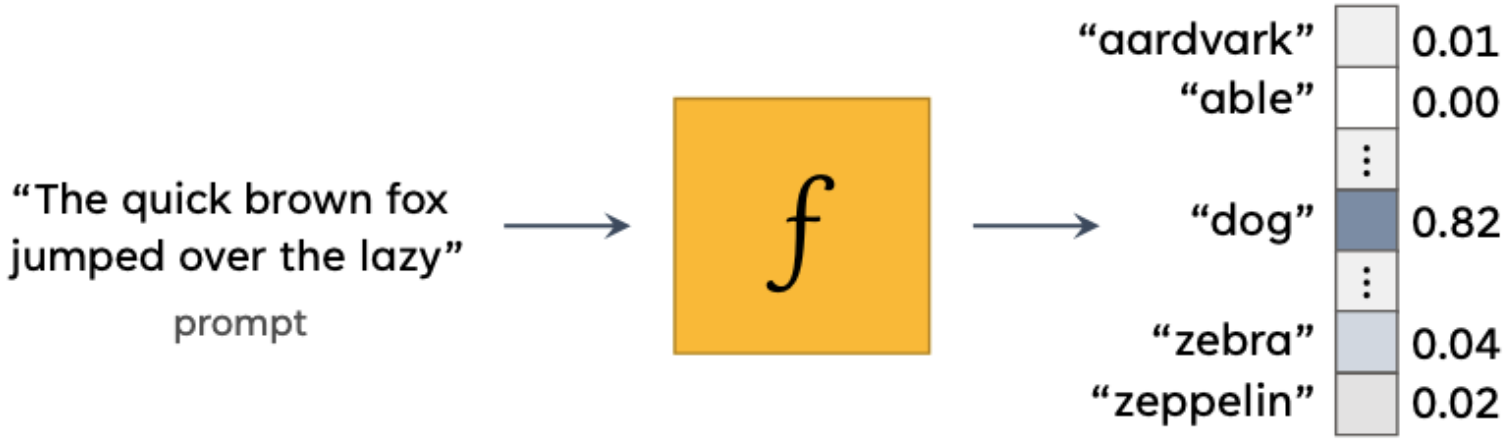
Neural Language Models

LANGUAGE MODELING

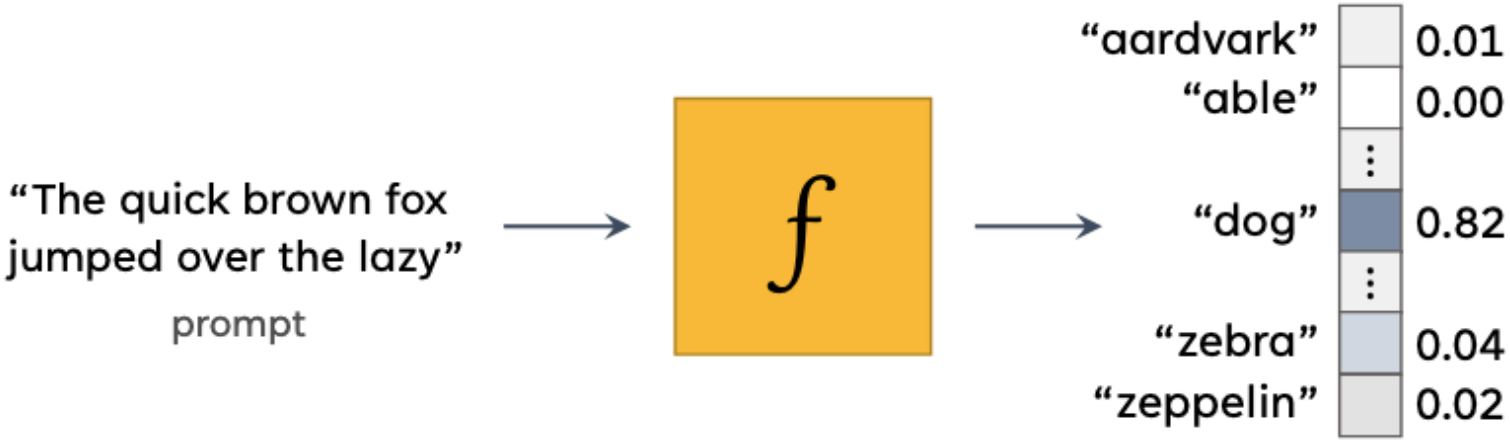
- In simple terms, it is the task of predicting the next word, given the previous *n* words.



- Language modeling is a multi-class classification task.
- Each word in the vocabulary is an output class.
- More precisely: if $x_1, x_2, \ldots, x_n$ is a sequence of words, $f(x_1, \ldots, x_n) = p(x_n \mid x_1, \ldots, x_{n-1})$
- Language is ambiguous, and so language modeling is a probabilistic task.

LANGUAGE MODELING

- What is f?
- It is a machine learning model, trained on many examples of input-output pairs.



- It is easy to find training data:
  - Take any n-word sequence of text (such as from the internet),
  - The first $n-1$ words are the input, and the last word is the output.
- Language modeling is unsupervised/self-supervised.
- But we must be mindful of the training data. There may be inaccuracies or noise.

LANGUAGE MODELING

- What is f?
- It is a machine learning model, trained on many examples of input-output pairs.

- We can use language models to assign probabilities of phrases, or sentences.

$$p(\text{'The quick fox'}) = p(\text{'The'})\ p(\text{'quick'}|\text{'The'})\ p(\text{'fox'}|\text{'The'},\text{'quick'})$$

- By the chain rule of probability theory:

$$p(x_1, \ldots, x_n) = p(x_n \mid x_1, \ldots, x_{n-1})\ p(x_1, \ldots, x_{n-1})$$
$$= p(x_1)\ p(x_2 \mid x_1)\ p(x_3 \mid x_1, x_2) \ldots p(x_n \mid x_1, \ldots, x_{n-1})$$

- We can use language models to compare the probabilities of different phrases/sentences of arbitrary length:
- p('The quick fox') > p('The quick turtle')
- p('The quick fox') > p('The quikc fox')

OTHER NLP TASKS AS LANGUAGE MODELING

- Many other NLP tasks can be "reduced" into language modeling:
- Spam detection
  - Prompt: "Email: Dear customer, … Question: Is this (a) spam, or (b) not spam? Answer:"
- Machine translation
  - Prompt: "Translate the following into Spanish: 'The quick brown fox…' Translation:"
- Sentiment analysis
  - Prompt: "Review: This product was not great… Is this review (a) positive, (b) negative, or (c) neutral? Answer:"
- Question answering
  - Prompt: "Question: Bob has 5 apples. Alice gave 10 apples to Bob. Alice now has 23 apples. How many apples did Alice start with? Answer:"

…

- Thus, if we can train a model to do well on language modeling, it may be able to perform well on many other NLP tasks.

UNIGRAM MODEL

- If we assume that each word is independent, we obtain a simple model.

$$p(x_n \mid x_1, \ldots, x_{n-1}) = p(x_n) \text{ for all } n$$

- Imagine putting all words of a large corpus in a bag, shuffling their order, and picking one at random.
- How can we estimate the probability of picking a specific word, say "cat"?
- Simple approach:
  - Count the number of times "cat" appears in your training data,
  - Then divide by the total number of words in the data.
- This is called a unigram model.

BIGRAM MODEL

- What are some shortcomings of this model?
- The strong independence assumption causes the model to throw away all word order information.
- What if we instead made a slightly weaker assumption:

$$p(x_n \mid x_1, \ldots, x_{n-1}) = p(x_n \mid x_{n-1}) \text{ for all } n$$

- Each word depends on only the previous word.
- We can extend the counting procedure from the unigram model:
  - For every pair of words in the vocabulary (w_1, w_2), count the number of times w_2 appears after w_1.
  - Then we can estimate: $p(w_n \mid w_{n-1}) = \dfrac{\#\text{ of times } w_n \text{ appeared after } w_{n-1} \text{ in the training set}}{\#\text{ of times } w_{n-1} \text{ appeared in the training set}}$

N-GRAM MODEL

- We can extend this to n-gram models, for arbitrary n.
- Each word depends on only the previous $n - 1$ words.
- We can extend the counting procedure:
  - For every sequence of n words in the vocabulary $(w_1, \ldots, w_n)$, count the number of times w_n appears after $(w_1, \ldots, w_{n-1})$.
  - Then we can estimate:

$$p(w_n \mid w_1, \ldots, w_{n-1}) = \frac{\text{\# of times } w_n \text{ appeared after } w_1, \ldots, w_{n-1} \text{ in the training set}}{\text{\# of times } w_1, \ldots, w_{n-1} \text{ appeared in the training set}}$$

Sampling from a language model

1 gram	–To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have
	–Hill he late speaks; or! a more to leg less first you enter
2 gram	–Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow.
	–What means, sir. I confess she? then all sorts, he is trim, captain.
3 gram	–Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done.
	–This shall forbid it should be branded, if renown made it empty.
4 gram	–King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in;
	–It cannot be but so.

Figure 3.4 Eight sentences randomly generated from four n-gram models computed from Shakespeare's works. All characters were mapped to lower-case and punctuation marks were treated as words. Output is hand-corrected for capitalization to improve readability.

https://web.stanford.edu/~jurafsky/slp3/3.pdf

Hmm..

write a fake quote that sounds like it comes from a Shakespeare's play.

