CPSC 670: Topics in Natural Language Processing

Yale University

SPRING 2023

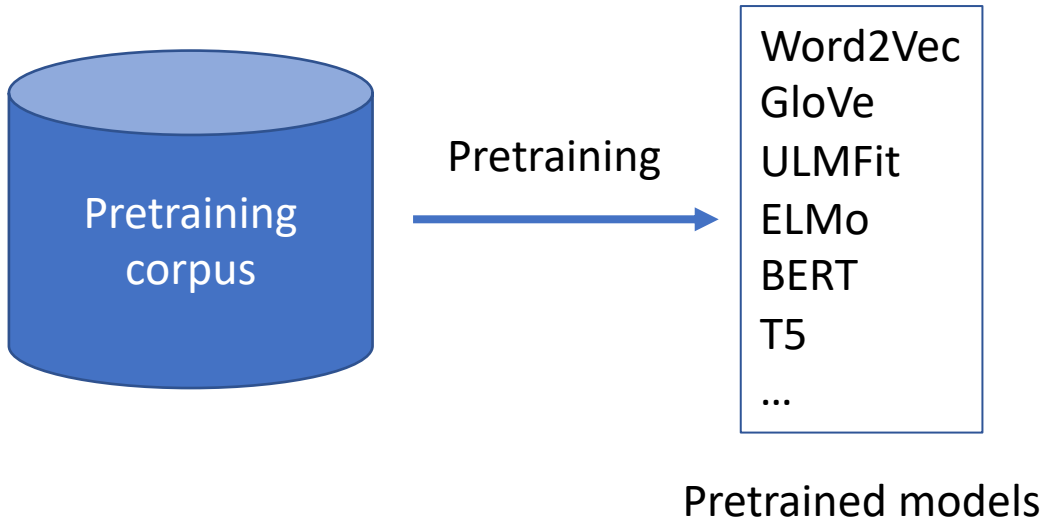
Lecture 2: Transfer learning in NLP

Transfer learning in NLP

- The general transfer learning refers:
  - Training a model to perform one task/dataset/set of tasks
  - Then transfer to another task/dataset/set of tasks
  - Often refers to training a language model then transferring to downstream tasks

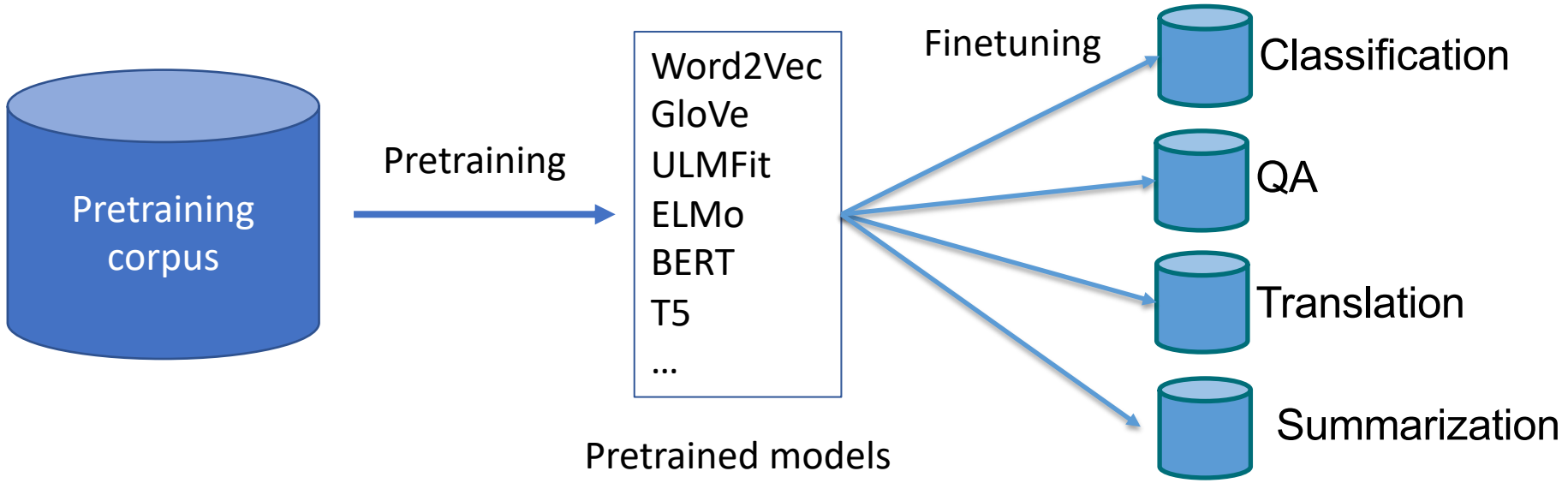
Transfer learning in NLP

- The general transfer learning refers:
  - Training a model to perform one task/dataset/set of tasks
  - Then transfer to another task/dataset/set of tasks
  - Often refers to training a language model then transferring to downstream tasks



Transfer learning in NLP

- The general transfer learning refers:
  - Training a model to perform one task/dataset/set of tasks
  - Then transfer to another task/dataset/set of tasks
  - Often refers to training a language model then transferring to downstream tasks



Transfer learning in NLP

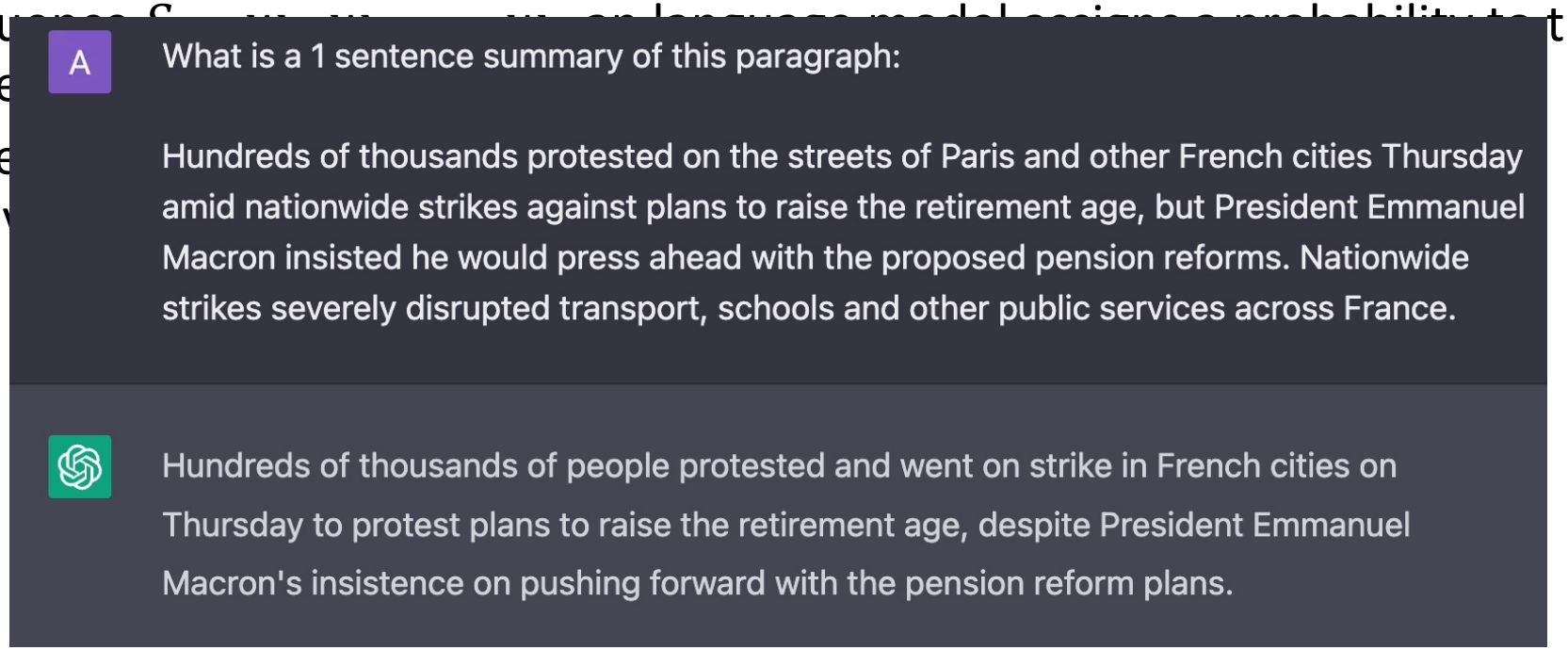
- **Language Model Pretraining** is among the most successful instances of transfer learning
- What is a Language Model?

