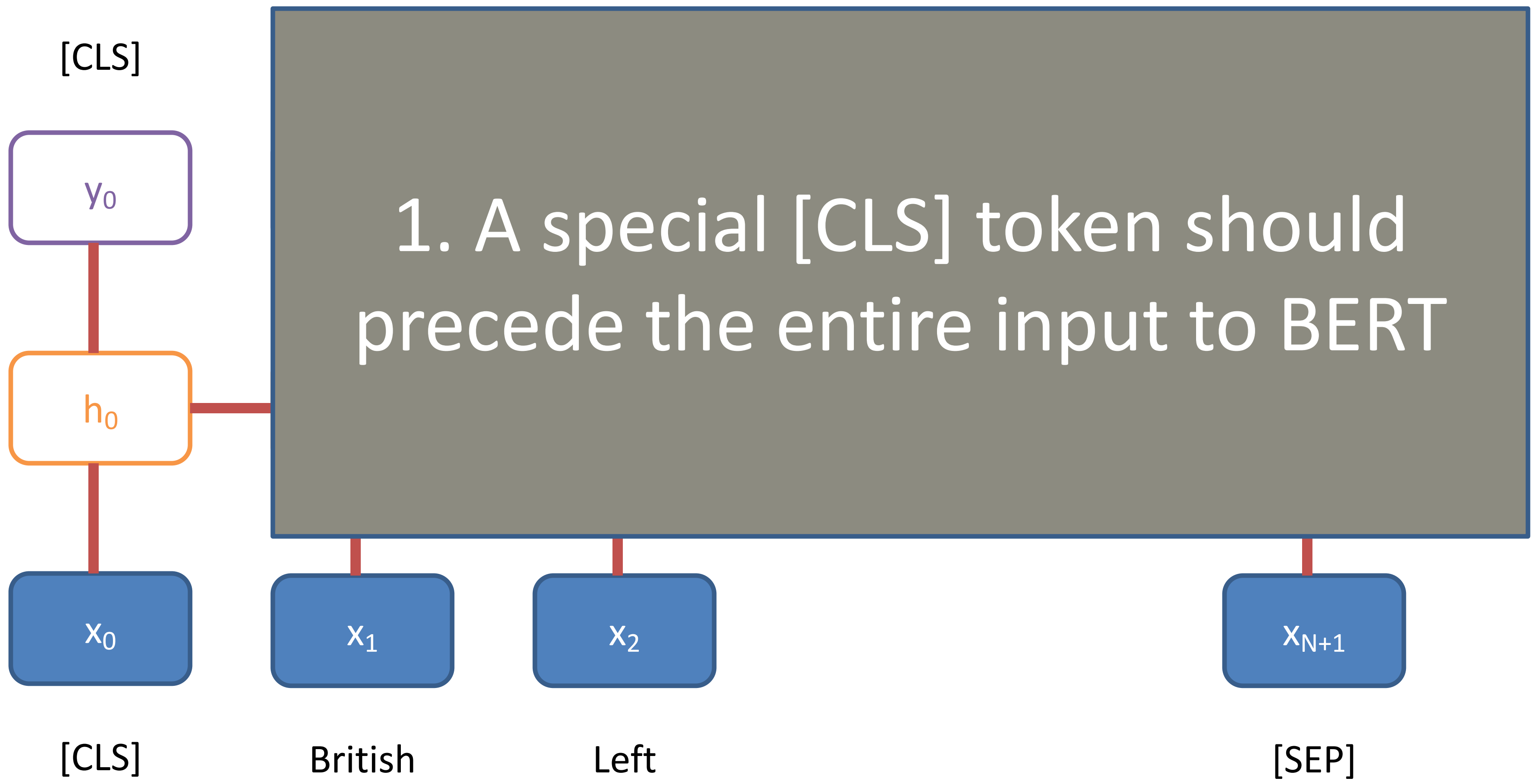
BERT Representation

1. A special [CLS] token should precede the entire input to BERT
2. Every sentence should be followed by a special [SEP] token
3. The input must be tokenized in a special way
4. Segment & *position* embeddings must be provided

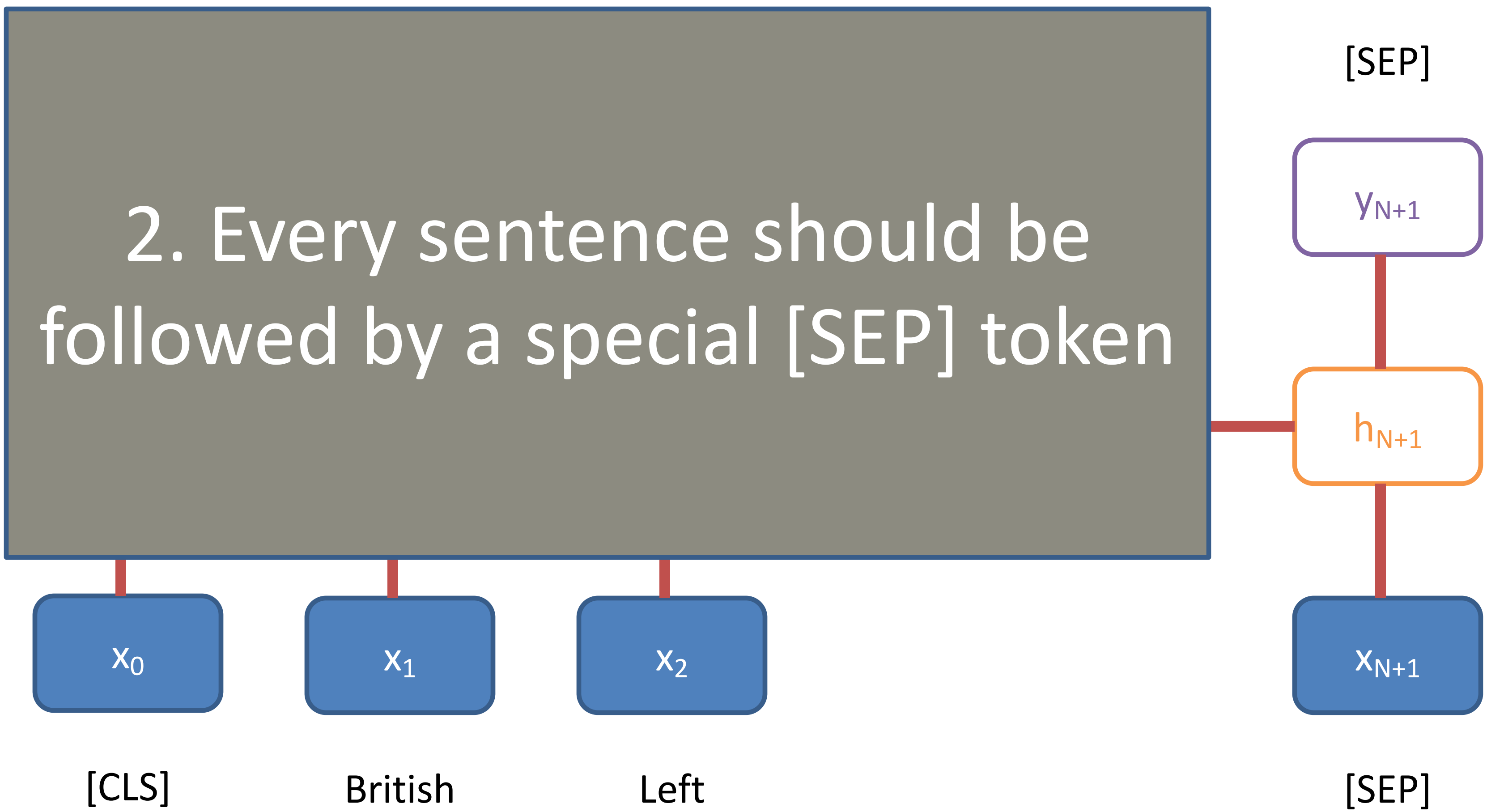
BERT Representation



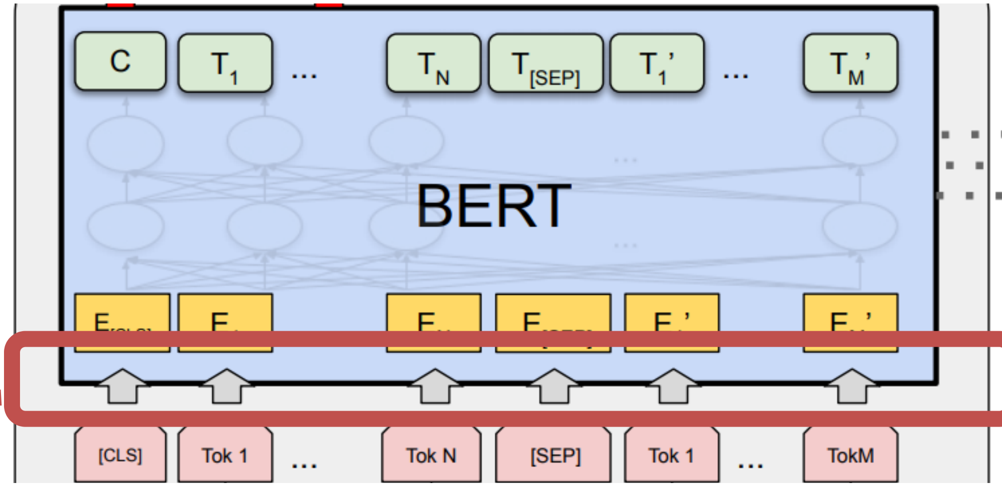
BERT Representation



BERT Representation



BERT Representation (Even More)