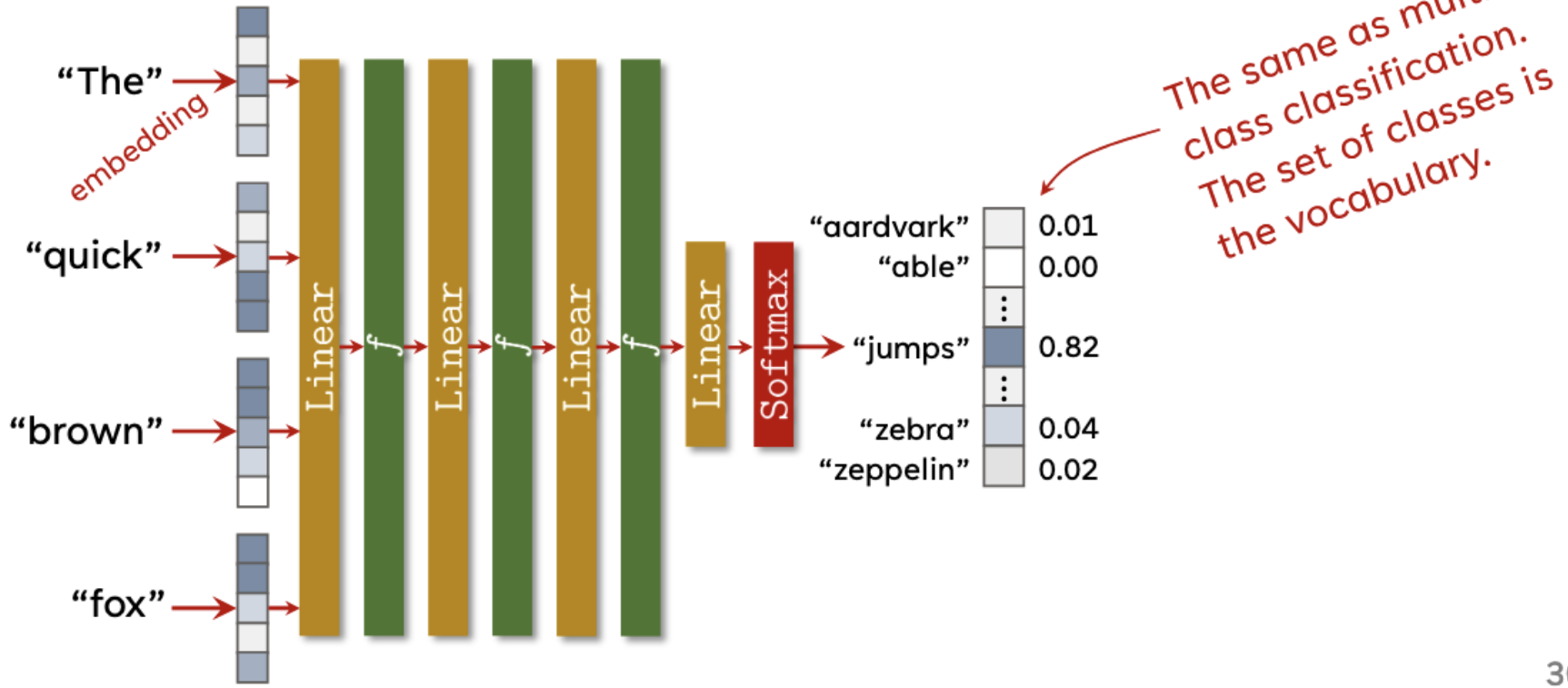
“When shadows whisper truth where daylight dares not tread,
I find thy absence louder than the clash of kings;
For in the hollow court of mine own breast,
Thy memory sits enthroned, and will not yield.”

Not just a numbers game

- Let's assume an English vocabulary of about 50K words.
- 4-gram language model: **6,250,000,000,000,000,000**
- Current LLMs **~ 1,000,000,000,000**
- **Several major factors**: better representation of words and their dependencies (neural vs. symbolic), neural architectures and learning "tricks", **a lot** of compute and training data

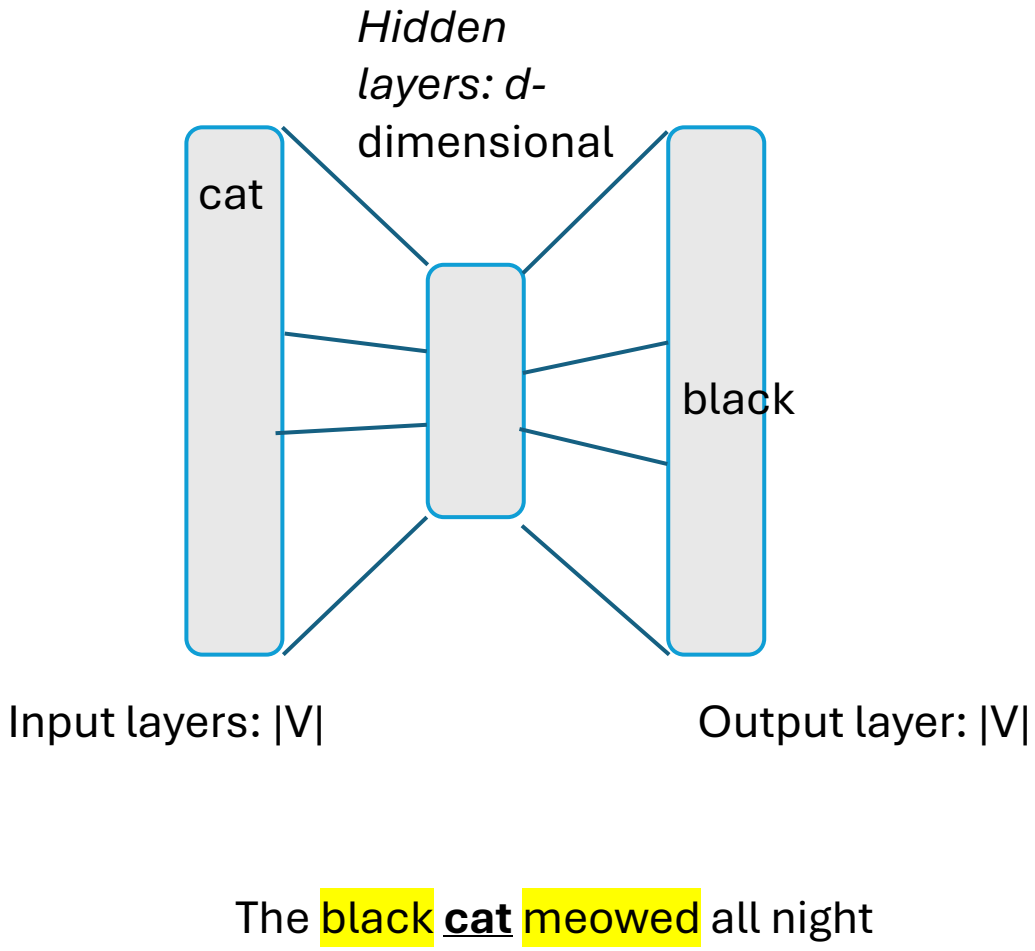
FEEDFORWARD LANGUAGE MODELS

- MLPs can be applied to perform language modeling.
- Predict the next word from the last layer's activations.



Word2Vec and Skip-Grams

- Predict context word, from a context window.
- Skip-gram sampling, from a forward and backward window of a fixed size.
- Result: d-dimensional embedding for each word in V.