Transfer learning in NLP

- **Language Model Pretraining** is among the most successful instances of transfer learning
- What is a Language Model?
  - A language model is a probability distribution over sequences of words
  - Given a sequence $S = w_1, w_2, \ldots, w_n$ an language model assigns a probability to the whole sequence: $P(S) = P(w_1, \ldots, w_n)$
  - We can have a conditional language model that predicts the probability of one sequence given another sequence
  - $P(S_1|S_2)$

Transfer learning in NLP

- **Language Model Pretraining** is among the most successful instances of transfer learning
- What is a Language Model?
  - A language model is a probability distribution over sequences of words
  - Given a sequ… …the whole seque…
  - We can have… sequence giv…
  - $P(S_1|S_2)$



Background - Prior work

- Learning word representations (word embeddings)
- How can we learn powerful word representations?

Background - Prior work

- Learning word representations (word embeddings)
- A very simple recipe:
  - Given a word predict its context words

Wind turbines generate electricity

Background - Prior work

- Learning word representations (word embeddings)
- A very simple recipe:
  - Given a word predict its context words
  - For example given the word **turbines** we want to predict **wind**

Wind turbines generate electricity

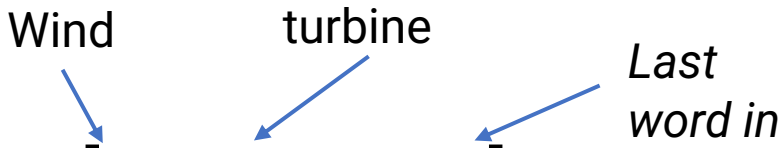
Background - Prior work

- Learning word representations (word embeddings)
- A very simple recipe:
  - Given a word in a sequence, predict its context words
  - For example given the word **turbines** we want to predict **wind**
  - How? Train a neural network that can assign high probability to the word wind

Wind turbines generate electricity

Background – Word representation learning

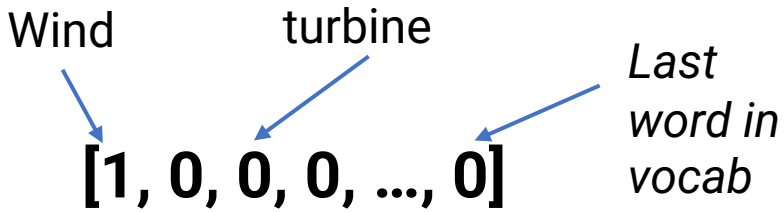
- First assume we have a fixed vocabulary of size ***V***.
- We use 1-hot representations
  - A vector where the entry to relevant word is **1** and everything else is **0**



Wind	**[1, 0, 0, 0, …, 0]**
turbines	**[0, 0, 1, 0, …, 0]**
generate	**[0, 0, 0, 1, …, 0]**
electricity	**[0, 1, 0, 0, …, 0]**

Background – Word representation learning

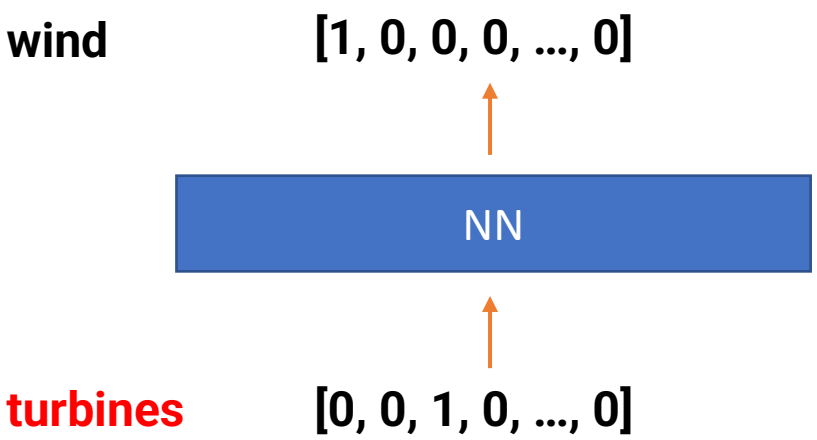
- First assume we have a fixed vocabulary of size ***V***.
- We use 1-hot representations
  - A vector where the entry to relevant word is **1** and everything else is **0**



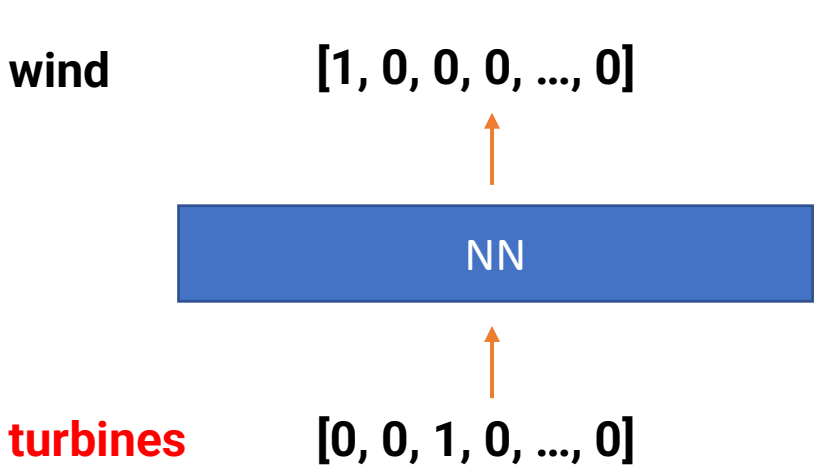
Wind **[1, 0, 0, 0, …, 0]**
turbines **[0, 0, 1, 0, …, 0]**
generate **[0, 0, 0, 1, …, 0]**
electricity **[0, 1, 0, 0, …, 0]**

Then the goal is to transform an input 1-hot representation (context word) to an output 1-hot representation (target word)