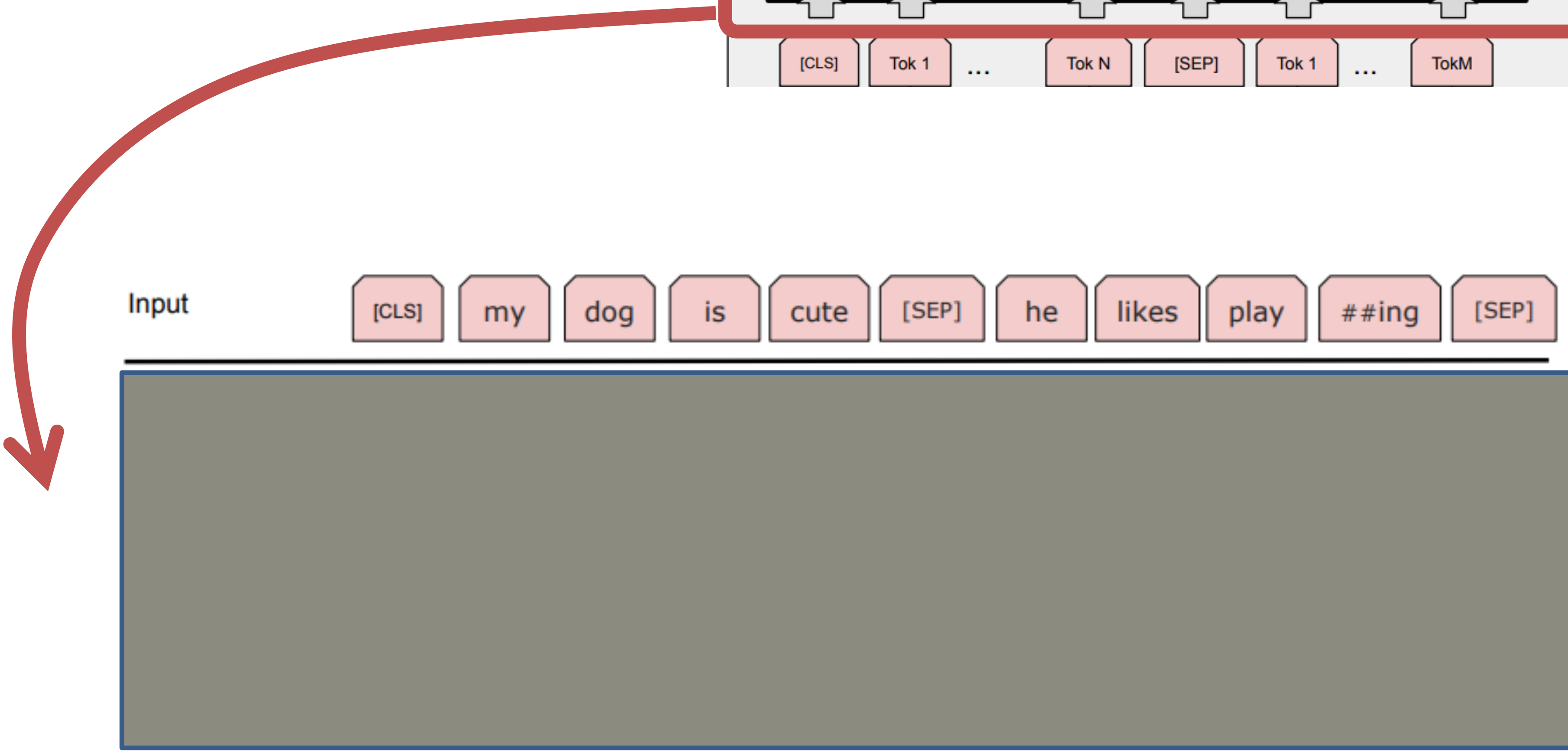


Fig. 2

BERT Representation (Even More)

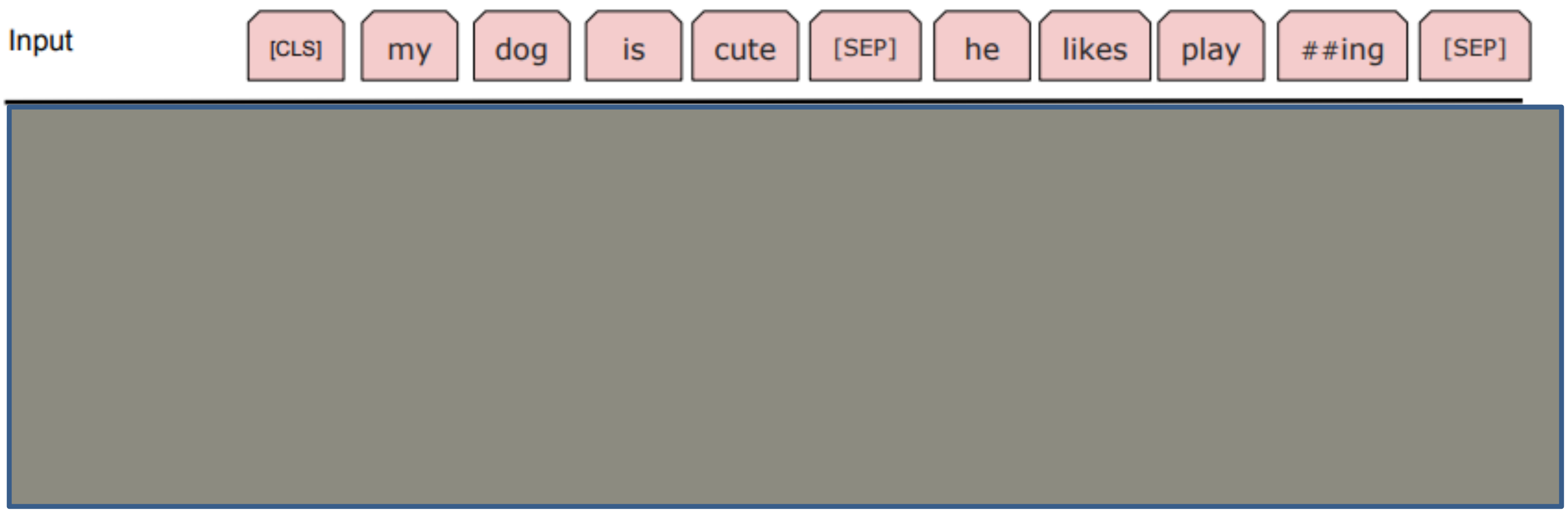
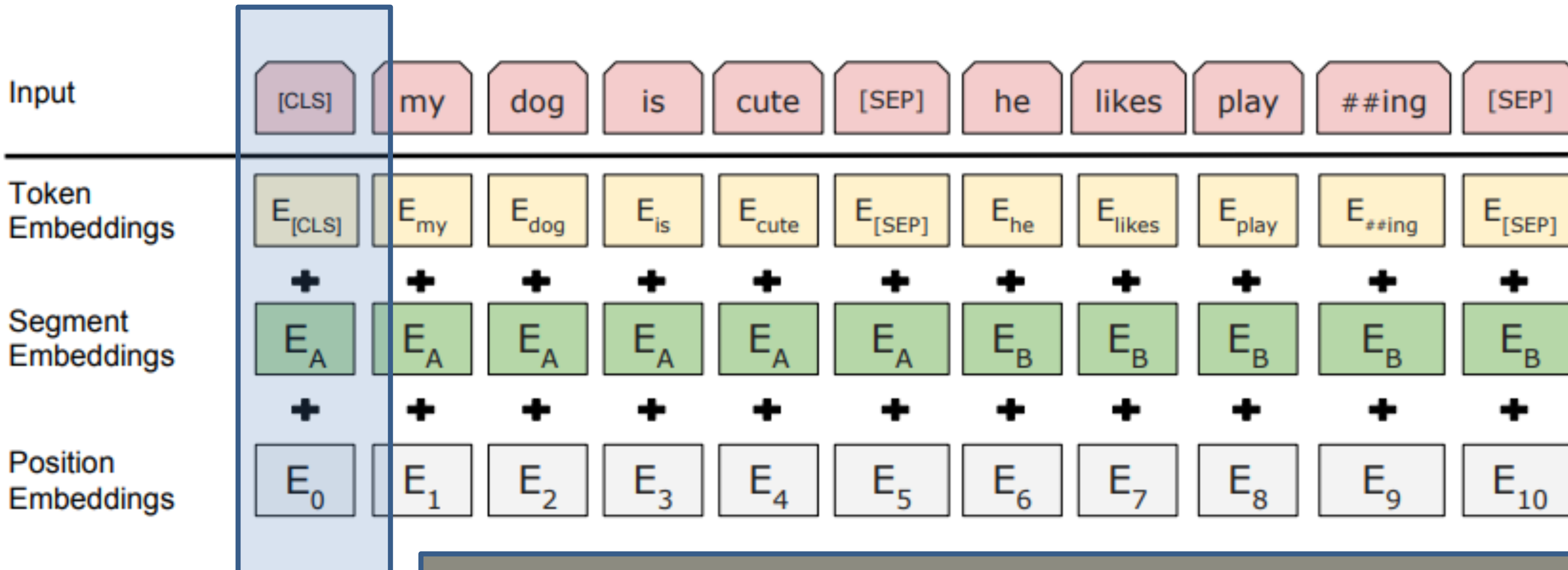


Fig. 2

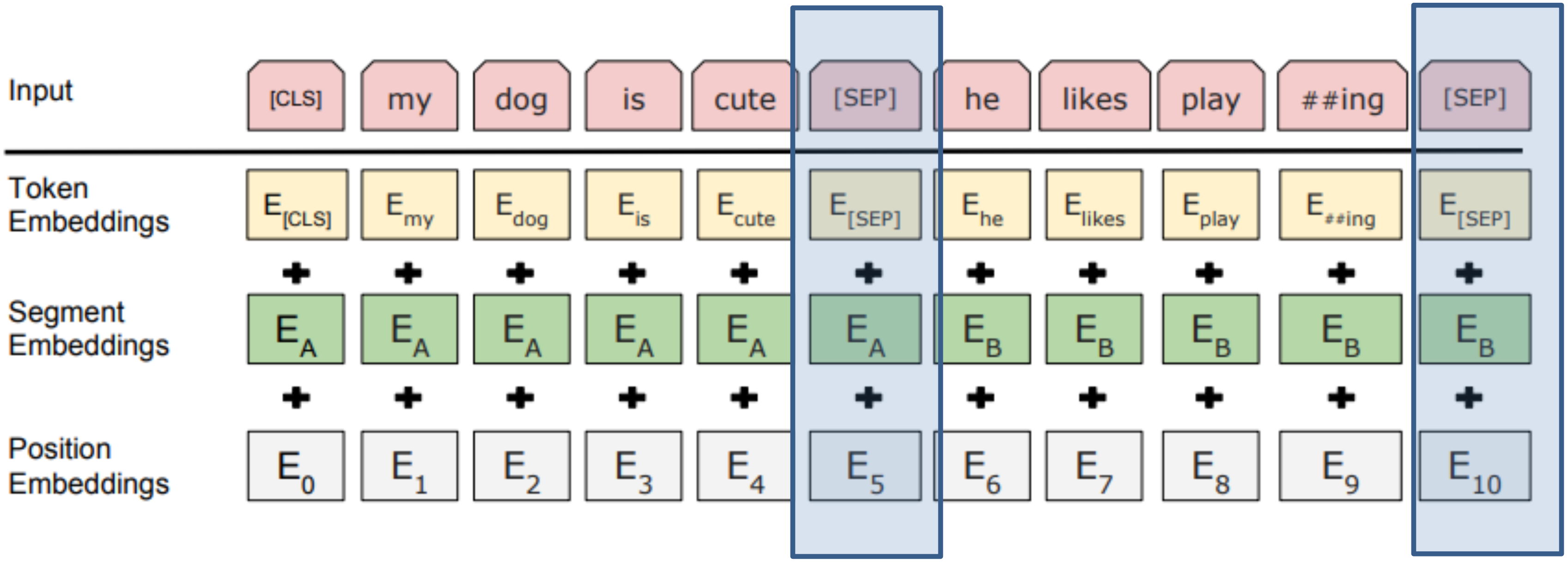
BERT Representation (Even More)