The black **cat** meowed all night

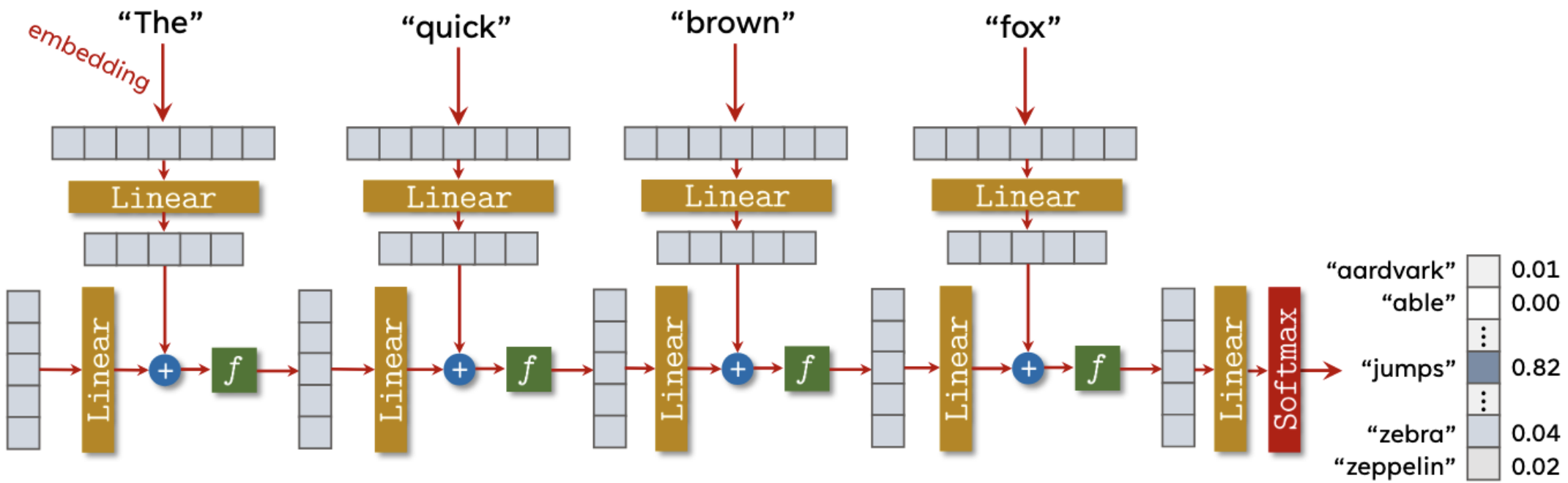
https://arxiv.org/abs/1301.3781

Word2Vec vs. Feedforward LMs

- The two models are very similar!
- **If we assume a bigram LM, what is the difference?**
  - FFLM: can only learn backward relationships (model $p(w_i|w_{i-1})$)
  - W2V: both forward and backwards relationships $p(w_i|w_{i-1})$ and $p(w_i|w_{i+1})$
  - W2V learns relationships between close though non-consecutive words (e.g., $p(w_i|w_{i-2})$) assuming the context window is larger than 1.
- These differences are also repeated in modern neural "LM"s :
  - Actual LM: only model relationship to previous tokens (i.e., causal/autoregressive LMs)
  - *Not-really-LMs*, modeling both previous and future tokens (bidirectional).

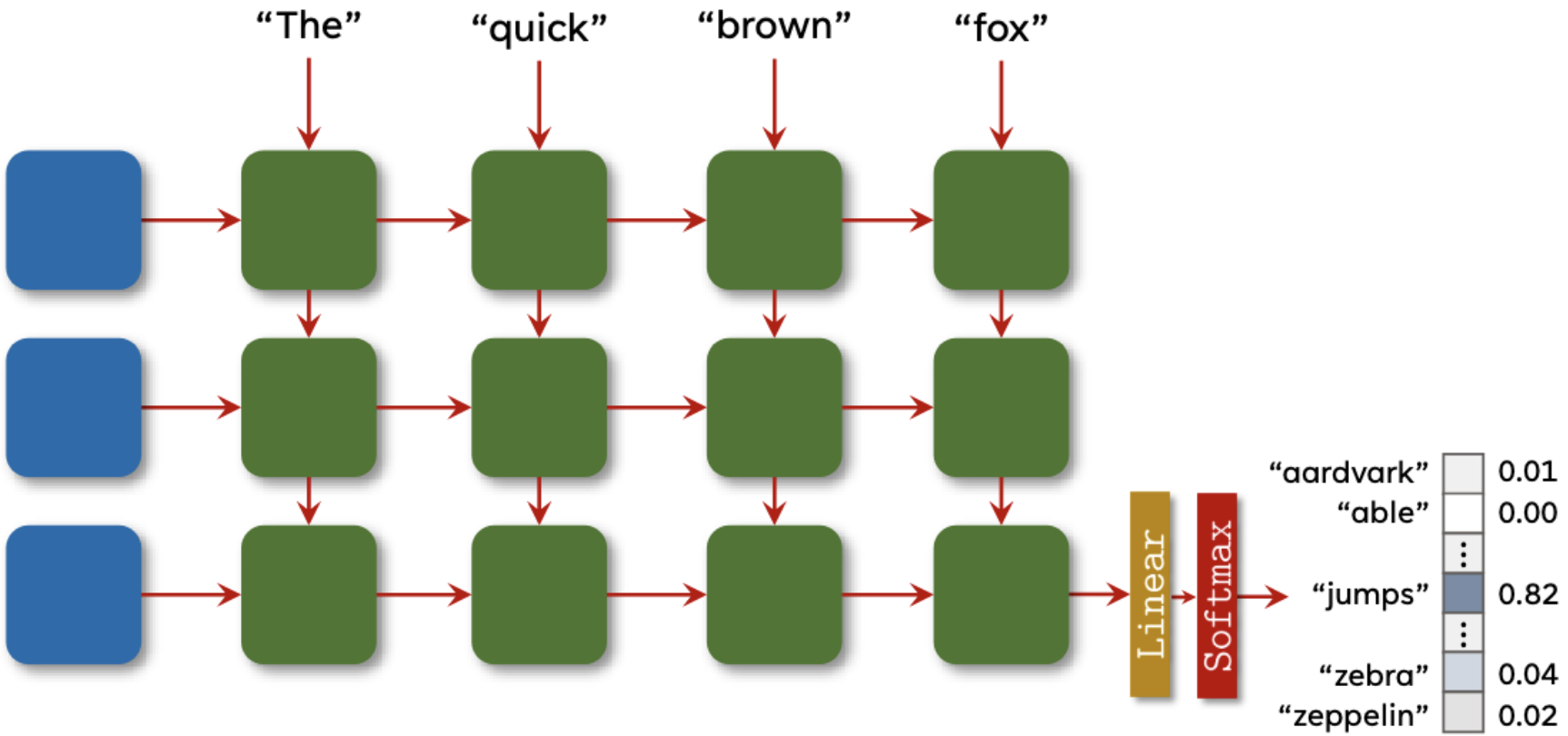
RNN LANGUAGE MODELS

- RNNs can be applied to perform language modeling.
- Each word/token of the input provides an update to the hidden state.
- Predict the next word from the last hidden state vector.



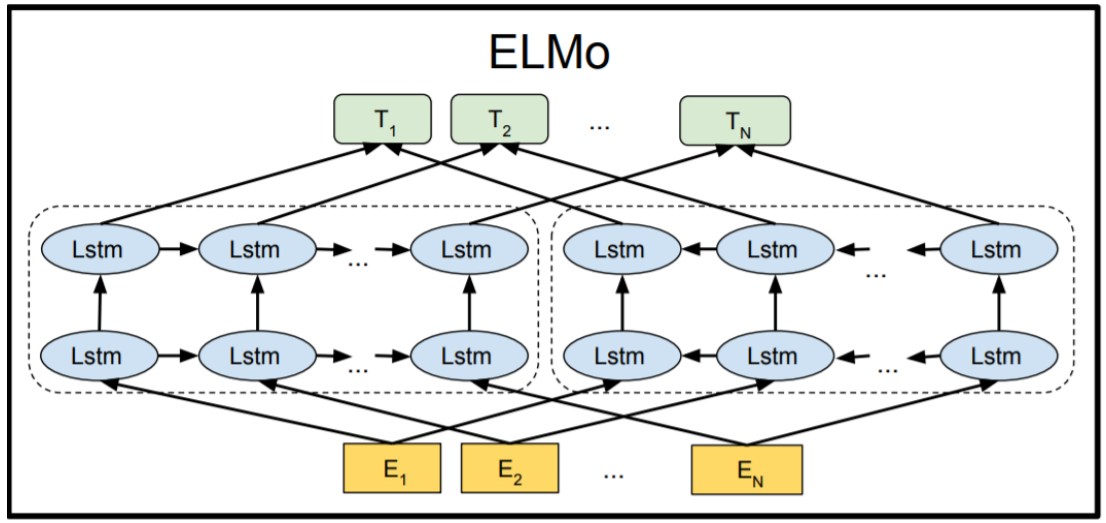
RNN LANGUAGE MODELS

- We can use different RNN architectures for language modeling.



ELMO (Embeddings from Language Models)

- Training **contextualized** word embeddings.
  - The **bank** is flooded.
  - The **bank** is bankrupt.
- Bidirectional model, combining a forward-looking and backward looking LSTMs.
- Architecture combines character-level CNN for creating initial word embeddings and 2-layer LSTM.



https://aclanthology.org/N18-1202/

ELMO (Embeddings from Language Models)

- Representation Trained over massive datasets (for the time)
- Embedding used as input to downstream task.
- Significant improvement compared to W2V/Glove
- Multiple NLP tasks, QA, NLI, SRL, NER, etc.
- Attempts to train ELMO jointly with these tasks were mixed