Background – Word representation learning



Background – Word representation learning



$[y_1, y_2, y_3, \dots y_j, \dots, y_V]$

$[h_1, h_2, \dots, h_N]$

$[x_1, x_2, x_3, \dots x_i, \dots, x_V]$

$x_i, y_j \in \{0, 1\}$

$h_k \in \mathbb{R}$

Background – Word representation learning

$[y_1, y_2, y_3, \dots y_j, \dots, y_V]$

$[h_1, h_2, \dots, h_N]$

1

Linear transformation 1

$\mathbf{h} = \mathbf{x}^T \mathbf{W_1}$

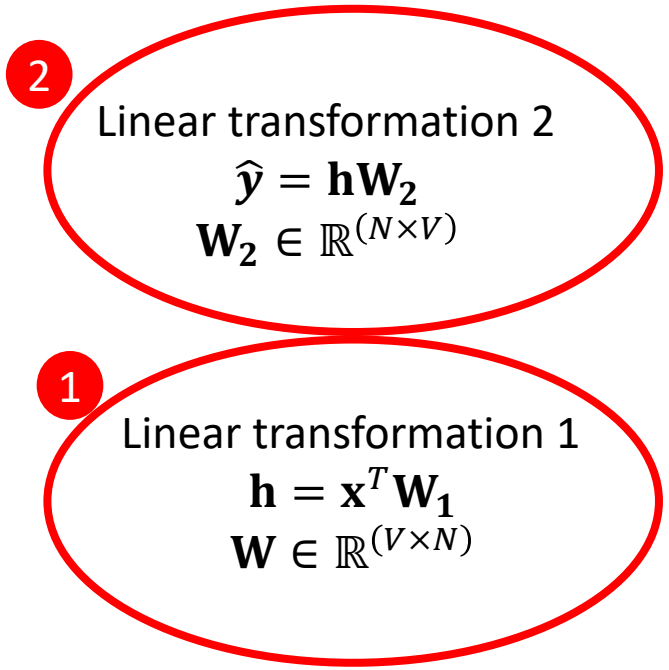
$\mathbf{W} \in \mathbb{R}^{(V \times N)}$

$[x_1, x_2, x_3, \dots x_i, \dots, x_V]$

$x_i, y_j \in \{0, 1\}$

$h_k \in \mathbb{R}$

Background – Word representation learning



$[y_1, y_2, y_3, \dots y_j, \dots, y_V]$

2

$[h_1, h_2, \dots, h_N]$

1

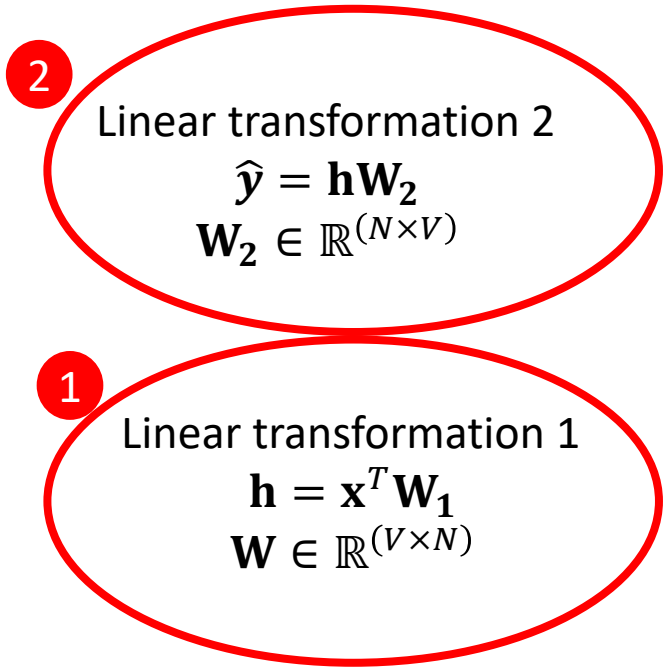
$[x_1, x_2, x_3, \dots x_i, \dots, x_V]$

$x_i, y_j \in \{0, 1\}$

$h_k \in \mathbb{R}$

Background – Word representation learning

Train by minimizing $\mathbf{loss}(\hat{y}, y)$



$[y_1, y_2, y_3, \dots y_j, \dots, y_V]$

2

$[h_1, h_2, \dots, h_N]$

1

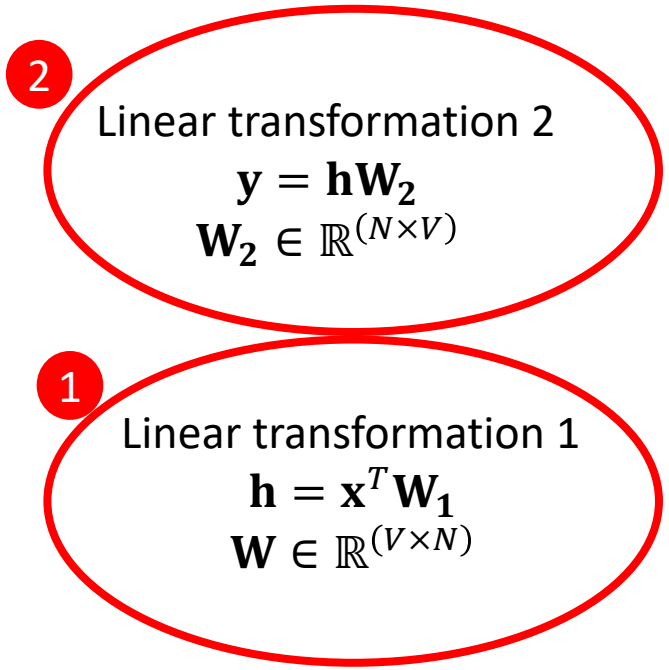
$[x_1, x_2, x_3, \dots x_i, \dots, x_V]$

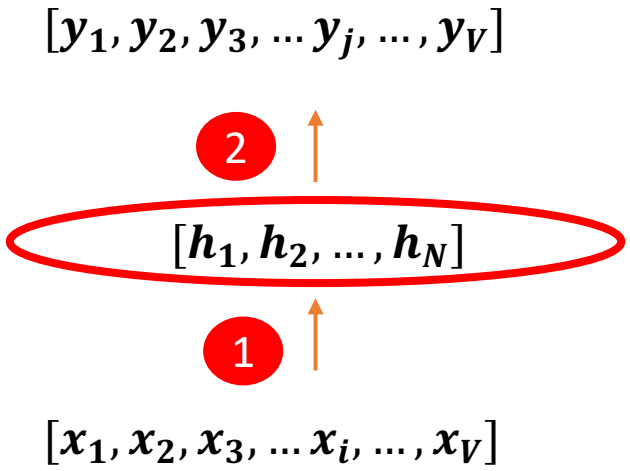
$x_i, y_j \in \{0, 1\}$

$h_k \in \mathbb{R}$

Background – Word representation learning

Train by minimizing $\mathbf{loss}(\hat{y}, y)$





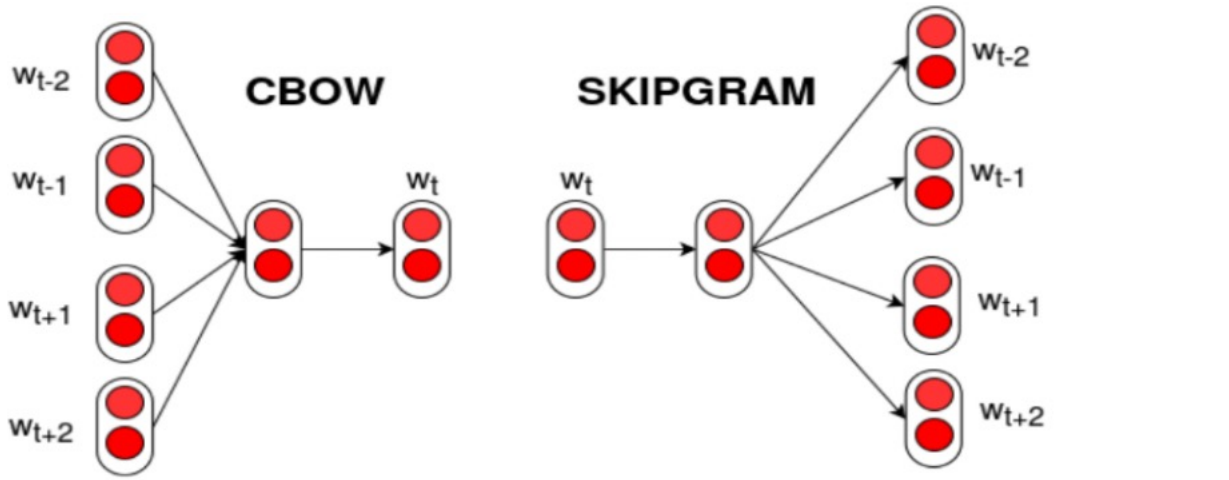
Vector $\mathbf{h}$ will be our word embeddings