1. A special [CLS] token should precede the entire input to BERT

Fig. 2

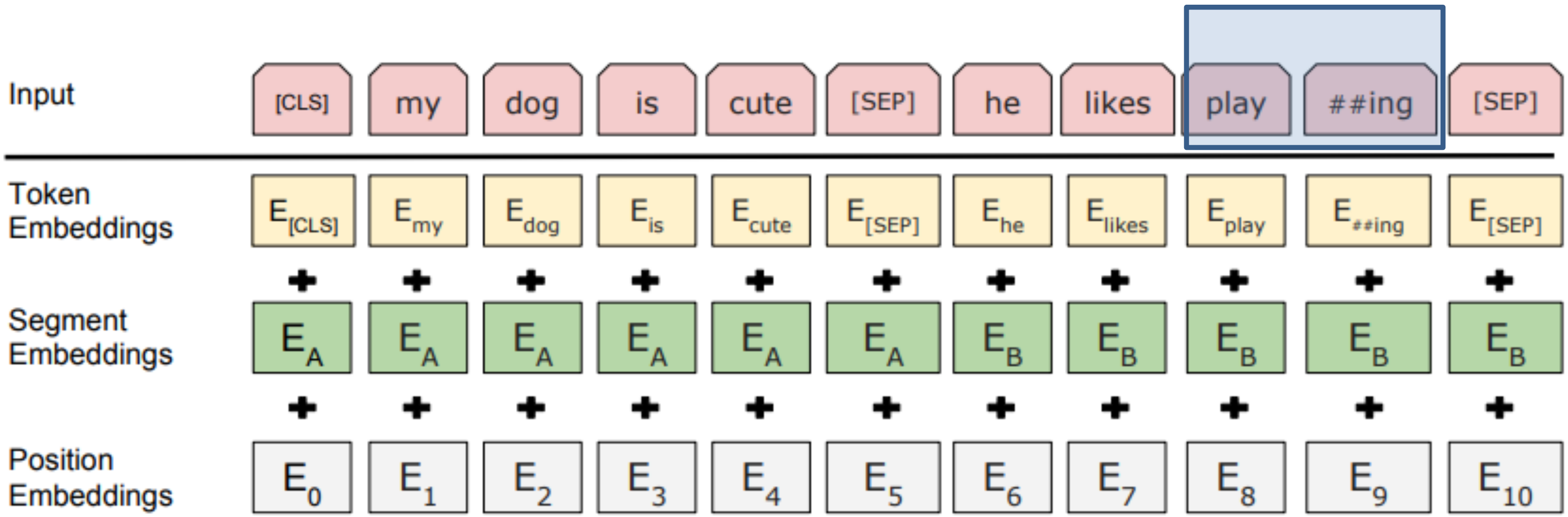
BERT Representation (Even More)



2. Every sentence should be followed by a special [SEP] token

Fig. 2

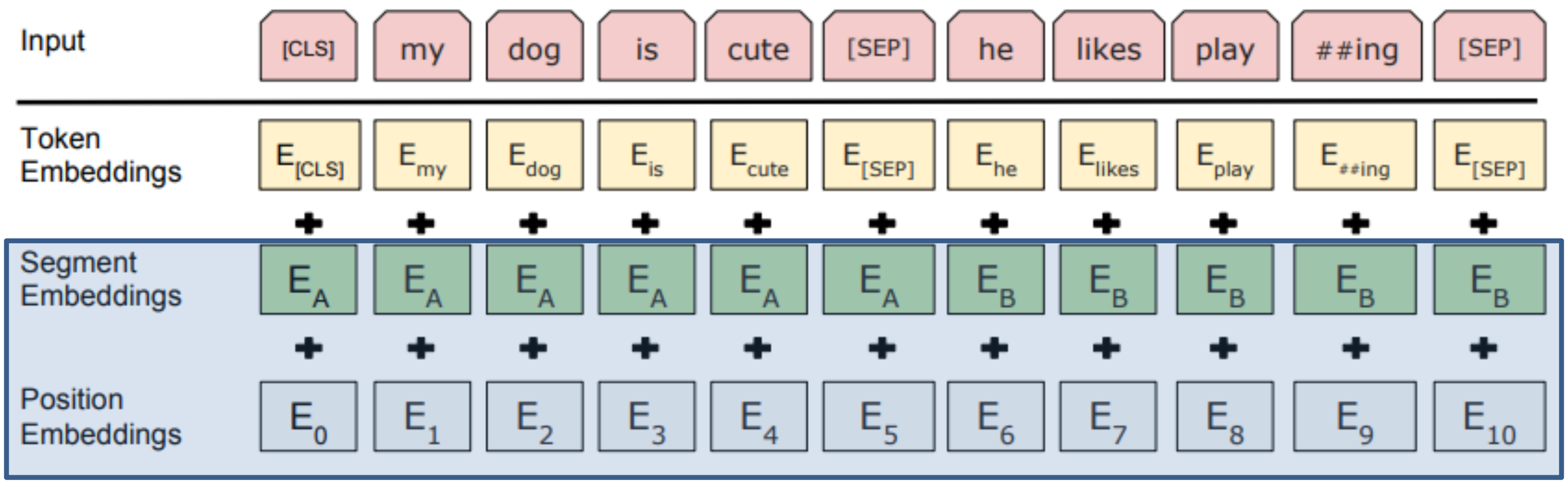
BERT Representation (Even More)



3. The input must be tokenized in a special way

Fig. 2

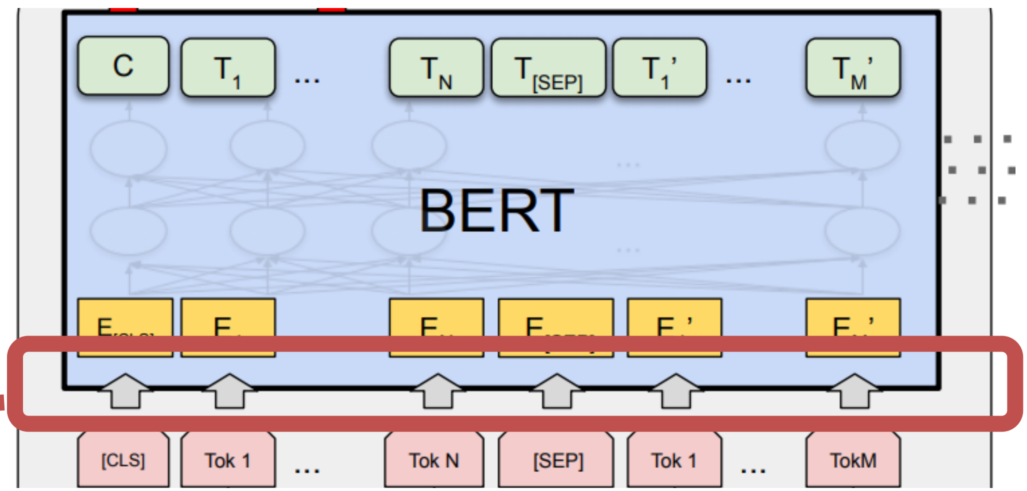
BERT Representation (Even More)



4. Segment & *position* embeddings must be provided

Fig. 2

BERT Representation (Even More)



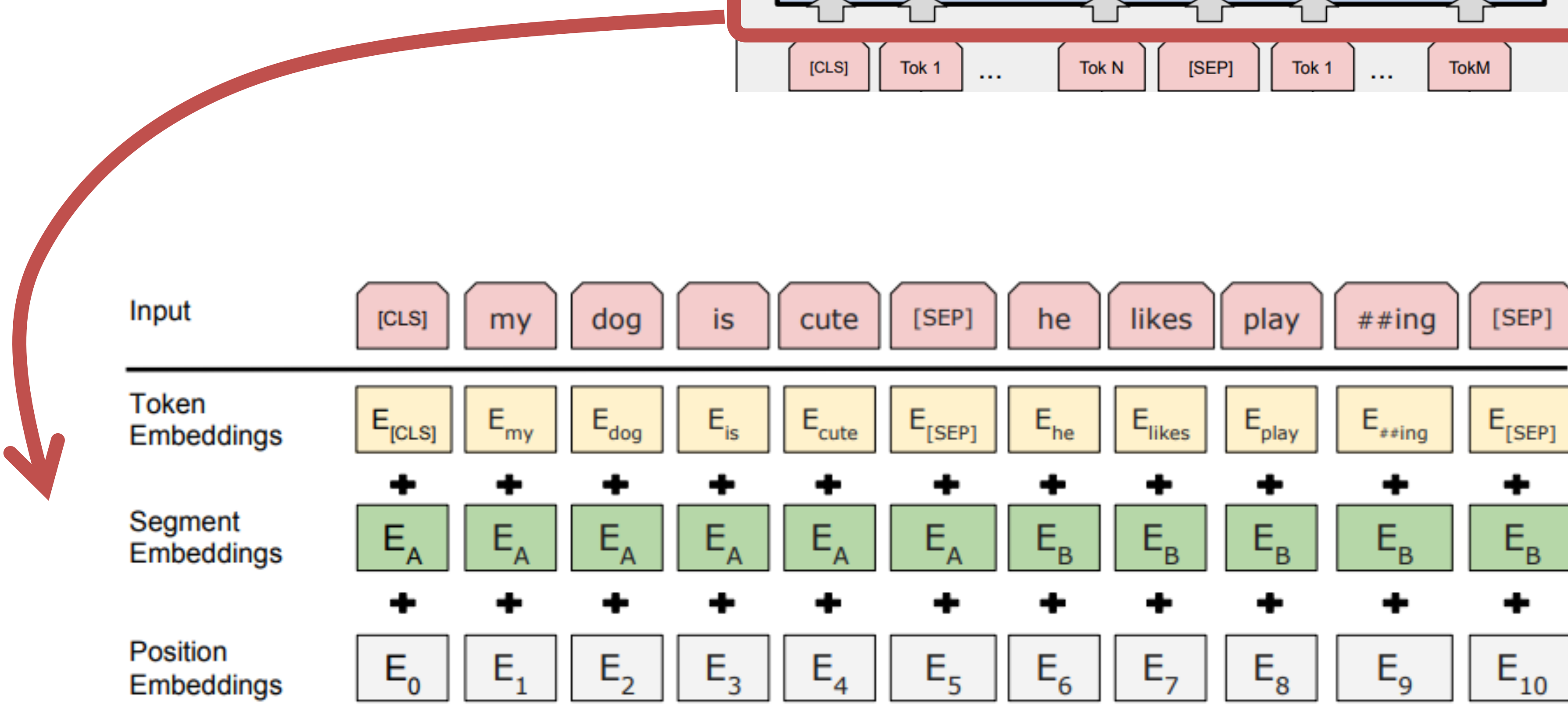


Fig. 2

Transformer Language Model Take-Aways

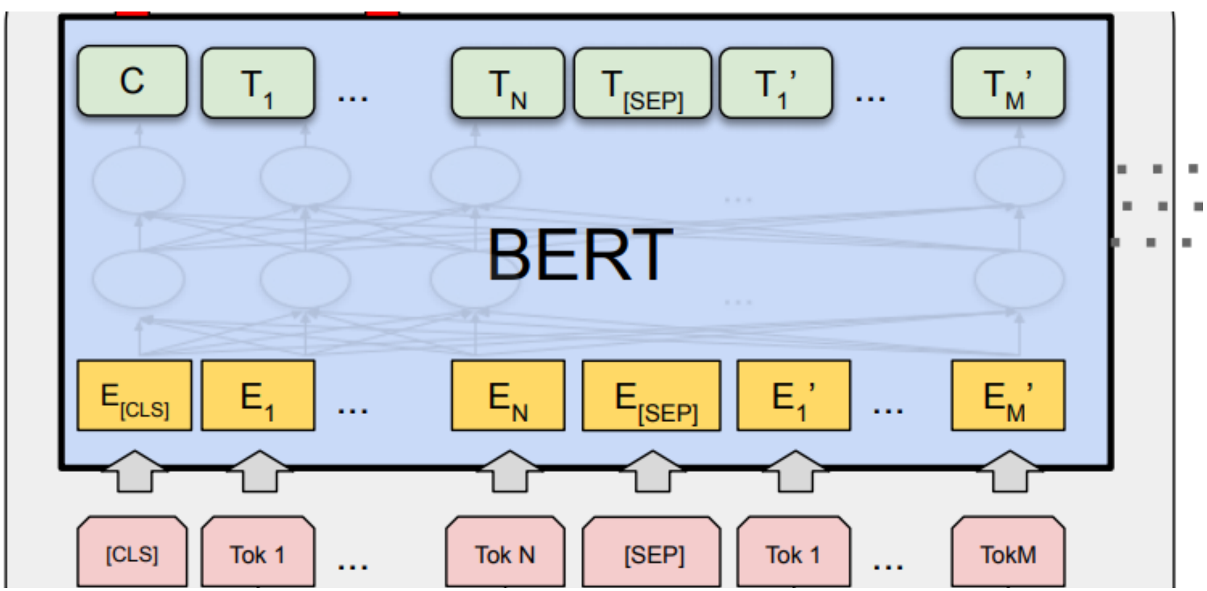
1. Clean separation of “pre-training” and “fine-tuning” tasks