https://aclanthology.org/W19-4302/

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%

Recall our discussion about attention

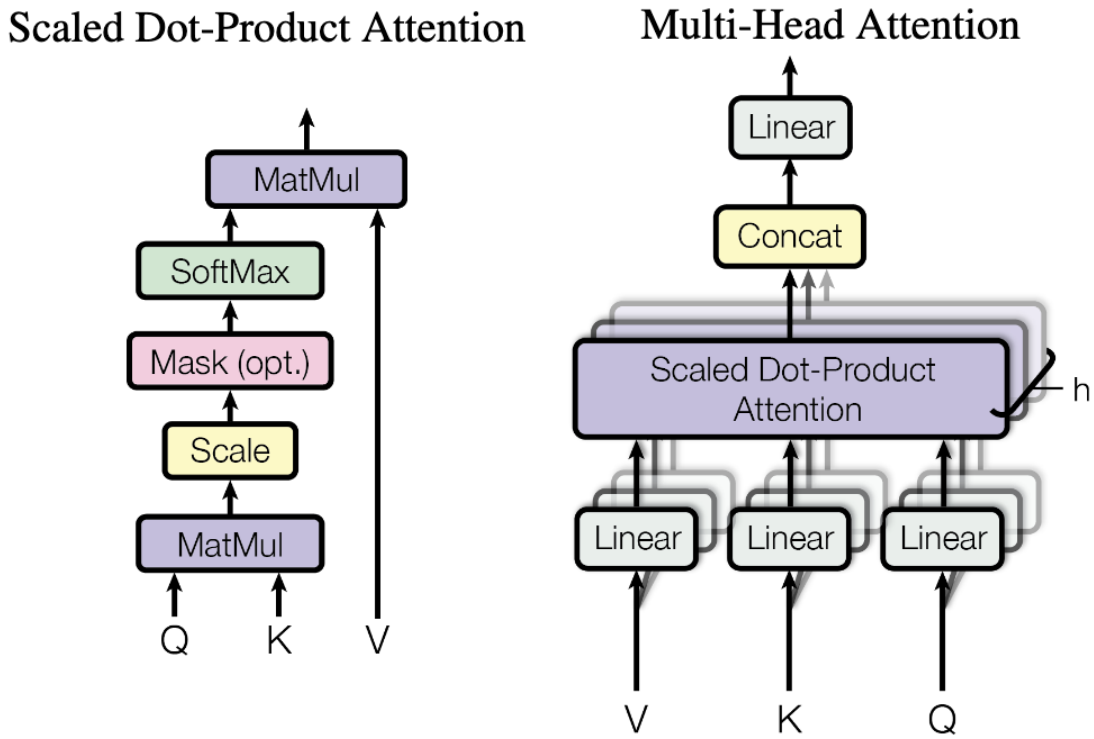
- We saw two example of LM vs. bidirectional embeddings, based on relatively simple architectures.
- We can extend the task of a language modeling to attention-based architectures, such as the transformer.
- We will also see the two modeling approaches here –
  - Encoder models, capturing bidirectional context
  - Decoder models (AKA causal/autoregressive), capturing backward context only.

https://dl.acm.org/doi/10.5555/3295222.3295349

Self-Attention (reminder)

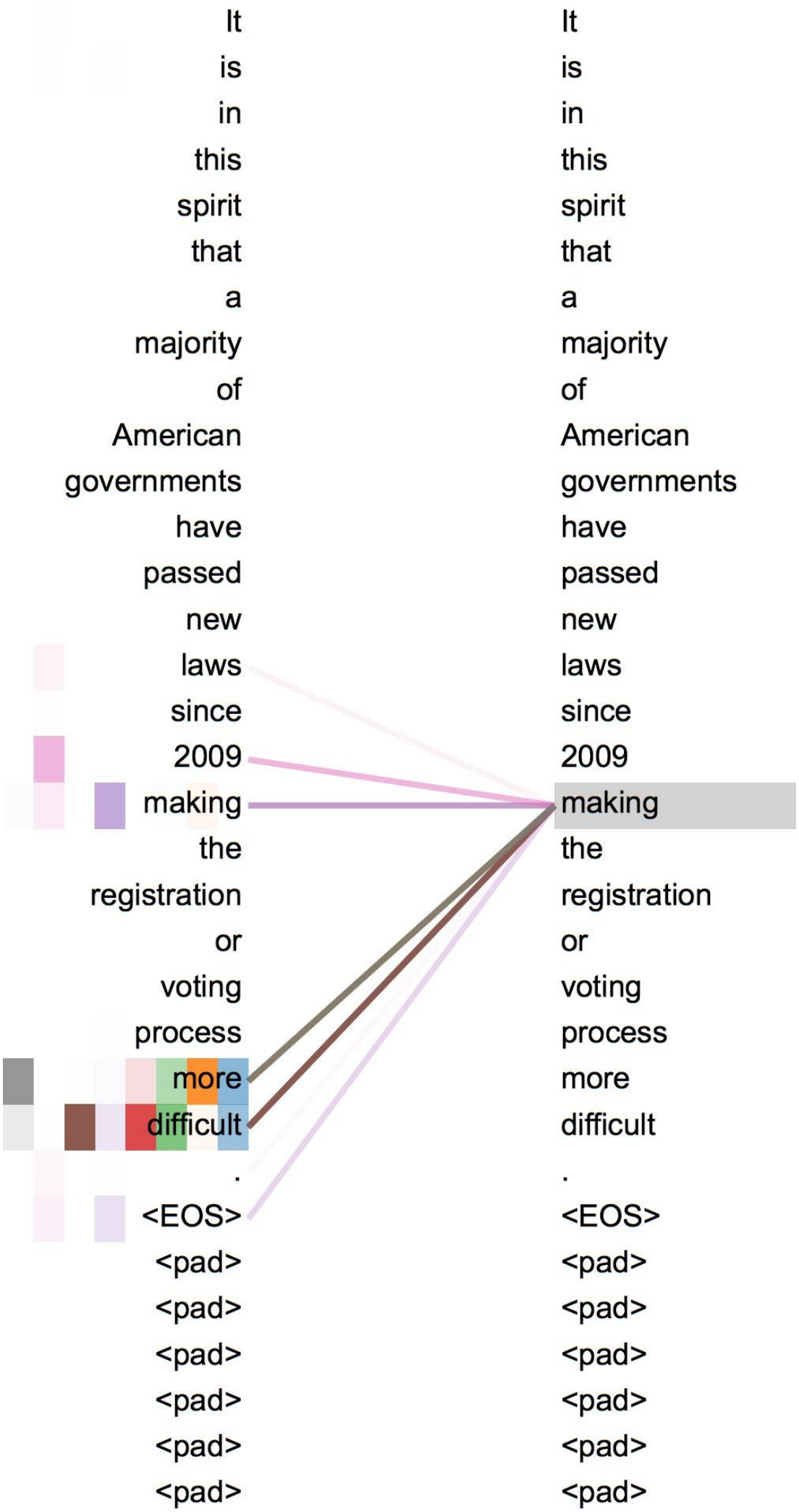
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\textbf{where } \text{head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