$x_i, y_j \in \{0, 1\}$
$h_k \in \mathbb{R}$

Background – Word representation learning

- Word2Vec (Mikolov et al., 2013)

Efficient algorithm + large scale training → high quality word vectors



$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} logp(w_t|w_{t+j})$$

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} logp(w_{t+j}|w_t)$$

ELMo: Deep contextualized word representations

Peters et al., 2018

ELMo (Deep contextualized representations)

- Earlier word embedding methods (e.g., Word2Vec, GloVe, FastText) learn a single “static” vector for each word

ELMo (Deep contextualized representations)

- Earlier word embedding methods (e.g., Word2Vec, GloVe, FastText) learn a single “static” vector for each word
- Problem: Static embeddings are not flexible and expressive enough

ELMo (Deep contextualized representations)

- Earlier word embedding methods (e.g., Word2Vec, GloVe, FastText) learn a single “static” vector for each word
- Problem: Static embeddings are not flexible and expressive enough

The children love to play outside in the park.

She went to see a play at the local theater.

They play the piano beautifully.

ELMo (Deep contextualized representations)

- Earlier word embedding methods (e.g., Word2Vec, GloVe, FastText) learn a single “static” vector for each word
- Problem: Static embeddings are not flexible and expressive enough

The children love to play outside in the park.

She went to see a play at the local theater.

They play the piano beautifully.

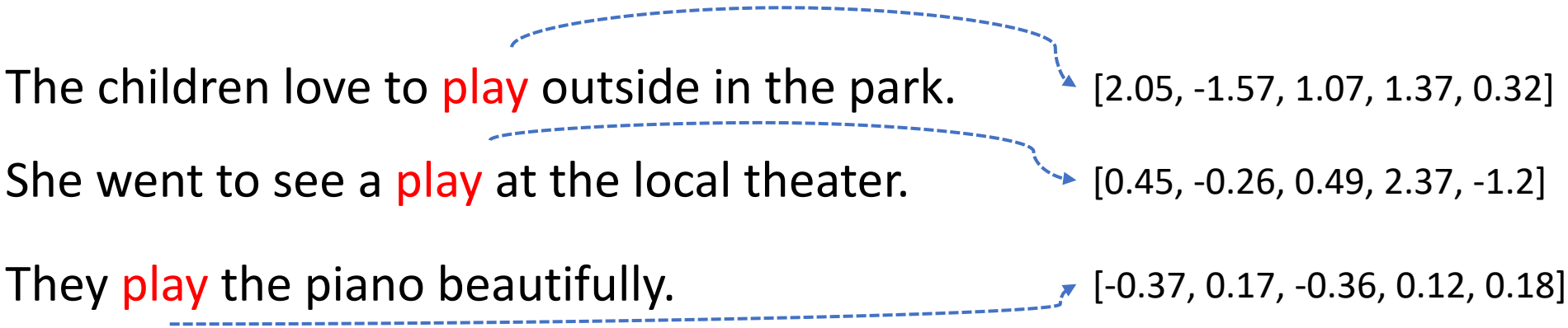
Information from context is necessary to capture the correct meaning of the word.

ELMo (Peters, et al 2018)

- Tries to address the same exact problem
  - Prior methods: Static embeddings
  - ELMo: Context-dependent embeddings
    - Word representations are a function of the entire sequence

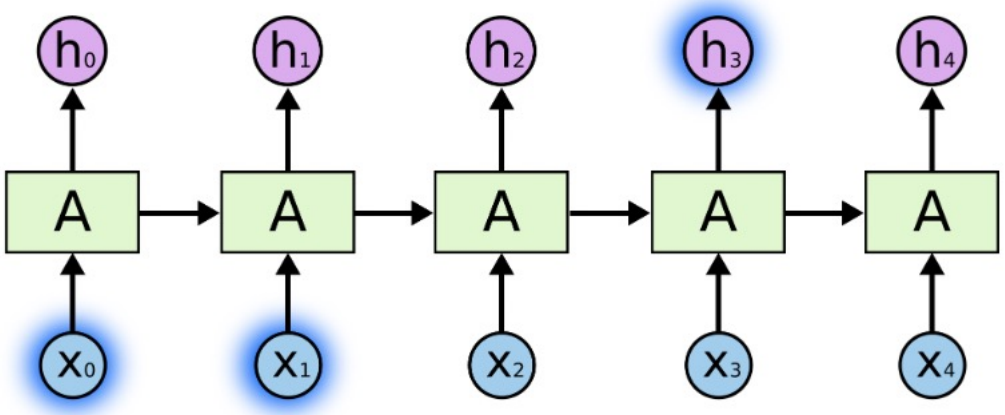
ELMo (Peters, et al 2018)

- Tries to address the same exact problem
  - Prior methods: Static embeddings
  - ELMo: Context-dependent embeddings
    - Word representations are a function of the entire sequence



ELMo Building Blocks

- A recurrent neural network (RNN)



  - The output of each step depends on the input to that step and previous state of the network.
  - Convenient approach for modeling sequences

Fig from https://colah.github.io/posts/2015-08-Understanding-LSTMs/

ELMo Building Blocks

- LSTMs (Long Short Term Memory)
  - A type of RNN that shows better performance and is stable in training
  - Includes additional parameters and gating mechanism to better control the follow of information
  - Allows it to “forget” irrelevant info

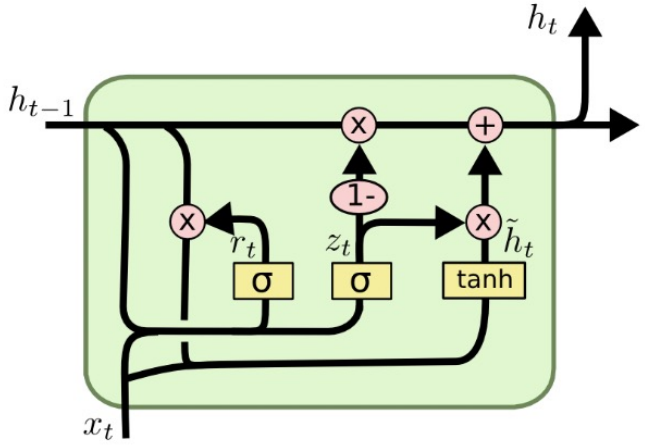


Fig from https://colah.github.io/posts/2015-08-Understanding-LSTMs/

ELMo Building Blocks

- Bidirectional LSTM
  - Leverage both left-side and right-side context
  - In practice shows better performance