2. Clear demonstration that language model “pre-training” can yield useful embeddings

BERTFor<X>

		Heads
Name	Input	Output
Language Modeling	$x_{1:n-1}$	$x_n \in \mathcal{V}$
Sequence Classification	$x_{1:N}$	$y \in \mathcal{C}$
Question Answering	$x_{1:M}, x_{M:N}$	y span $[1:N]$
Token Classification	$x_{1:N}$	$y_{1:N} \in \mathcal{C}^N$
Multiple Choice	$x_{1:N}, \mathcal{X}$	$y \in \mathcal{X}$
Masked LM	$x_{1:N \backslash n}$	$x_n \in \mathcal{V}$
[redacted]		

Fig. 2 (Wolf et al., 2020: https://arxiv.org/pdf/1910.03771.pdf)



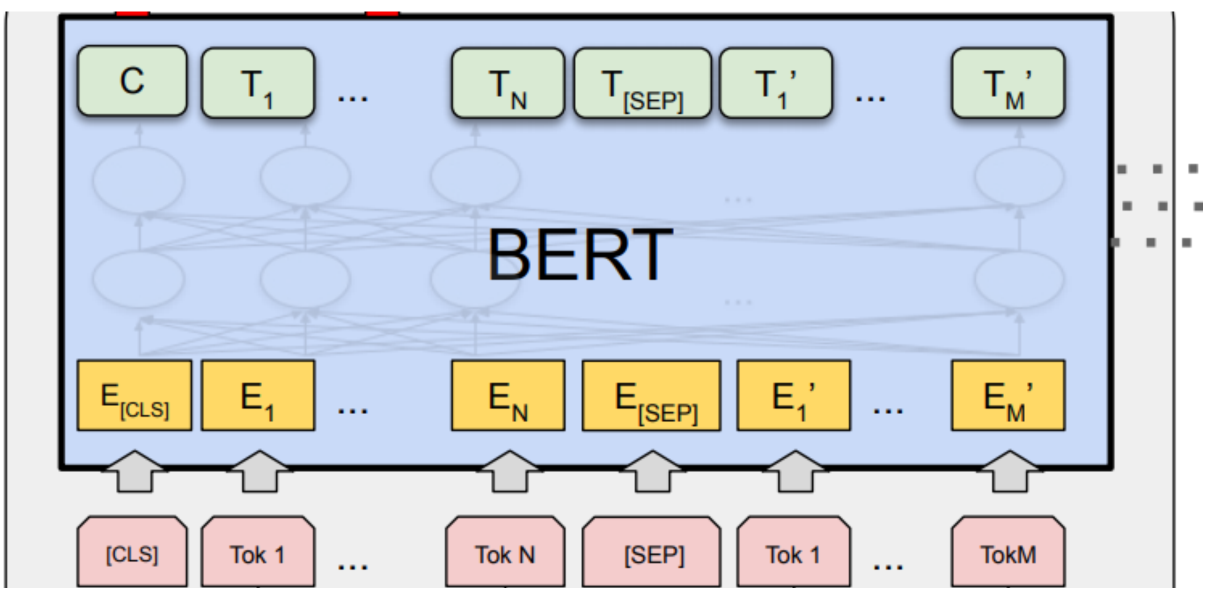
BERTFor<X>

Name	Input	Heads Output
Language Modeling	$x_{1:n-1}$	$x_n \in \mathcal{V}$
Sequence Classification	$x_{1:N}$	$y \in \mathcal{C}$
Question Answering	$x_{1:M}, x_{M:N}$	y span $[1:N]$
Token Classification	$x_{1:N}$	$y_{1:N} \in \mathcal{C}^N$
Multiple Choice	$x_{1:N}, \mathcal{X}$	$y \in \mathcal{X}$
Masked LM	$x_{1:N \setminus n}$	$x_n \in \mathcal{V}$
[redacted]		

Fig. 2 (Wolf et al., 2020: https://arxiv.org/pdf/1910.03771.pdf)

Both SequenceClassification and TokenClassification need some form of a *classifier*.

Q: How do we do (compute, represent) that?



BERTFor<X>

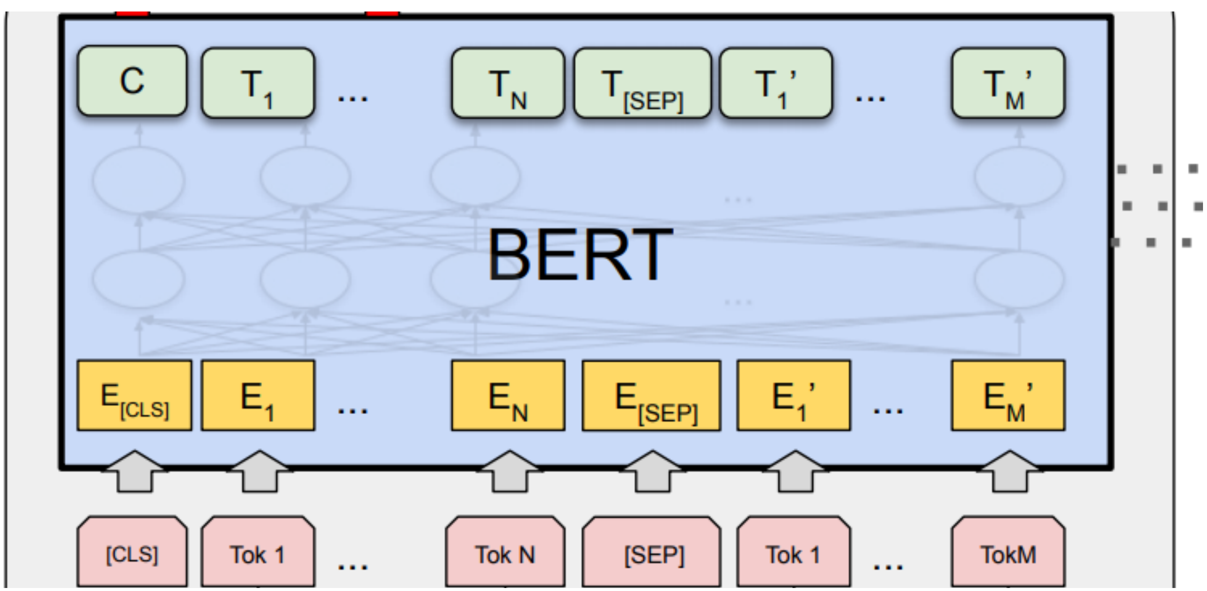
		Heads
Name	Input	Output
Language Modeling	$x_{1:n-1}$	$x_n \in \mathcal{V}$
Sequence Classification	$x_{1:N}$	$y \in \mathcal{C}$
Question Answering	$x_{1:M}, x_{M:N}$	y span $[1:N]$
Token Classification	$x_{1:N}$	$y_{1:N} \in \mathcal{C}^N$
Multiple Choice	$x_{1:N}, \mathcal{X}$	$y \in \mathcal{X}$
Masked LM	$x_{1:N \backslash n}$	$x_n \in \mathcal{V}$
[redacted]		

Fig. 2 (Wolf et al., 2020: https://arxiv.org/pdf/1910.03771.pdf)

Both SequenceClassification and TokenClassification need some form of a *classifier*.

Q: How do we do (compute, represent) that?

A: Linear layer (+ cross-entropy loss, and conceptually softmax)



		Heads
Name	Input	Output
Language Modeling	$x_{1:n-1}$	$x_n \in \mathcal{V}$
Sequence Classification	$x_{1:N}$	$y \in \mathcal{C}$
Question Answering	$x_{1:M}, x_{M:N}$	y span $[1:N]$
Token Classification	$x_{1:N}$	$y_{1:N} \in \mathcal{C}^N$
Multiple Choice	$x_{1:N}, \mathcal{X}$	$y \in \mathcal{X}$
Masked LM	$x_{1:N\setminus n}$	$x_n \in \mathcal{V}$
[redacted]		

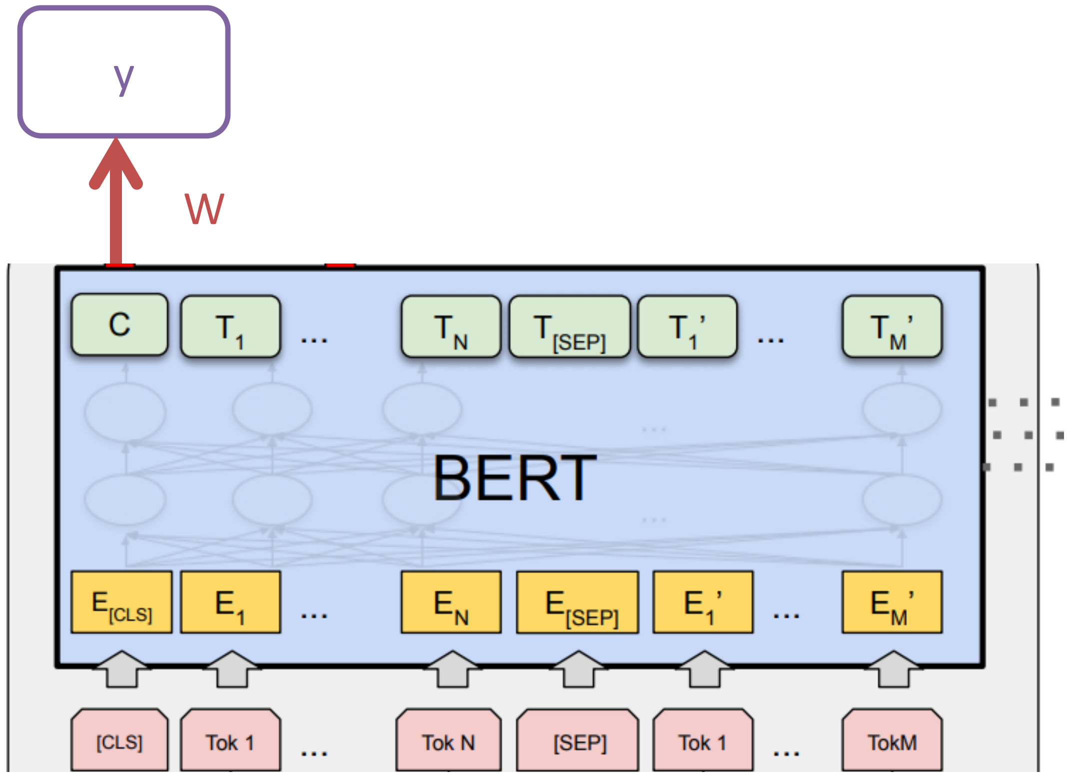
Fig. 2 (Wolf et al., 2020: https://arxiv.org/pdf/1910.03771.pdf)

BERTForSequenceClassification

Both SequenceClassification and TokenClassification need some form of a *classifier*.