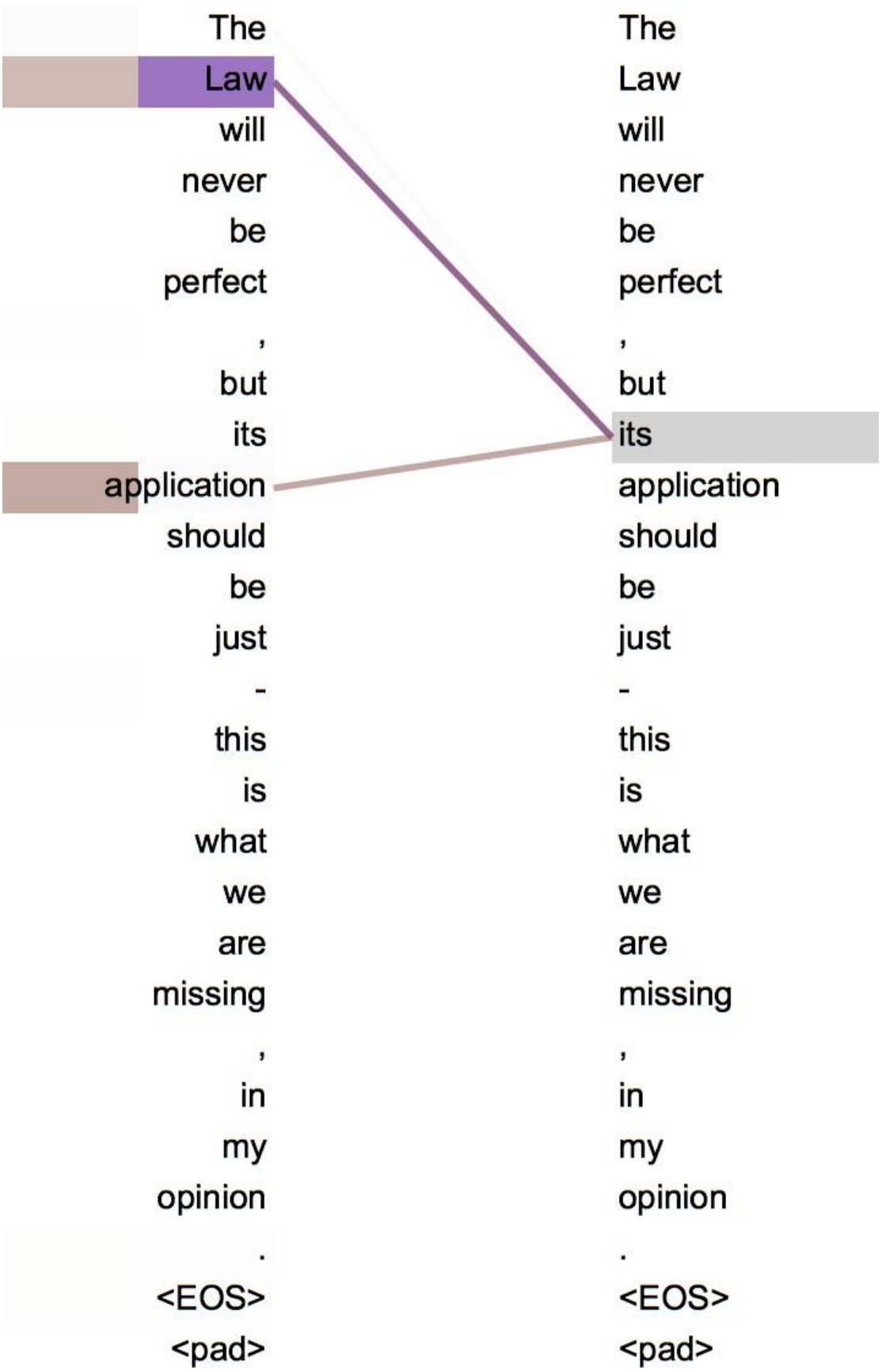


Vaswani et-al 2017

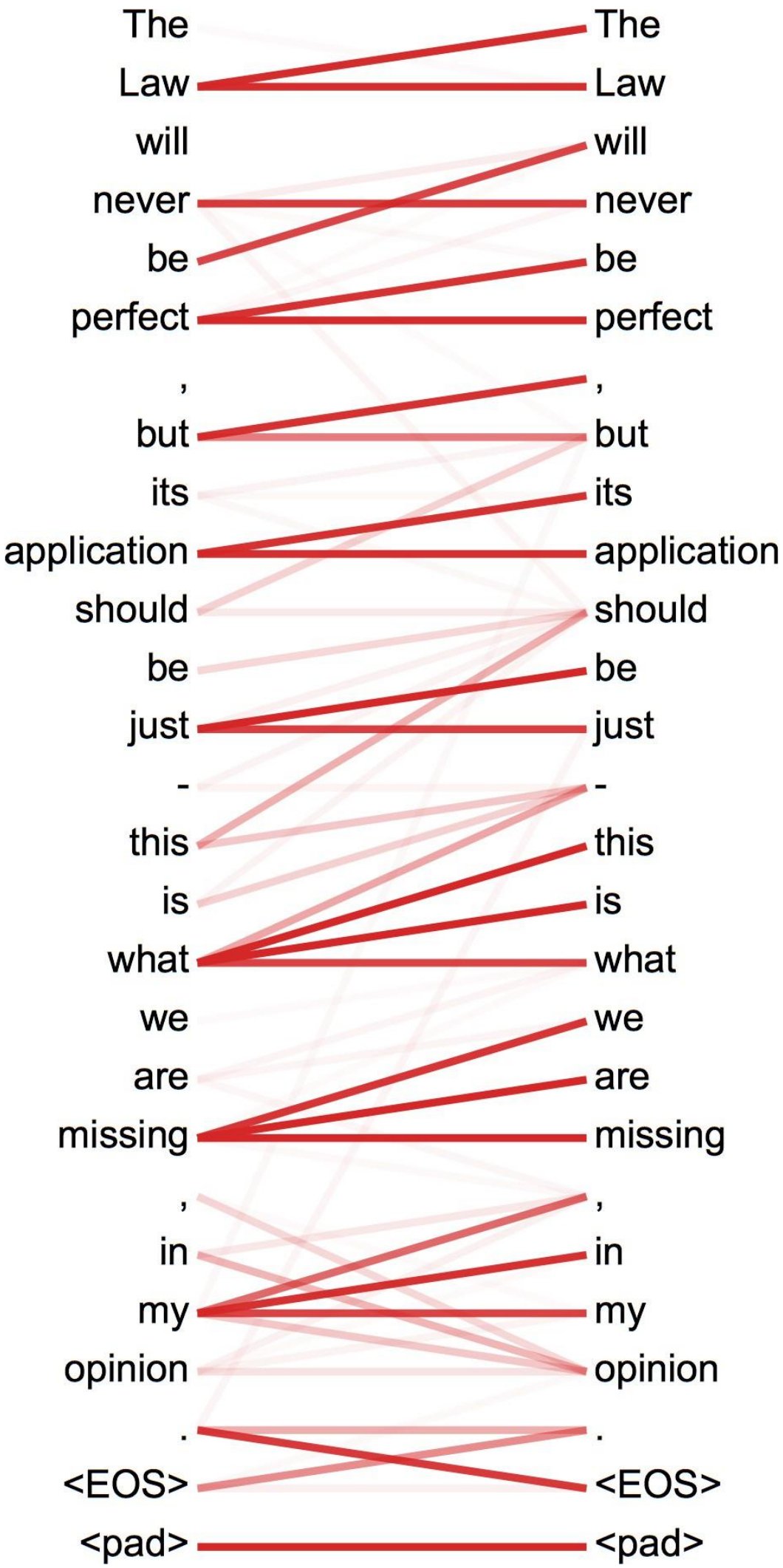
Vaswani et-al 2017



Vaswani et-al 2017

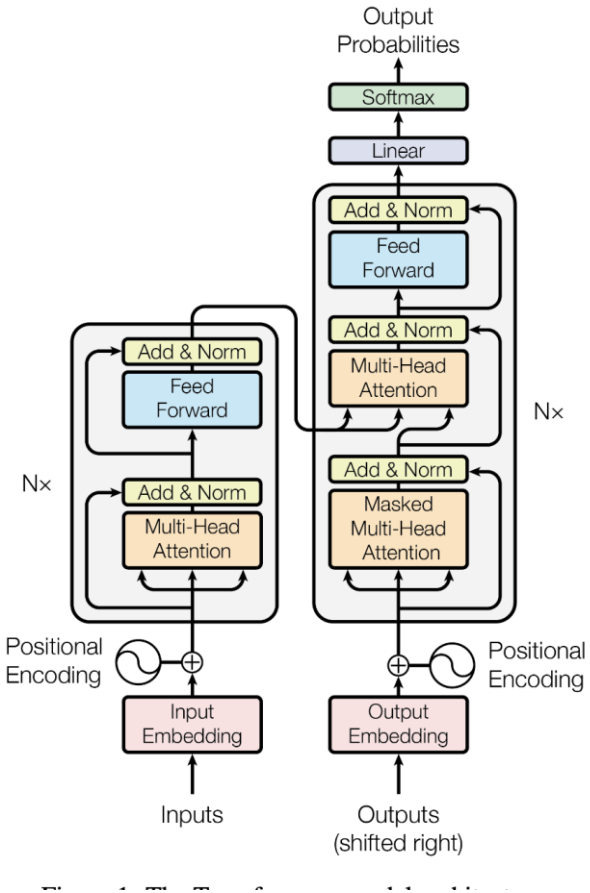


Vaswani et-al 2017



Transformer (reminder)

- Encoder-decoder transformer architecture.
- Both are transformers, but with different role
- Encoder: represent input, **bidirectional** representation.
- Decoder: generates output, cannot “peak” into the future.



Transformer-based LMs

- **Encoder**
  - e.g., BERT, RoBERTa, mmBERT
  - Captures bidirectional context
  - Trained with masked/denoising LM objectives
- **Encoder-Decoder**
  - BART, T5
- **Decoder-only**
  - Autoregressive/Causal LMs.
  - e.g., GPT, LLaMA

Masked Language Modeling

- We've seen autoregressive (causal, left-to-right) LMs.
- But what about tasks for which we want to peak at future tokens?
  - Especially true for tasks where we map each input token to an output token

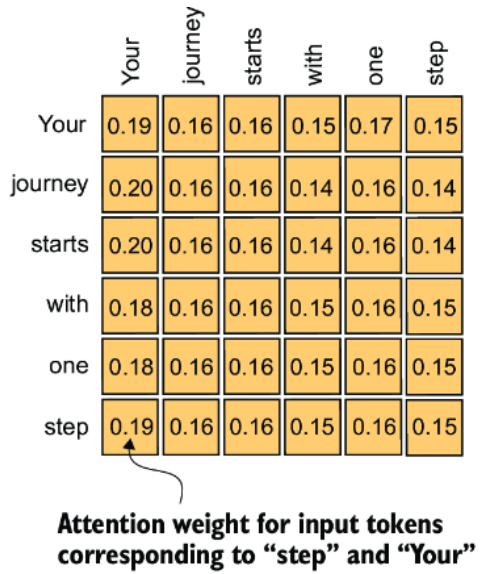
- **Bidirectional** encoders use **masked self-attention** to
  - map sequences of input embeddings (x1,...,x*n*)
  - to sequences of output embeddings of the same length (h1,...,h*n*),
  - where the output vectors have been contextualized using information from the entire input sequence.