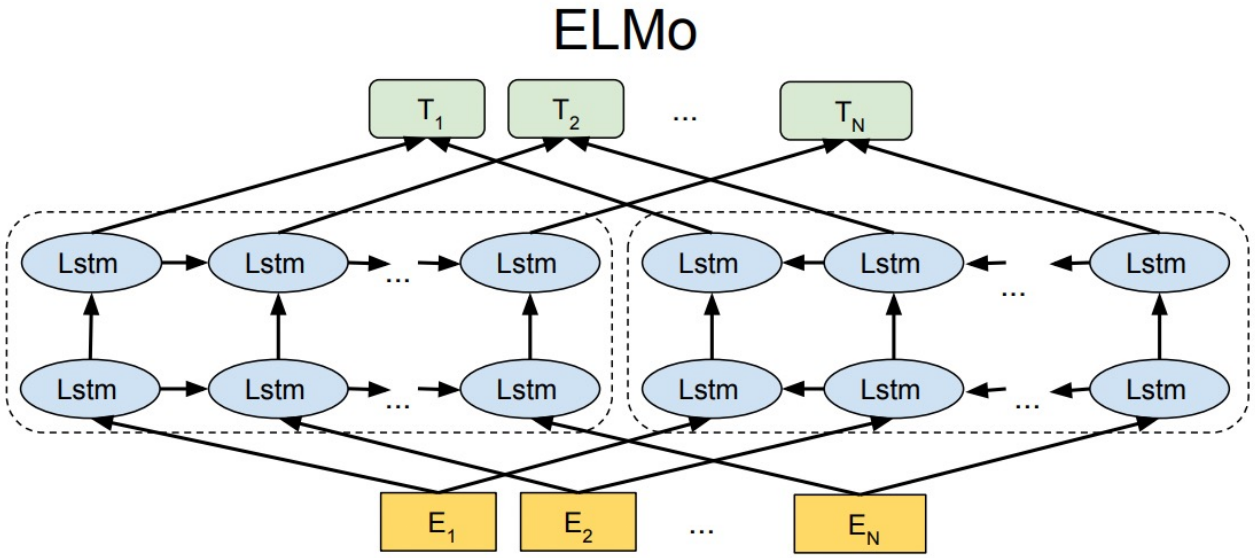
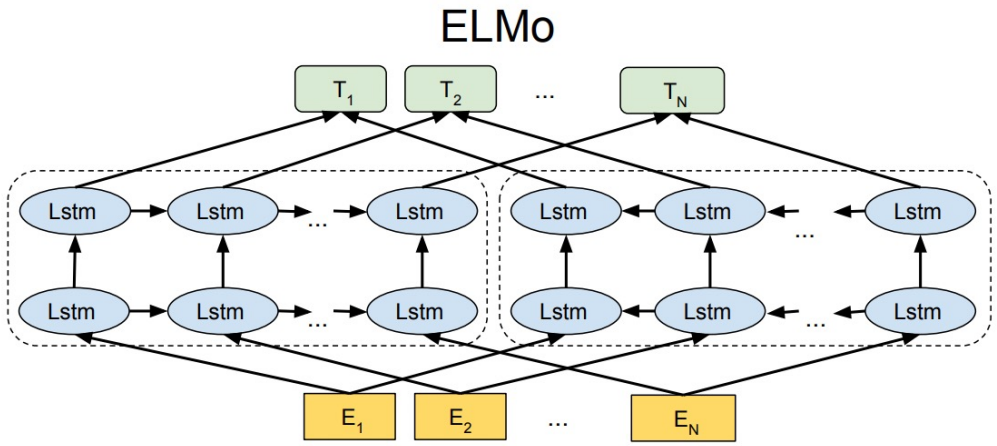


Fig from https://arxiv.org/pdf/1810.04805.pdf

ELMo architecture

- Bidirectional LSTM Language Model



A forward LM computes the probability of sequence by modeling the probability of token t_k given left-side history

$$p(t_1, \dots, t_k) = \prod_{k=1}^{N} p(t_k | t_1, \dots, t_{k-1})$$

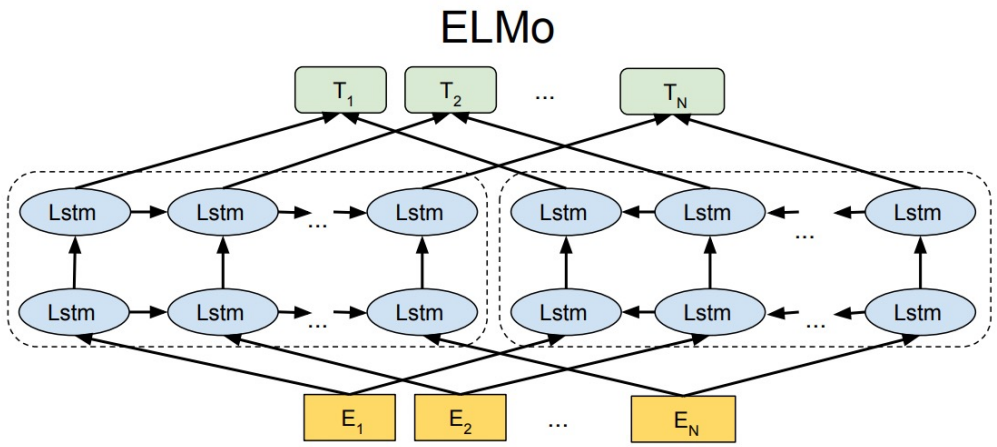
This is done by a Multi-Layer LSTM

The output of each layer j is a hidden representation $\vec{h}_{k,j}^{LM}$

The top layer output $\vec{h}_{k,L}^{LM}$ is used to predict the next word

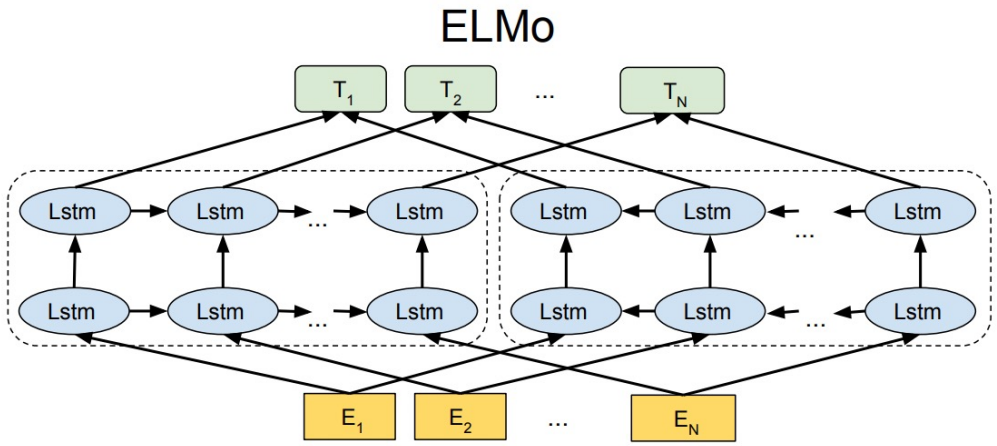
ELMo architecture

- Bidirectional LSTM Language Model



ELMo architecture

- Bidirectional LSTM Language Model



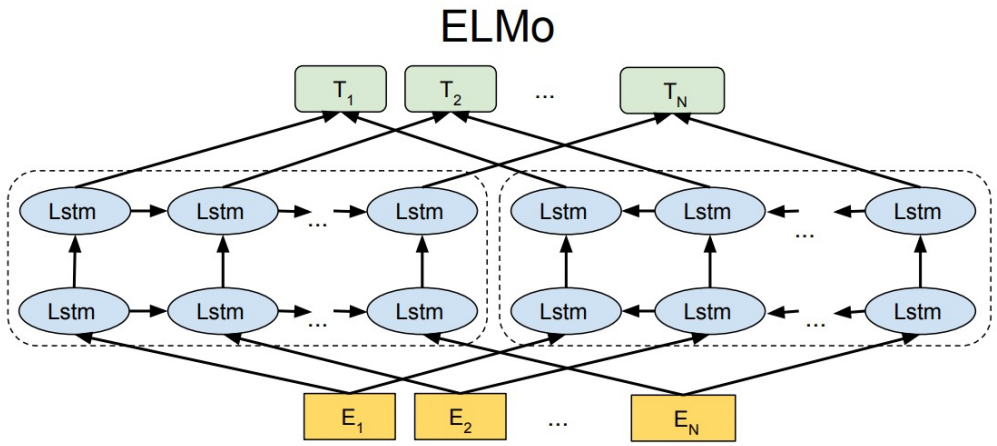
A forward LM computes the probability of sequence by modeling the probability of token t_k given left-side history

$$p(t_1, \dots, t_k) = \prod_{k=1}^{N} p(t_k | t_1, \dots, t_{k-1})$$

This is done by a Multi-Layer LSTM

ELMo architecture

- Bidirectional LSTM Language Model



A forward LM computes the probability of sequence by modeling the probability of token t_k given left-side history