Q: How do we do (compute, represent) that?

A: Linear layer (+ cross-entropy loss, and conceptually softmax)



Name	Input	Heads Output
Language Modeling	$x_{1:n-1}$	$x_n \in \mathcal{V}$
Sequence Classification	$x_{1:N}$	$y \in \mathcal{C}$
Question Answering	$x_{1:M}, x_{M:N}$	y span $[1:N]$
Token Classification	$x_{1:N}$	$y_{1:N} \in \mathcal{C}^N$
Multiple Choice	$x_{1:N}, \mathcal{X}$	$y \in \mathcal{X}$
Masked LM	$x_{1:N \setminus n}$	$x_n \in \mathcal{V}$
[redacted]		

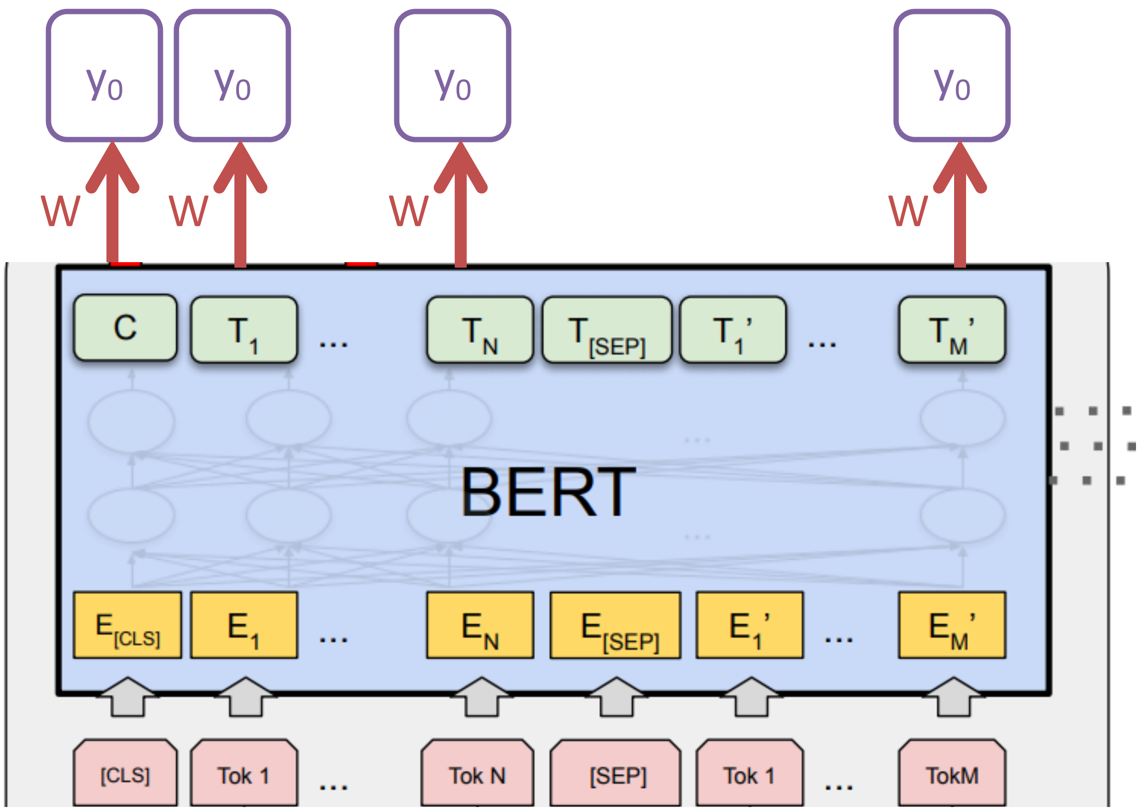
Fig. 2 (Wolf et al., 2020: https://arxiv.org/pdf/1910.03771.pdf)

BERTForTokenClassification

Both SequenceClassification and TokenClassification need some form of a *classifier*.

Q: How do we do (compute, represent) that?

A: Linear layer (+ cross-entropy loss, and conceptually softmax)



(single linear layer, re-used across the tokens)

Encoder vs. Decoder

Encoding: use a modeling structure that is effective. This could be:

- a bag-of-words style, or
- an auto-regressive (left-to-right) encoder, or
- a bi-directional / auto-encoding encoder

Decoding: an *auto-regressive* (left-to-right) structure, e.g., process one item, then another, then another

Transformers	
Masked $[x_{1:N\backslash n} \Rightarrow x_n]$	
BERT	(Devlin et al., 2018)
RoBERTa	(Liu et al., 2019a)
Autoregressive $[x_{1:n-1} \Rightarrow x_n]$	
GPT / GPT-2	(Radford et al., 2019)
Trans-XL	(Dai et al., 2019)
XLNet	(Yang et al., 2019)
Seq-to-Seq $[\sim x_{1:N} \Rightarrow x_{1:N}]$	
BART	(Lewis et al., 2019)
T5	(Raffel et al., 2019)
Marian	(J.-Dowmunt et al., 2018)

Fig. 2 (Wolf et al., 2020: https://arxiv.org/pdf/1910.03771.pdf)

Outline

Transformer Language Models as General Language Encoders

The Attention Mechanism

Effective, but challenges remain

- Sequence Input, Label Output (w/ time delay)
- Sequence Input, Sequence Output (w/o time delay)
- Sequence Input, Sequence Output (w/ time delay)

Core effective idea:
Use the basic recurrent (autoregressive) structure to capture longer-range dependencies that enable us to map from Input to Output

Challenges:

- Key, salient portions of the Input can become "buried"
- Knowing what to pay attention to is difficult

Attention

A mechanism for signaling where in the input to focus (“attend to”) when producing some output

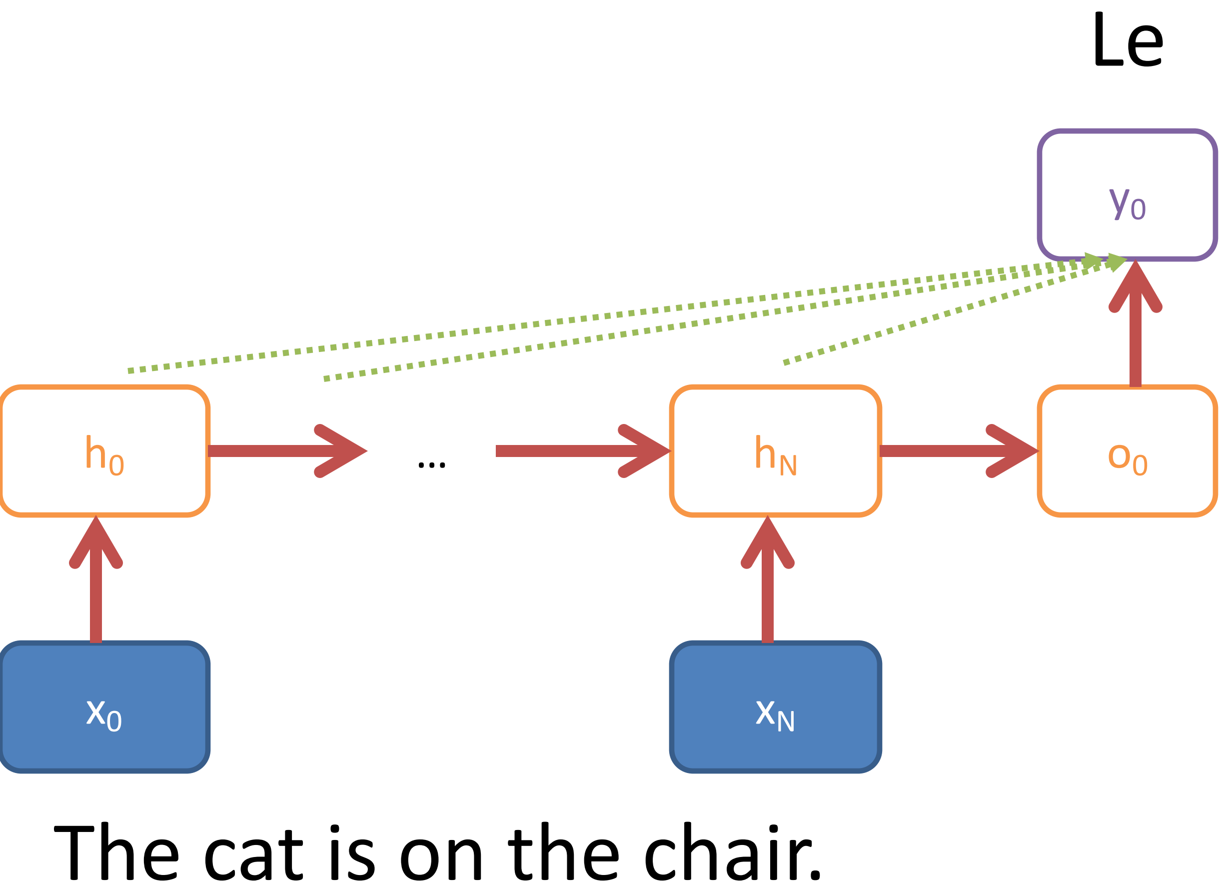
Each attention mechanism results in a probability distribution over the input