Causal vs. Bidirectional LMs

- The distinction can be operationalized easily by removing the attention values from future words

$$\textbf{head} = \text{softmax}\left(\frac{\mathbf{QK}^\mathsf{T}}{\sqrt{d_k}}\right)\mathbf{V}$$

$$\textbf{head} = \text{softmax}\left(\text{mask}\left(\frac{\mathbf{QK}^\mathsf{T}}{\sqrt{d_k}}\right)\right)\mathbf{V}$$



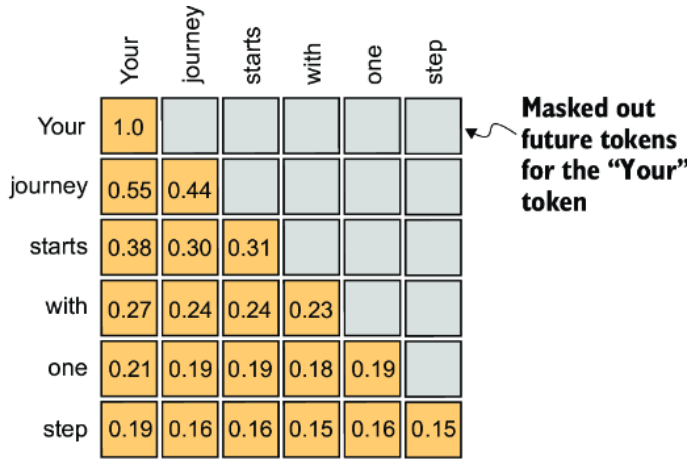


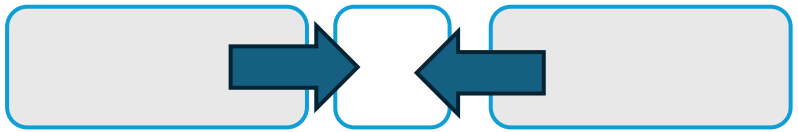
Figure credit: https://medium.com/@sanjjam/beginners-guide-to-causal-attention-b2e3fa9bc762

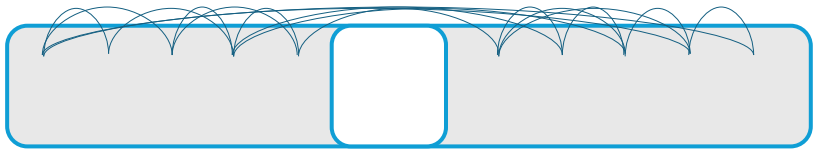
Encoder: BERT (Bidirectional Encoder Representations from Transformers)

- **BERT** (Devlin 2019), encoder only architecture
  - Multiple (12 small) transformer block layers, each with multiple attention heads (12).
  - Context window: 512 tokens.
  - 30,000 English-only tokens (WordPiece tokenizer)
  - Large model (for the time), 340M parameters
- **RoBERTa** (Conneau et al., 2020)
  - Improved version of BERT, more training data, multilingual token.

BERT vs. ELMO

- BERT allowed for direct comparison between the LSTM and transformer approaches.
- Note that ELMO represents backward and forward context separately vs. BERT that is truly bidirectional



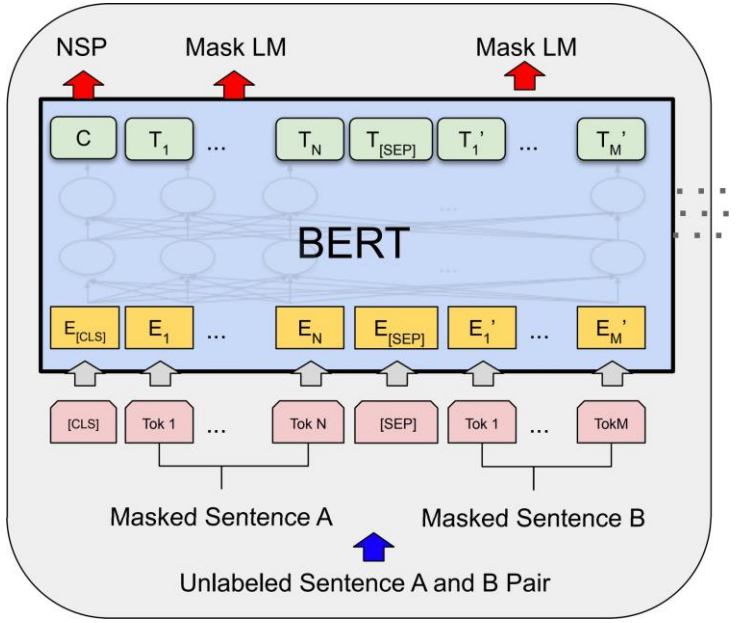


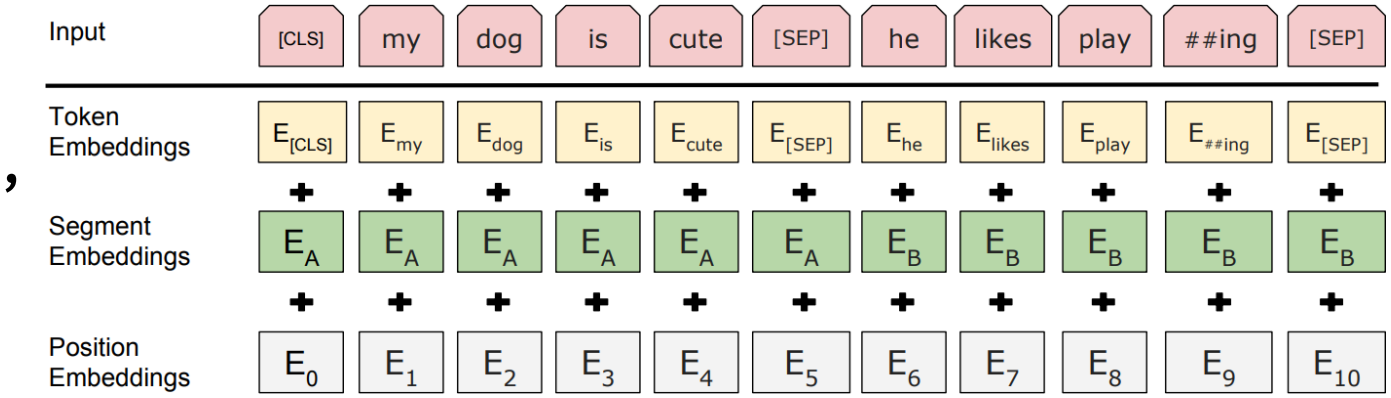
BERT: training

- Training: masked language modeling task:
  - Give a text, predict 15% of the tokens. Out of those, 80% replace the input token with [MASK], 10%, replace w/random, 10%, keep same.
  - [CLS] the black [MASK] meowed all day.
- Sentence level task:
- **Input**: [CLS] Sentence 1 [SEP] sentence2
  - 50% of the time, take the true next chunk of text, 50% of the time take a random other chunk.
  - Predict if the next text is the “true” next text
- BERT objective: masked LM + next sentence prediction

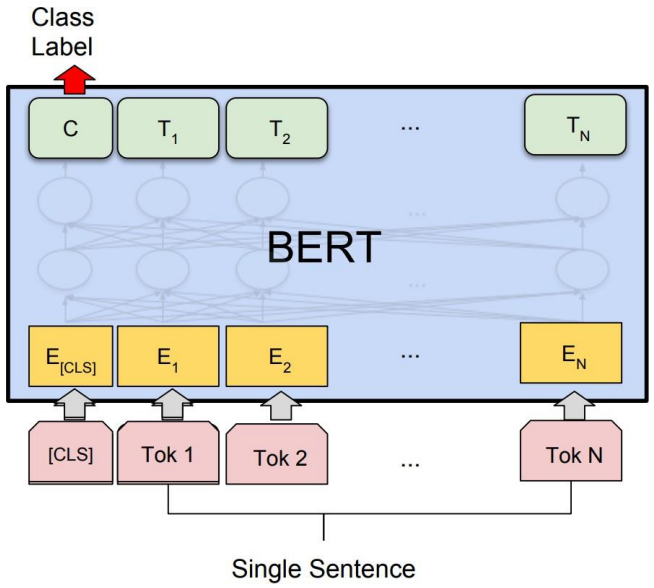
BERT Architecture

- **BERT Base**: 12 layers, 768-dim, 12 heads. Total params = 110M
- **BERT Large**: 24 layers, 1024-dim, 16 heads. Total params = 340M
- Positional embeddings and segment embeddings, 30k word pieces