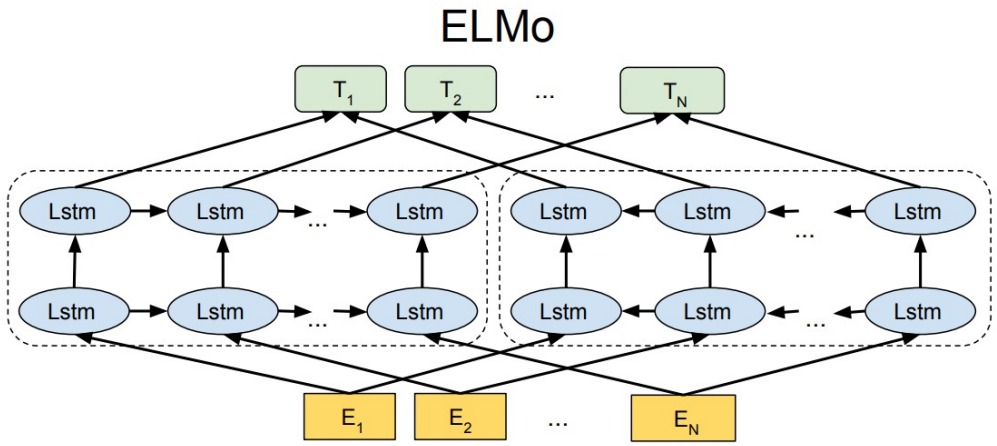
$$p(t_1, \dots, t_k) = \prod_{k=1}^{N} p(t_k | t_1, \dots, t_{k-1})$$

This is done by a Multi-Layer LSTM

The output of each layer j is a hidden representation $\vec{h}_{k,j}^{LM}$

ELMo architecture

- Bidirectional LSTM Language Model



A forward LM computes the probability of sequence by modeling the probability of token t_k given left-side history

$$p(t_1, \dots, t_k) = \prod_{k=1}^{N} p(t_k | t_1, \dots, t_{k-1})$$

This is done by a Multi-Layer LSTM

The output of each layer j is a hidden representation $\vec{h}_{k,j}^{LM}$

The top layer output $\vec{h}_{k,L}^{LM}$ is used to predict the next word

ELMo architecture

- Bidirectional LSTM Language Model

A forward LM computes the probability of sequence by modeling the probability of token t_k given left-side history

$$p(t_1, \dots, t_k) = \prod_{k=1}^{N} p(t_k | t_1, \dots, t_{k-1})$$

The output of each layer j is a hidden representation $\vec{h}_{k,j}^{LM}$

The top layer output $\vec{h}_{k,L}^{LM}$ is used to predict the **next** word

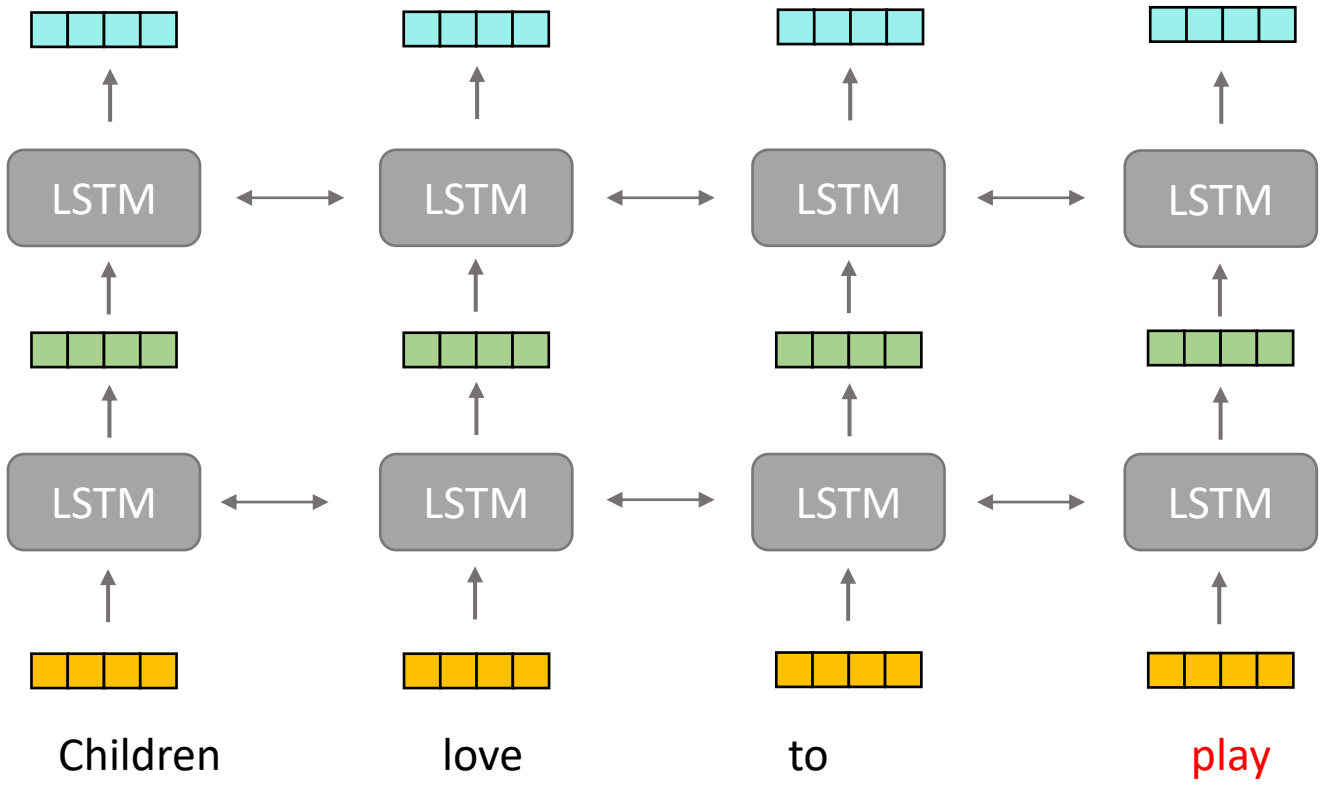
A backward LM computes the probability of sequence by modeling the probability of token t_k given right-side history

$$p(t_1, \dots, t_k) = \prod_{k=1}^{N} p(t_k | t_{k+1}, \dots, t_N)$$

The output of each layer j is a hidden representation $\overleftarrow{h}_{k,j}^{LM}$

The top layer output $\overleftarrow{h}_{k,j}^{LM}$ is used to predict the **previous** word