There are many ways of computing this

Attention results in learning how to form a “good” linear combination

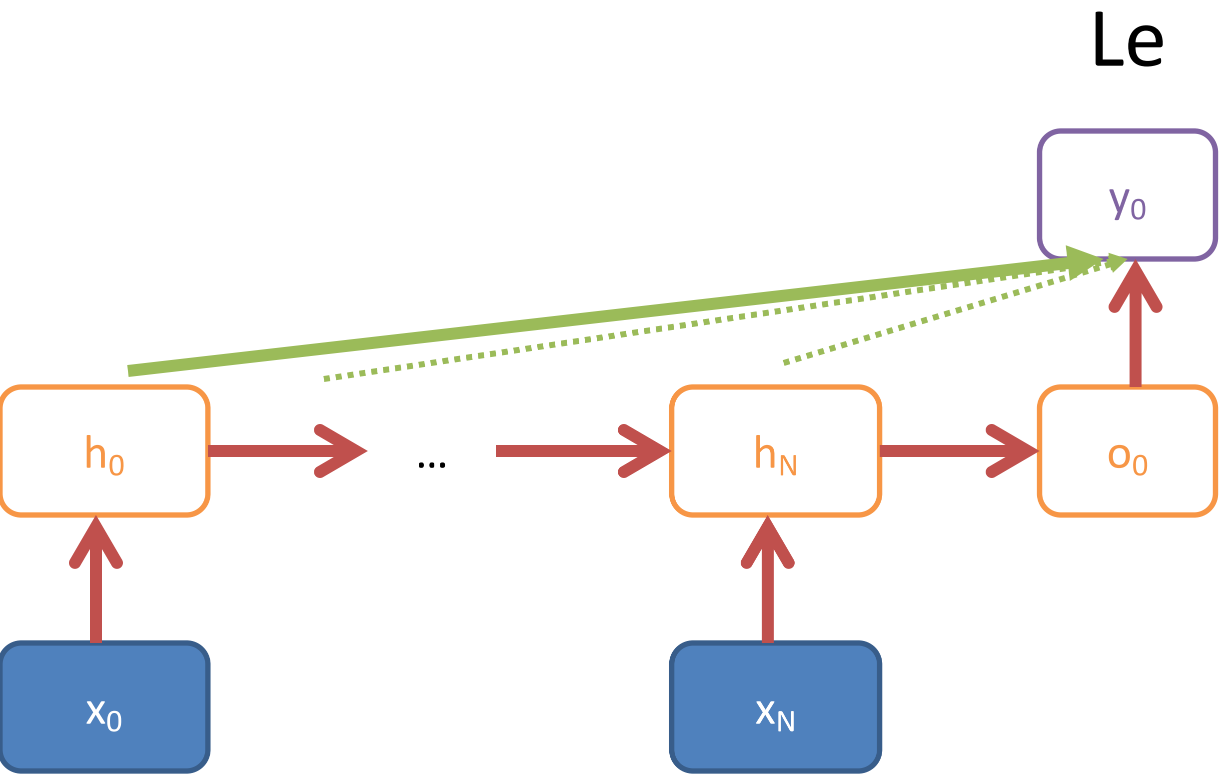
e.g., how to do a weighted average across a number of items

Example: Translation



Idea: generating the first word would be easier if we could look back to the input... but which word do we want to focus on?

Example: Translation

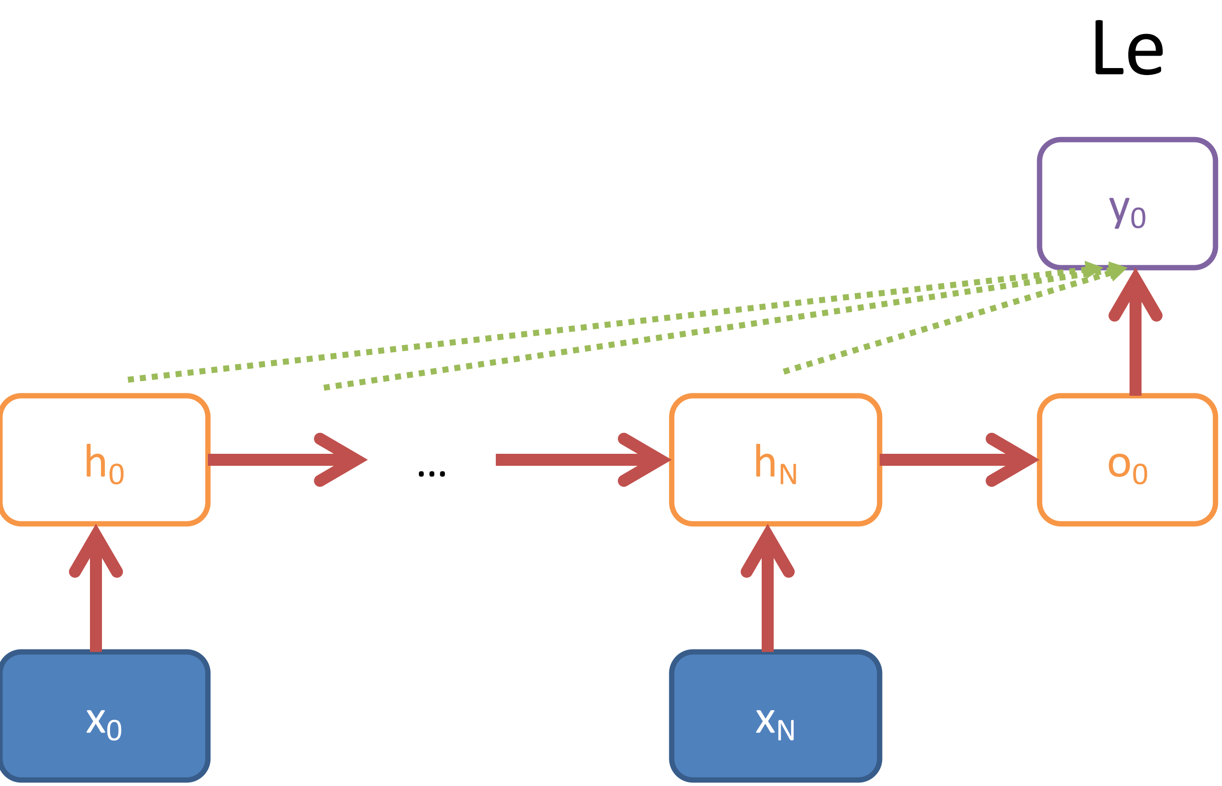


The cat is on the chair.

Idea: generating the first word would be easier if we could look back to the input... but which word do we want to focus on?

(Very high confidence in "The" would probably help produce "Le")

Solution: Attention



The cat is on the chair.

Idea: generating the first word would be easier if we could look back to the input... but which word do we want to focus on?

Attention: a learnable way of knowing what words to look back to

Attention

1. For a specific input x_i, **attention** computes a distribution α over K possible values $(z_1, z_2, \dots, z_K)$

2.

3.

Attention

1. For a specific input x_i, **attention** computes a distribution α over K possible values $(z_1, z_2, \dots, z_K)$

2. That distribution α is then used to linearly combine those K values together into a new representation

3.

Attention

1. For a specific input x_i, **attention** computes a distribution α over K possible values $(z_1, z_2, \dots, z_K)$

2. That distribution α is then used to linearly combine those K values together into a new representation

3. That representation is then used as features for classification (or input to another layer)

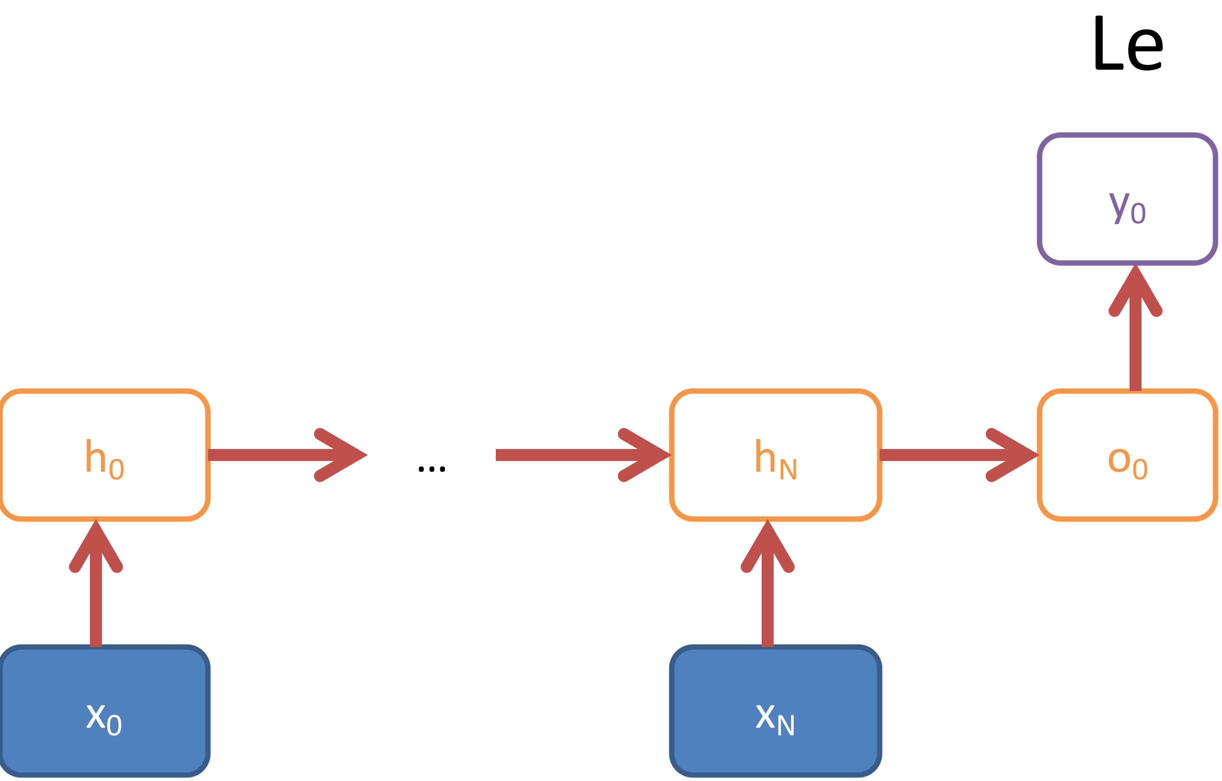
Attention

1. For a specific input x_i, **attention** computes a distribution α over K possible values $(z_1, z_2, \ldots, z_K)$
   – Often a form like
   $$\alpha(x_i)_k = \mathrm{softmax}(\mathrm{sim}(x_i, z_k))$$
2. That distribution α is then used to linearly combine those K values together into a new representation
3. That representation is then used as features for classification (or input to another layer)

Attention

1. For a specific input x_i, **attention** computes a distribution α over K possible values $(z_1, z_2, \ldots, z_K)$
   – Often a form like
$$\alpha(x_i)_k = \text{softmax}(\text{sim}(x_i, z_k))$$
2. That distribution α is then used to linearly combine those K values together into a new representation
   – Often a form like
$$u_i = \sum_{k=1}^{K} \alpha_k z_k$$
3. That representation is then used as features for classification (or input to another layer)