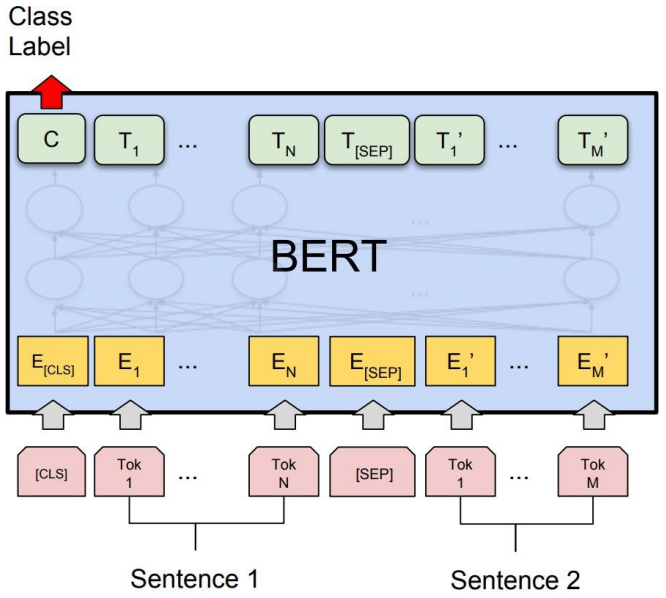




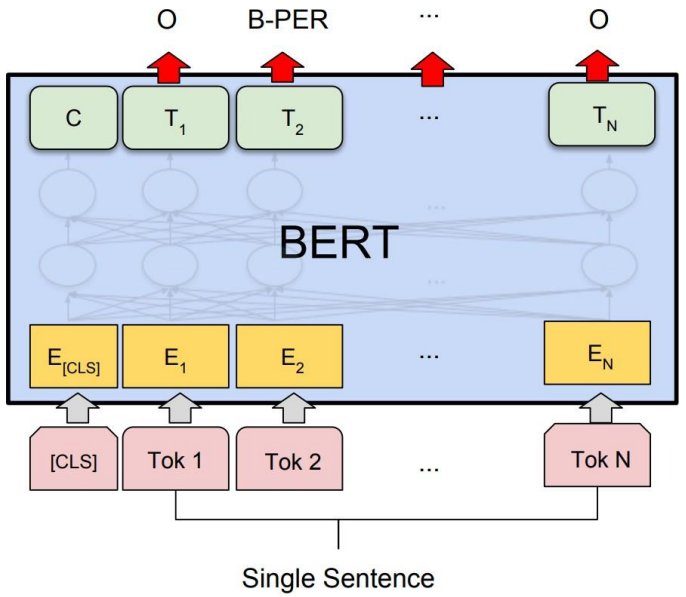
Using BERT



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



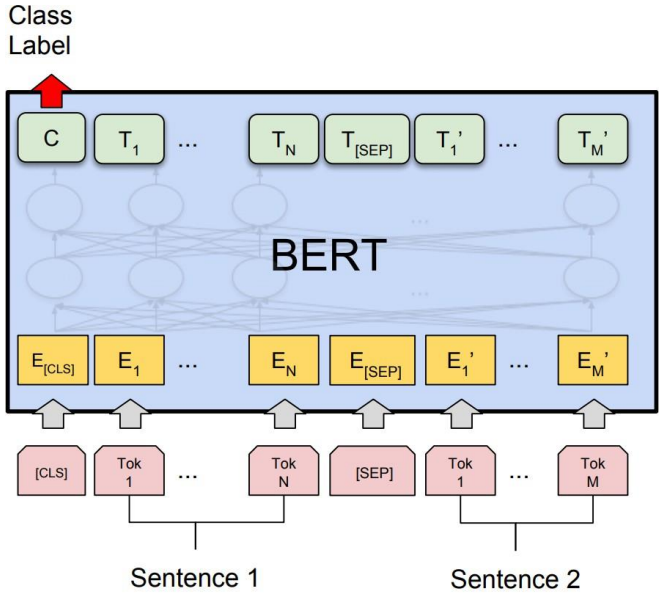
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



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

Using BERT: Textual Inference

- Many NLP task can be formulated as a variant of textual inference.
  - Text 1 + Text 2→ Entails, paraphrases, similar, etc.
- E.g., “Recognizing Textual Entailment” challenge (Dagan, Glickman, Magnini, 2006), SNLI (Bowman etal 2016), etc.
- BERT makes it easy to model such tasks!
  - [CLS] Sentence 1 [SEP] Sentence 2
- The Transformer captures the interactions between the two sentences.



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

What can BERT NOT do?

- BERT **cannot** generate text (at least not in an obvious way)
  - Can fill in [MASK] tokens, but can't generate left-to-right (you can put [MASK] at the end, then predict repeatedly, but this is slow)
- Masked language models are intended to be used primarily for "analysis" tasks, e.g., sequential tagging, semantic similarity between two sentences, …

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.

To Tune or not to Tune?

- Unlike ELMO, BERT can benefit from task-specific training (backprop to BERT)

Pretraining	Adaptation	NER CoNLL 2003	SA SST-2	Nat. lang. inference MNLI	Nat. lang. inference SICK-E	Semantic textual similarity SICK-R	Semantic textual similarity MRPC	Semantic textual similarity STS-B
Skip-thoughts	❄️	-	81.8	62.9	-	86.6	75.8	71.8
ELMo	❄️	91.7	**91.8**	**79.6**	**86.3**	**86.1**	**76.0**	**75.9**
	🔥	**91.9**	91.2	76.4	83.3	83.3	74.7	75.5
	Δ=🔥-❄️	0.2	-0.6	-3.2	-3.3	-2.8	-1.3	-0.4
BERT-base	❄️	92.2	93.0	**84.6**	84.8	86.4	78.1	82.9
	🔥	**92.4**	**93.5**	**84.6**	**85.8**	**88.7**	**84.8**	**87.1**
	Δ=🔥-❄️	0.2	0.5	0.0	1.0	2.3	6.7	4.2

https://aclanthology.org/W19-4302.pdf

ROBERTA (Robustly Optimized BERT Pretraining Approach)

- RoBERTa uses 10x more training data, larger batch sizes, dynamic masking, and removing the next-sentence prediction task.
- 160GB of data instead of 16 GB

Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7

Liu et al. (2019) https://arxiv.org/abs/1907.11692

DistilBERT

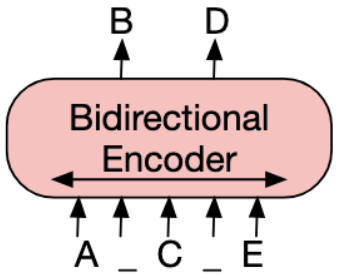
- "Reduce the size of a BERT model by 40%, while retaining 97%

of its language understanding capabilities and being 60% faster.” via knowledge distillation.

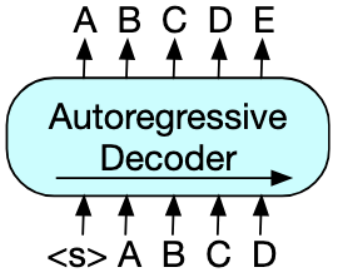
Sanh et-al , 2019 https://arxiv.org/abs/1910.01108

BERT (encoder only) vs. GPT (decoder only)

- **BERT** is an encoder only model, cannot be used for left-to-right generation tasks.
- GPT, and other decoder-only models, autoregressive LM can be used for generation tasks.



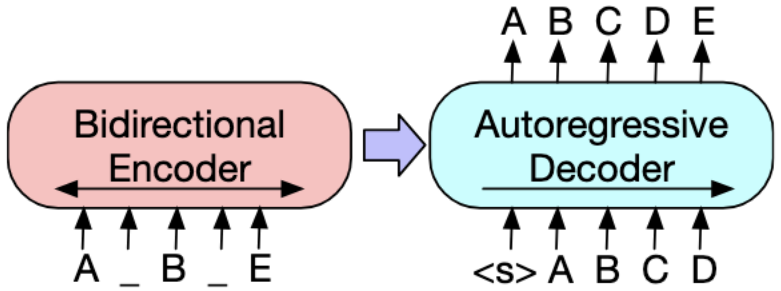
(a) BERT: Random tokens are replaced with masks, and the document is encoded bidirectionally. Missing tokens are predicted independently, so BERT cannot easily be used for generation.



(b) GPT: Tokens are predicted auto-regressively, meaning GPT can be used for generation. However words can only condition on leftward context, so it cannot learn bidirectional interactions.

Lewis et-al 2019 https://arxiv.org/pdf/1910.13461

Encoder-Decoder Models

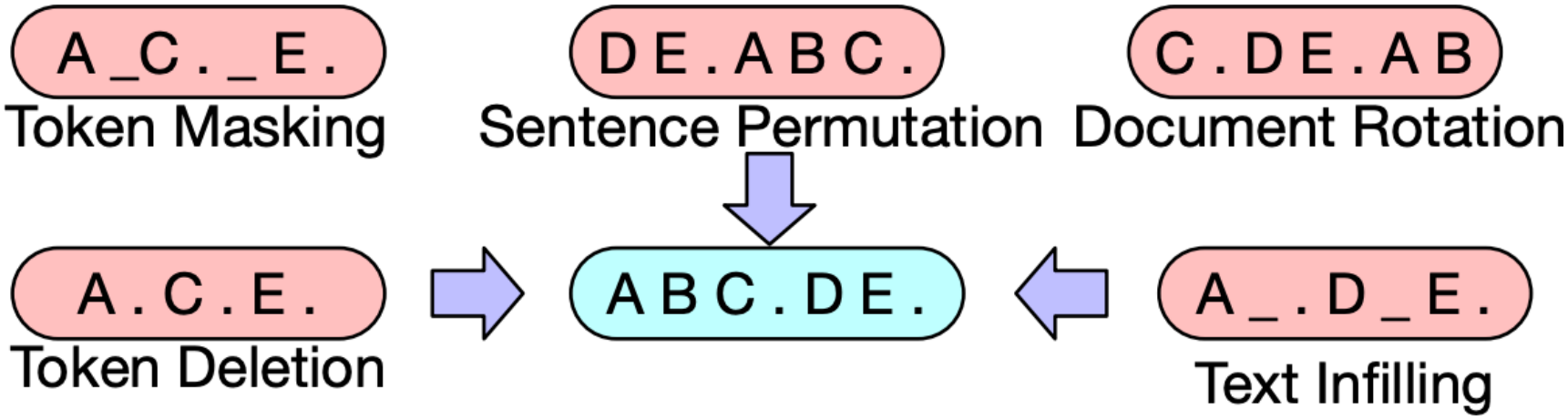


(c) BART: Inputs to the encoder need not be aligned with decoder outputs, allowing arbitary noise transformations. Here, a document has been corrupted by replacing spans of text with mask symbols. The corrupted document (left) is encoded with a bidirectional model, and then the likelihood of the original document (right) is calculated with an autoregressive decoder. For fine-tuning, an uncorrupted document is input to both the encoder and decoder, and we use representations from the final hidden state of the decoder.