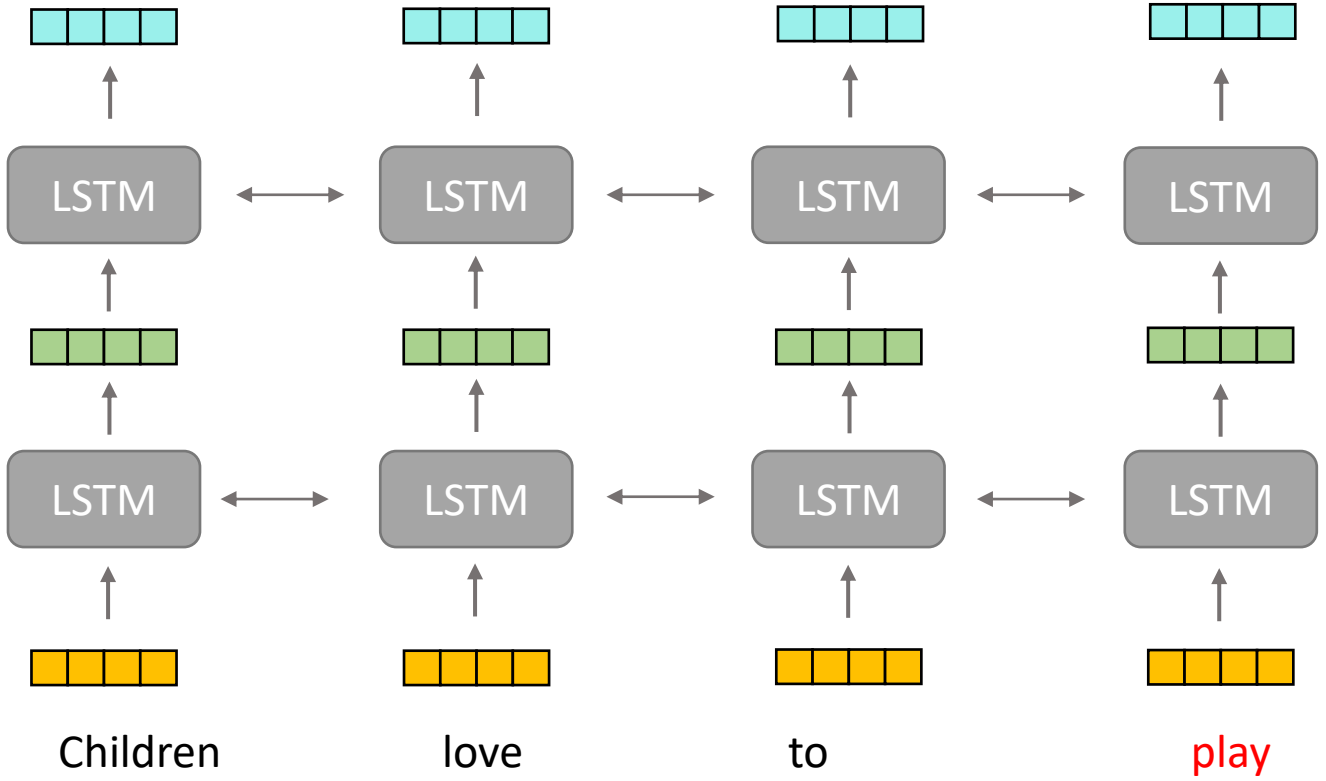
ELMo - Representations



ELMo - Representations

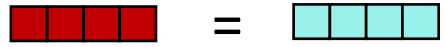
ELMo word embedding

?? We want a single vector for each word

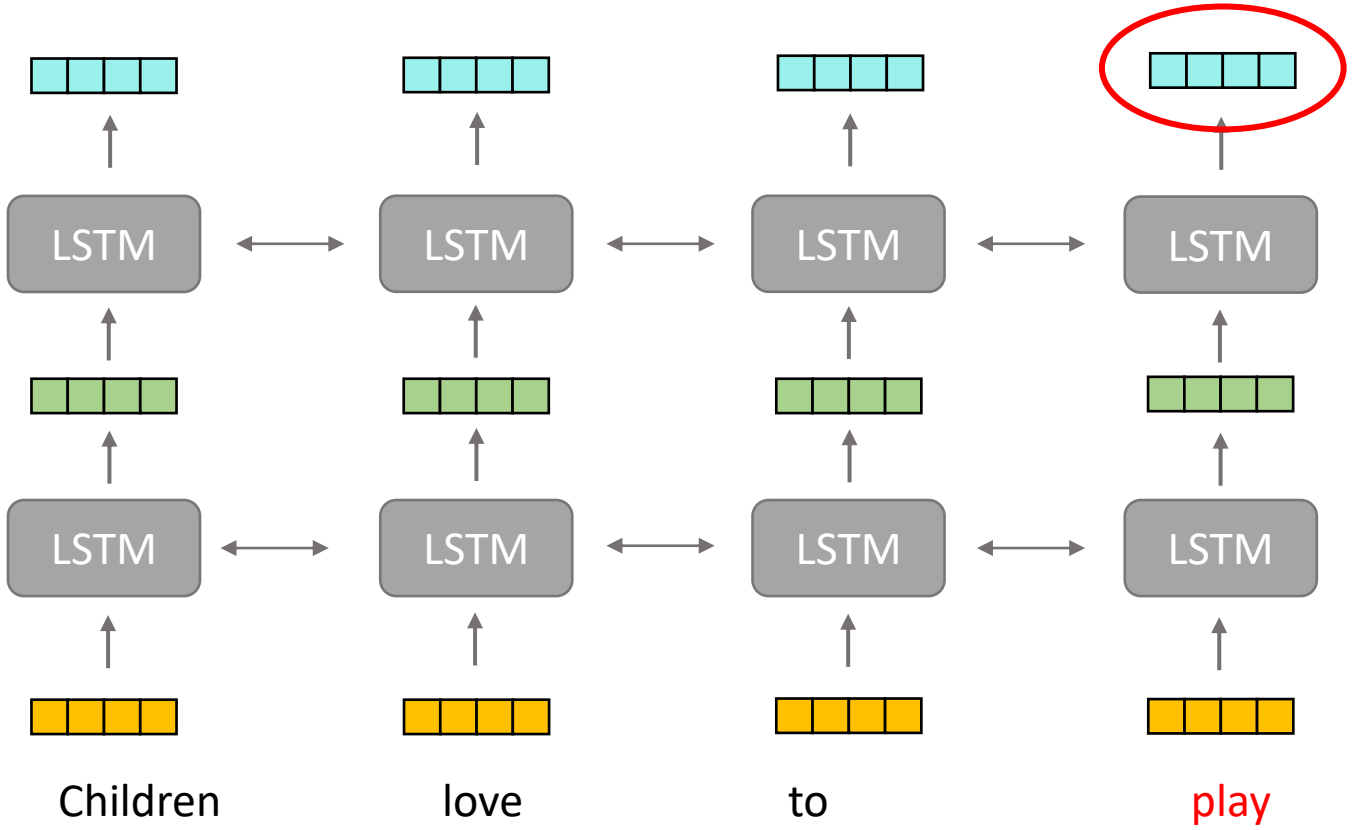


ELMo - Representations

ELMo word embedding

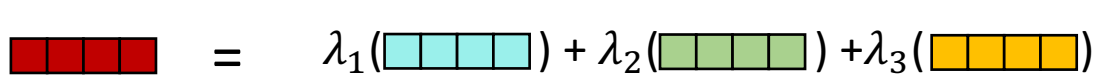


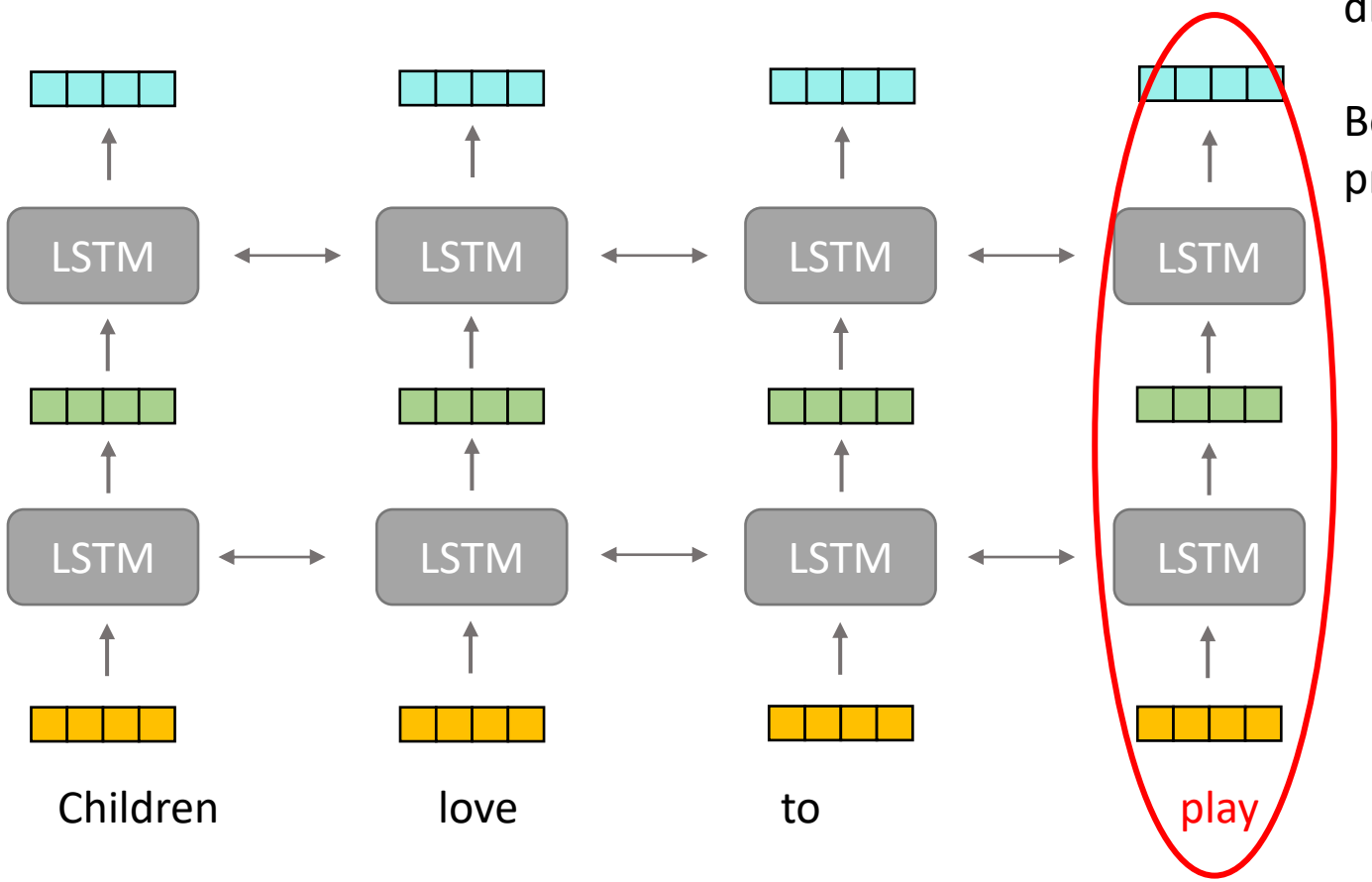
One option is to just take the last layer representation