Solution: Attention

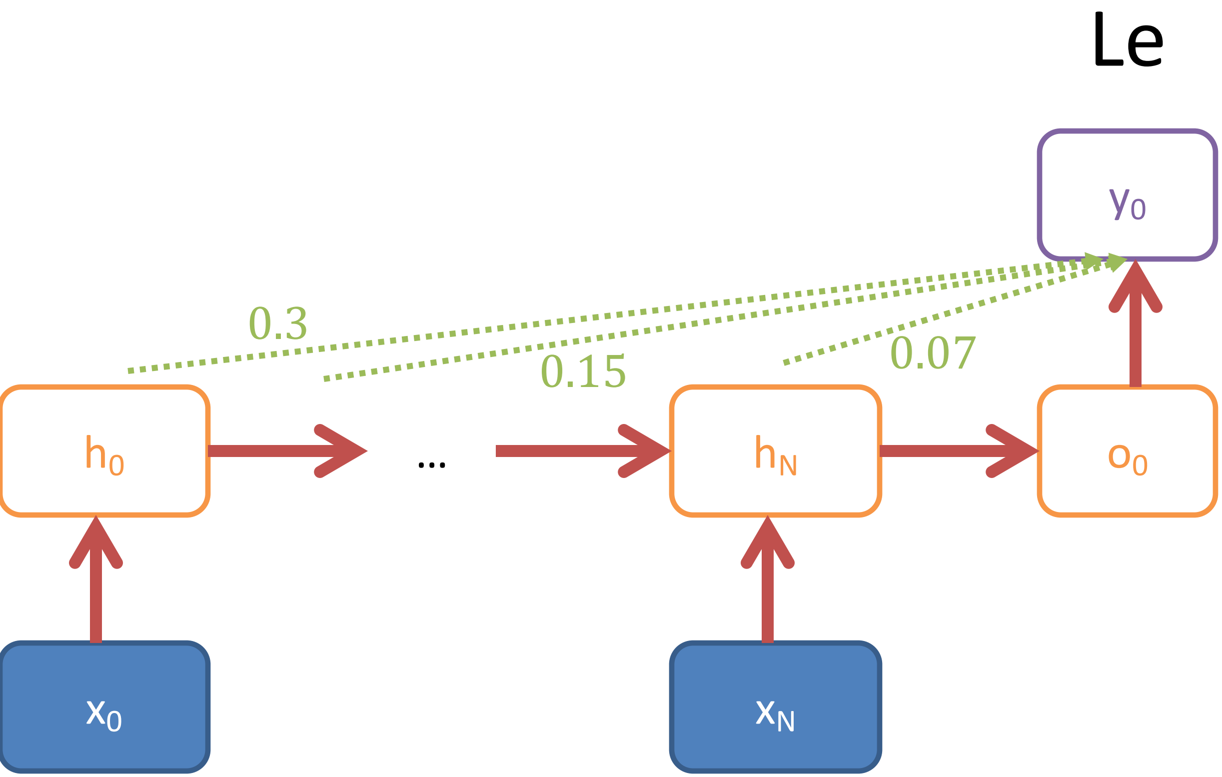


The cat is on the chair.

For a specific input o_0, **attention** computes a distribution α over (N+1) possible values $(h_0, h_1, \dots, h_N)$

$$\alpha(o_0)_k = \text{softmax}(\text{sim}(o_0, h_k))$$

Solution: Attention



The cat is on the chair.

For a specific input o_0, **attention** computes a distribution α over (N+1) possible values $(h_0, h_1, \dots, h_N)$

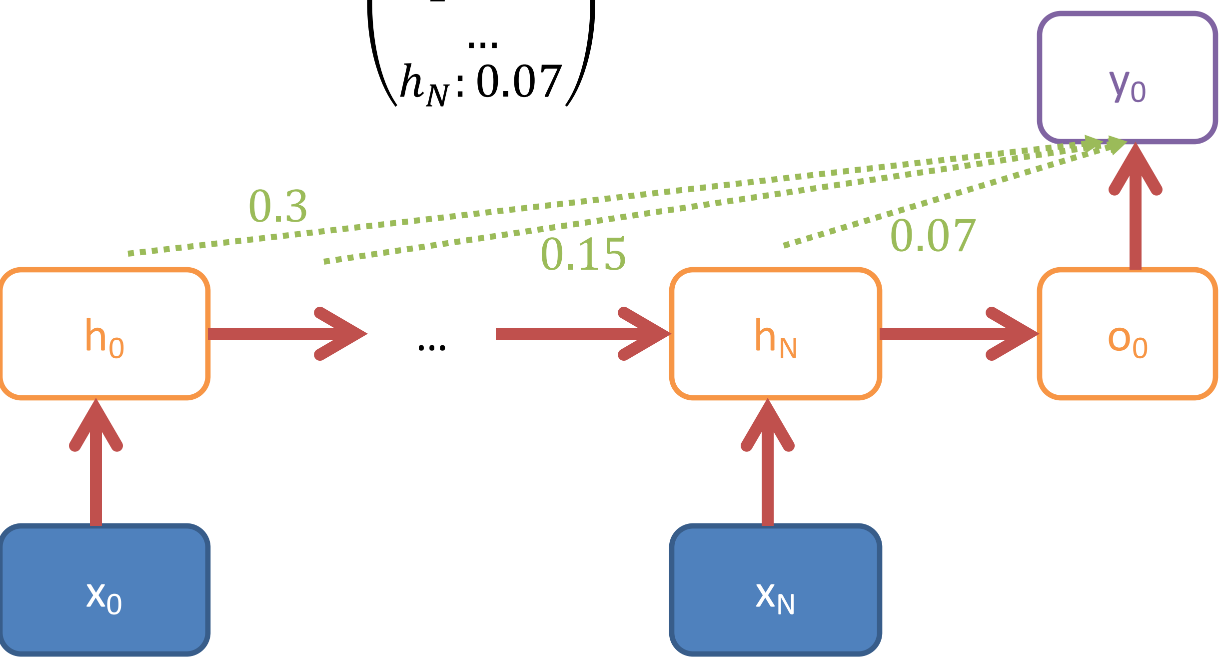
$$\alpha(o_0)_k = \text{softmax}(\text{sim}(o_0, h_k))$$

$$\alpha(o_0) = \begin{pmatrix} h_0: 0.3 \\ h_1: 0.15 \\ h_2: 0.07 \\ \dots \\ h_N: 0.07 \end{pmatrix}$$

Solution: Attention

$$\alpha(o_0) = \begin{pmatrix} h_0 : 0.3 \\ h_1 : 0.15 \\ h_2 : 0.07 \\ \dots \\ h_N : 0.07 \end{pmatrix}$$

Le



The cat is on the chair.

For a specific input o_0, **attention** computes a distribution α over (N+1) possible values $(h_0, h_1, \dots, h_N)$

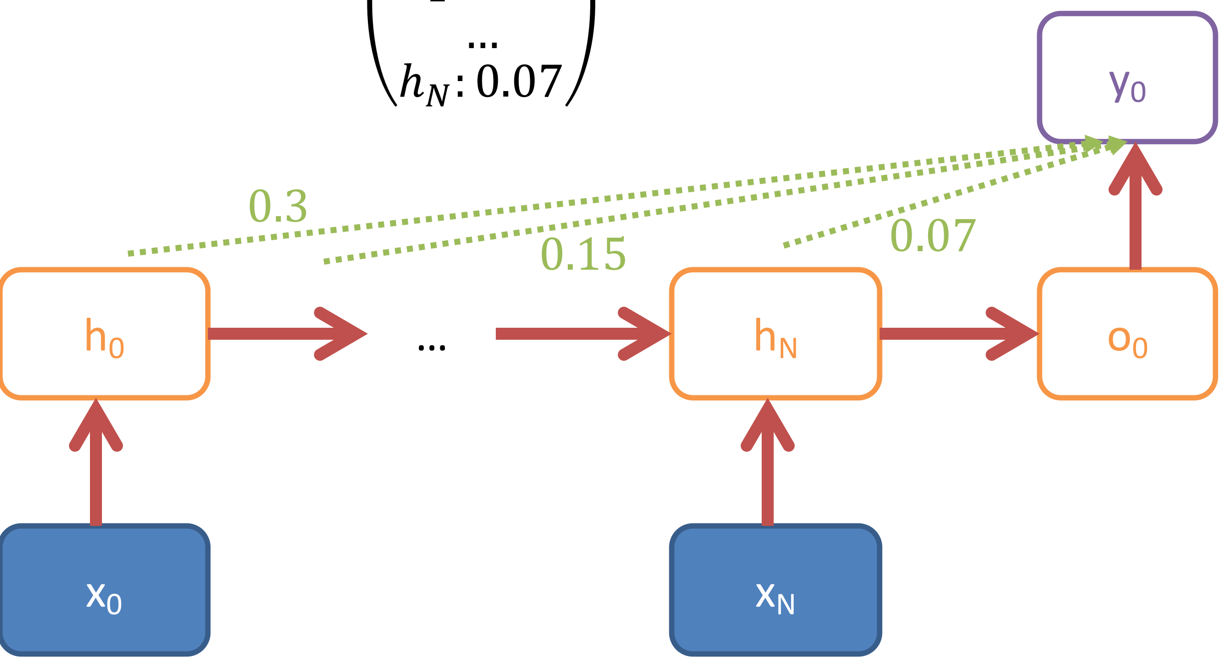
Use α to linearly combine those (N+1) values together into a new representation

$$u_i = \sum_{k=0}^{N+1} \alpha_k h_k$$

Solution: Attention

$$\alpha(o_0) = \begin{pmatrix} h_0: 0.3 \\ h_1: 0.15 \\ h_2: 0.07 \\ \dots \\ h_N: 0.07 \end{pmatrix}$$

Le



The cat is on the chair.

For a specific input o_0, **attention** computes a distribution α over (N+1) possible values $(h_0, h_1, \dots, h_N)$

Use α to linearly combine those (N+1) values together into a new representation

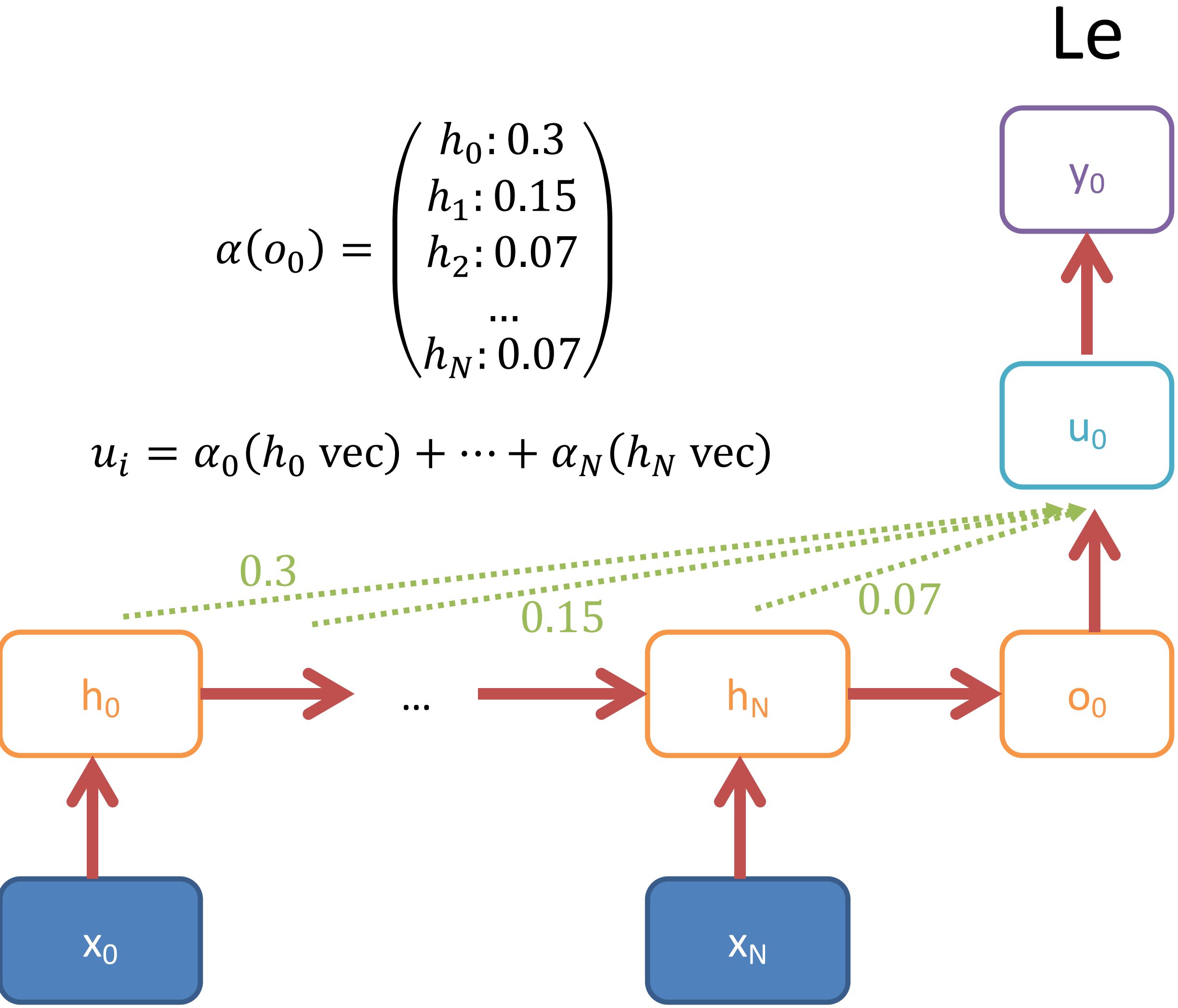
$$u_i = \sum_{k=0}^{N+1} \alpha_k h_k$$

$$u_i = \alpha_0 (h_0 \text{ vec}) + \cdots + \alpha_N (h_N \text{ vec})$$

Solution: Attention

$$\alpha(o_0) = \begin{pmatrix} h_0: 0.3 \\ h_1: 0.15 \\ h_2: 0.07 \\ \dots \\ h_N: 0.07 \end{pmatrix}$$