Lewis et-al 2019 https://arxiv.org/pdf/1910.13461

BART (Bidirectional and Auto-Regressive Transformers)

- sequence-to-sequence model.
- Training: (1) text is corrupted with several noise function. (2) Seq2Seq model predict the original text.
- Noise functions:



Lewis et-al 2019 https://arxiv.org/pdf/1910.13461

Model	SQuAD 1.1 F1	MNLI Acc	ELI5 PPL	XSum PPL	ConvAI2 PPL	CNN/DM PPL
BERT Base (Devlin et al., 2019)	88.5	**84.3**	-	-	-	-
Masked Language Model	90.0	83.5	24.77	7.87	12.59	7.06
Masked Seq2seq	87.0	82.1	23.40	6.80	11.43	6.19
Language Model	76.7	80.1	**21.40**	7.00	11.51	6.56
Permuted Language Model	89.1	83.7	24.03	7.69	12.23	6.96
Multitask Masked Language Model	89.2	82.4	23.73	7.50	12.39	6.74
BART Base						
w/ Token Masking	90.4	84.1	25.05	7.08	11.73	6.10
w/ Token Deletion	90.4	84.1	24.61	6.90	11.46	5.87
w/ Text Infilling	**90.8**	84.0	24.26	**6.61**	**11.05**	5.83
w/ Document Rotation	77.2	75.3	53.69	17.14	19.87	10.59
w/ Sentence Shuffling	85.4	81.5	41.87	10.93	16.67	7.89
w/ Text Infilling + Sentence Shuffling	**90.8**	83.8	24.17	6.62	11.12	**5.41**

Table 1: Comparison of pre-training objectives. All models are of comparable size and are trained for 1M steps on a combination of books and Wikipedia data. Entries in the bottom two blocks are trained on identical data using the same code-base, and fine-tuned with the same procedures. Entries in the second block are inspired by pre-training objectives proposed in previous work, but have been simplified to focus on evaluation objectives (see §4.1). Performance varies considerably across tasks, but the BART models with text infilling demonstrate the most consistently strong performance.

	CNN/DailyMail			XSum		
	R1	R2	RL	R1	R2	RL
Lead-3	40.42	17.62	36.67	16.30	1.60	11.95
PTGEN (See et al., 2017)	36.44	15.66	33.42	29.70	9.21	23.24
PTGEN+COV (See et al., 2017)	39.53	17.28	36.38	28.10	8.02	21.72
UniLM	43.33	20.21	40.51	-	-	-
BERTSUMABS (Liu & Lapata, 2019)	41.72	19.39	38.76	38.76	16.33	31.15
BERTSUMEXTABS (Liu & Lapata, 2019)	42.13	19.60	39.18	38.81	16.50	31.27
BART	**44.16**	**21.28**	**40.90**	**45.14**	**22.27**	**37.25**

	SQuAD 1.1	SQuAD 2.0	MNLI	SST	QQP	QNLI	STS-B	RTE	MRPC	CoLA
	EM/F1	EM/F1	m/mm	Acc	Acc	Acc	Acc	Acc	Acc	Mcc
BERT	84.1/90.9	79.0/81.8	86.6/-	93.2	91.3	92.3	90.0	70.4	88.0	60.6
UniLM	-/-	80.5/83.4	87.0/85.9	94.5	-	92.7	-	70.9	-	61.1
XLNet	**89.0**/94.5	86.1/88.8	89.8/-	95.6	91.8	93.9	91.8	83.8	89.2	63.6
RoBERTa	88.9/**94.6**	**86.5/89.4**	**90.2/90.2**	96.4	92.2	94.7	**92.4**	86.6	**90.9**	**68.0**
BART	88.8/**94.6**	86.1/89.2	89.9/90.1	**96.6**	**92.5**	**94.9**	91.2	**87.0**	90.4	62.8

Table 2: Results for large models on SQuAD and GLUE tasks. BART performs comparably to RoBERTa and XLNet, suggesting that BART's uni-directional decoder layers do not reduce performance on discriminative tasks.

T5 (Text-to-Text Transfer Transformer)

- treats every NLP task (translation, summarization, classification) as a text-to-text problem.

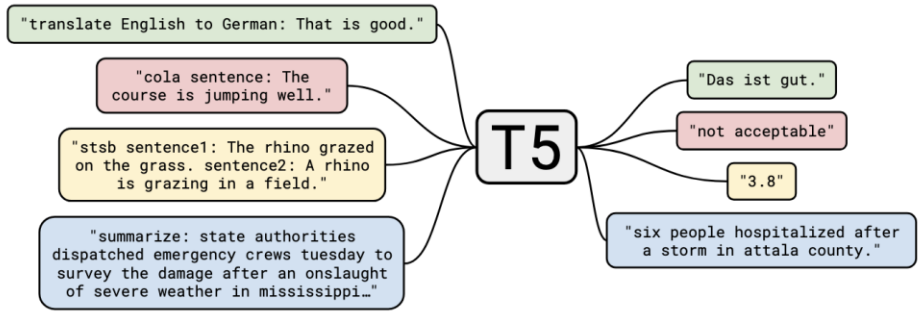


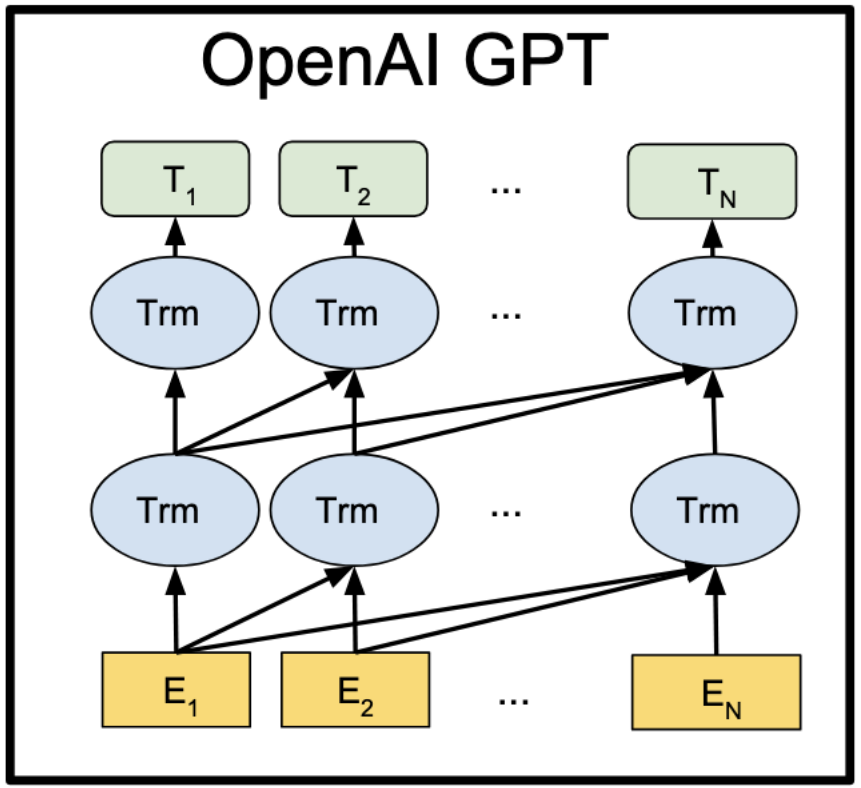
Figure 1: A diagram of our text-to-text framework. Every task we consider—including translation, question answering, and classification—is cast as feeding our model text as input and training it to generate some target text. This allows us to use the same model, loss function, hyperparameters, etc. across 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".

Raffel et-al 2020 https://arxiv.org/abs/1910.10683

GPT/GPT2 (Generative Pre-Training)

- **decoder-only** model, doing autoregressive next-token prediction
- GPT2: trained on 40GB of text collected from reddit, tested on downstream tasks
- 1.5B parameters, largest at the time

Parameters	Layers	d_{model}
117M	12	768
345M	24	1024
762M	36	1280
1542M	48	1600



Radford et al 2019

Zero-Shot Learning

	LAMBADA (PPL)	LAMBADA (ACC)	CBT-CN (ACC)	CBT-NE (ACC)	WikiText2 (PPL)	PTB (PPL)	enwik8 (BPB)	text8 (BPC)	WikiText103 (PPL)	1BW (PPL)
SOTA	99.8	59.23	85.7	82.3	39.14	46.54	0.99	1.08	18.3	**21.8**
117M	**35.13**	45.99	**87.65**	**83.4**	**29.41**	65.85	1.16	1.17	37.50	75.20
345M	**15.60**	55.48	**92.35**	**87.1**	**22.76**	47.33	1.01	**1.06**	26.37	55.72
762M	**10.87**	**60.12**	**93.45**	**88.0**	**19.93**	**40.31**	**0.97**	**1.02**	22.05	44.575
1542M	**8.63**	**63.24**	**93.30**	**89.05**	**18.34**	**35.76**	**0.93**	**0.98**	**17.48**	42.16

Table 3. Zero-shot results on many datasets. No training or fine-tuning was performed for any of these results. PTB and WikiText-2 results are from (Gong et al., 2018). CBT results are from (Bajgar et al., 2016). LAMBADA accuracy result is from (Hoang et al., 2018) and LAMBADA perplexity result is from (Grave et al., 2016). Other results are from (Dai et al., 2019).