$$u_i = \alpha_0(h_0 \text{ vec}) + \cdots + \alpha_N(h_N \text{ vec})$$



For a specific input o_0, **attention** computes a distribution α over (N+1) possible values $(h_0, h_1, \dots, h_N)$

Use α to linearly combine those (N+1) values together into a new representation

Use u_0 as features for classification

The cat is on the chair.

Vaswani et al. (NeurIPS, 2017)

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

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.0 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature.

“Attention Is All You Need”: Take-Aways

1. Formulation of attention as a query-key-value triple

2. “Transformer” model that uses self-attention

3. Demonstration that the transformer can outperform sequence-to-sequence recurrent models (but at a large computational cost!)

“Attention Is All You Need” Description of Attention

“An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors.

*these query, key, value, and output items will be **task dependent***

"Attention Is All You Need" Description of Attention

"An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors.

The output is computed as a weighted sum of the values,

“Attention Is All You Need” Description of Attention

“An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors.

The output is computed as a weighted sum of the values,

where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.”

“Attention Is All You Need” Description of Attention

“An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors.

query: input & current translation

key: English words

value: French words

output: next translated word

Le ◊ chat

[Le] [chat] … [fromage] …

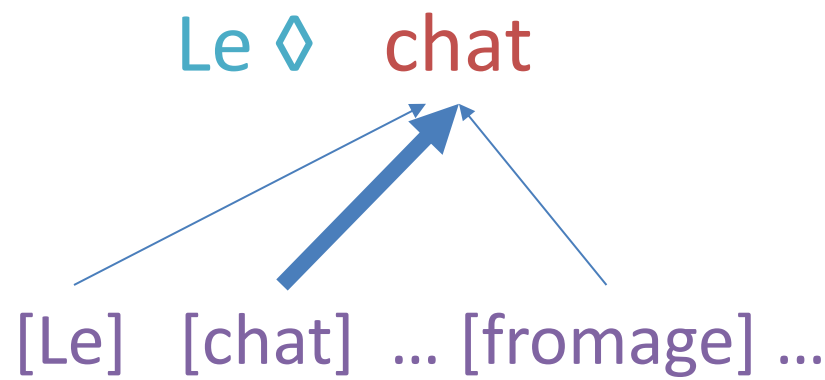
[The] [cat] … [bandage] …

The cat is on the chair.

“Attention Is All You Need” Description of Attention

“An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors.

The output is computed as a weighted sum of the values,



[The] [cat] … [bandage] …

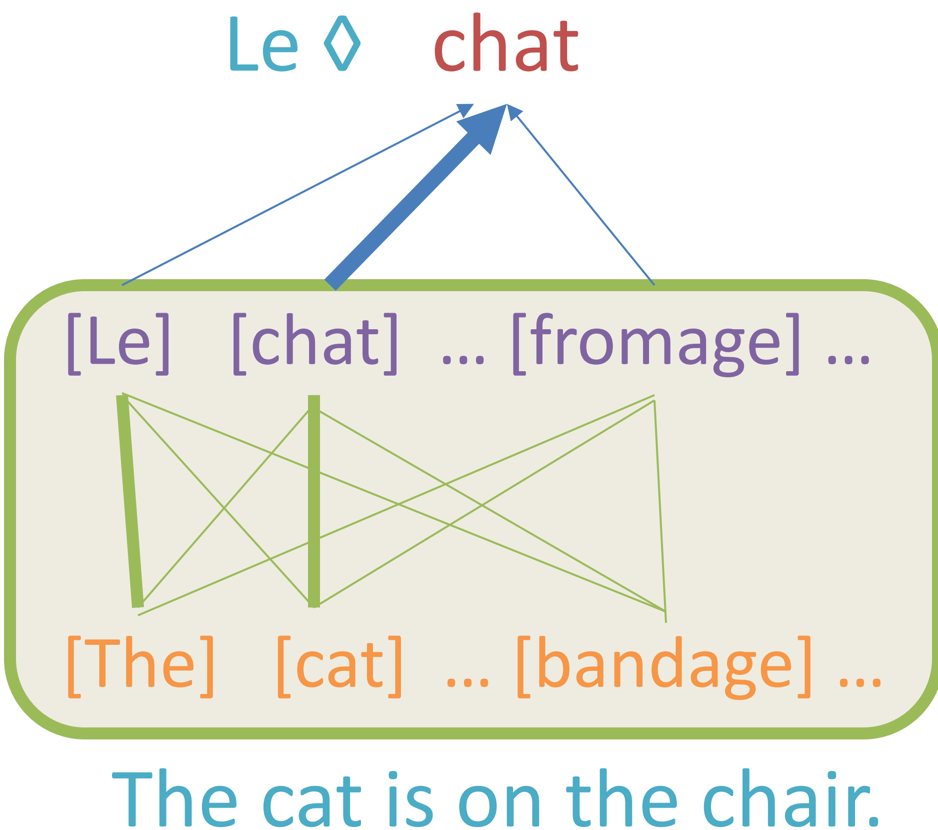
The cat is on the chair.

“Attention Is All You Need” Description of Attention

“An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors.

The output is computed as a weighted sum of the values,

where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.”



"Attention Is All You Need" Description of Attention (advanced)

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\underbrace{\frac{1}{d_k} QK^T}_{\text{similarity function (}d_k\text{ is embedding dim.)}}\right) V$$

“Attention Is All You Need” Description of Attention (advanced)

$$\text{Attention}(Q,K,V) = \overbrace{\text{softmax}\left(\underbrace{\frac{1}{d_k}QK^T}_{\text{similarity function (d_k is embedding dim.)}}\right)}^{\text{attending distribution α from before}} V$$

“Attention Is All You Need” Description of Attention (advanced)

$$\text{Attention}(Q, K, V) = \underbrace{\overbrace{\text{softmax}\left(\underbrace{\frac{1}{d_k} Q K^T}_{\text{similarity function (d_k is embedding dim.)}}\right)}^{\text{attending distribution α from before}} V}_{\text{linear combination over outputs}}$$

Multi-head Attention

MULTIHEADATTENTION

CLASS torch.nn.MultiheadAttention(*embed_dim*, *num_heads*, *dropout=0.0*, *bias=True*, *add_bias_kv=False*, *add_zero_attn=False*, *kdim=None*, *vdim=None*, *batch_first=False*, *device=None*, *dtype=None*) [SOURCE]

Allows the model to jointly attend to information from different representation subspaces as described in the paper: Attention Is All You Need.

Multi-Head Attention is defined as:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(head_1, \ldots, head_h)W^O$$

where $head_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$.

Multi-head Attention

MULTIHEADATTENTION

CLASS `torch.nn.MultiheadAttention(embed_dim, num_heads, dropout=0.0, bias=True, add_bias_kv=False, add_zero_attn=False, kdim=None, vdim=None, batch_first=False, device=None, dtype=None)` [SOURCE]

Allows the model to jointly attend to information from different representation subspaces as described in the paper: Attention Is All You Need.

Multi-Head Attention is defined as:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(head_1, \ldots, head_h)W^O$$

where $head_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{1}{d_k} Q K^T\right) V$$

Multi-head Attention

MULTIHEADATTENTION

CLASS torch.nn.MultiheadAttention(*embed_dim*, *num_heads*, *dropout=0.0*, *bias=True*, *add_bias_kv=False*, *add_zero_attn=False*, *kdim=None*, *vdim=None*, *batch_first=False*, *device=None*, *dtype=None*) [SOURCE]

Allows the model to jointly attend to information from different representation subspaces as described in the paper: Attention Is All You Need.

Multi-Head Attention is defined as:

$$\mathrm{MultiHead}(Q,K,V) = \mathrm{Concat}(head_1,\dots,head_h)W^O$$

where $head_i = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$.

$$\mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

$$= \mathrm{softmax}\left(\frac{1}{d_k}(QW_i^Q)(KW_i^K)\right)VW_i^V$$

Outline

Transformer Language Models as General Language Encoders

The Attention Mechanism

Key Highlights

- While there are a number of different types of networks, it's helpful to think of them as *encoding* (learning to featurize) the input, and then making an appropriate prediction ("decode")
- This *encoding* is driven by learning what is effective for language modeling
- This *decoding* can be "prediction" or "language modeling"
- *Attention* is the building block behind many of these approaches
- *Attention* learns how to perform linear combinations of embeddings

Single Input, Single Output

Single Input, Multiple Outputs

Multiple Inputs, Single Output

Multiple Inputs, Multiple Outputs ("sequence prediction": no time delay)

Multiple Inputs, Multiple Outputs ("sequence-to-sequence": with time delay)

	Heads			
Name	Input	Output	Tasks	Ex. Datasets
Language Modeling	$x_{1:n-1}$	$x_n \in \mathcal{V}$	Generation	WikiText-103
Sequence Classification	$x_{1:N}$	$y \in \mathcal{C}$	Classification, Sentiment Analysis	GLUE, SST, MNLI
Question Answering	$x_{1:M}, x_{M:N}$	y span $[1:N]$	QA, Reading Comprehension	SQuAD, Natural Questions
Token Classification	$x_{1:N}$	$y_{1:N} \in \mathcal{C}^N$	NER, Tagging	OntoNotes, WNUT
Multiple Choice	$x_{1:N}, \mathcal{X}$	$y \in \mathcal{X}$	Text Selection	SWAG, ARC
Masked LM	$x_{1:N \setminus n}$	$x_n \in \mathcal{V}$	Pretraining	Wikitext, C4
Conditional Generation	$x_{1:N}$	$y_{1:M} \in \mathcal{V}^M$	Translation, Summarization	WMT, IWSLT, CNN/DM, XSum