ELMo - Representations

ELMo word embedding





Linearly mix representations from different layers

Best performance in practice

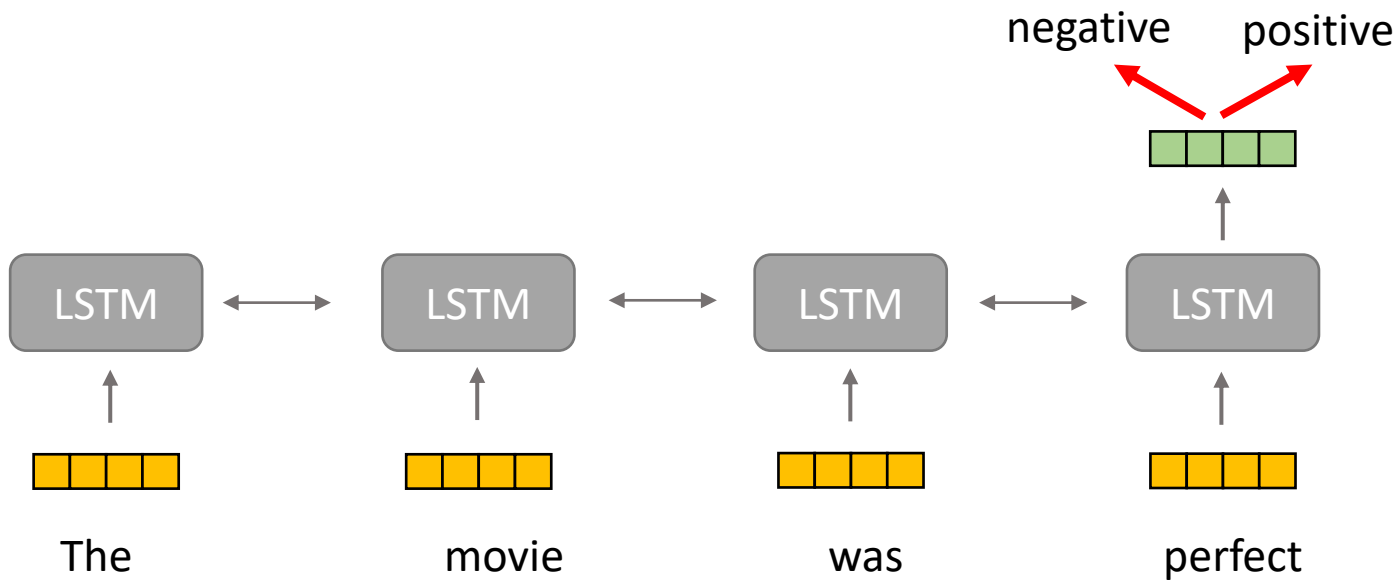
ELMo Usage

- At the time most model architectures were task specific
  - And most architectures were based on LSTMs/RNNs or CNNs

E.g., sentiment classification task

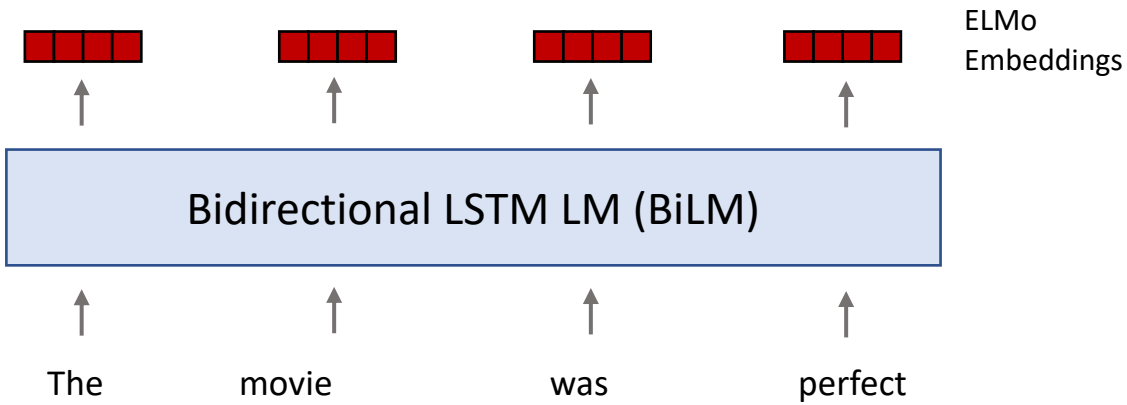
The movie was boring → negative

Storyline was perfect → positive



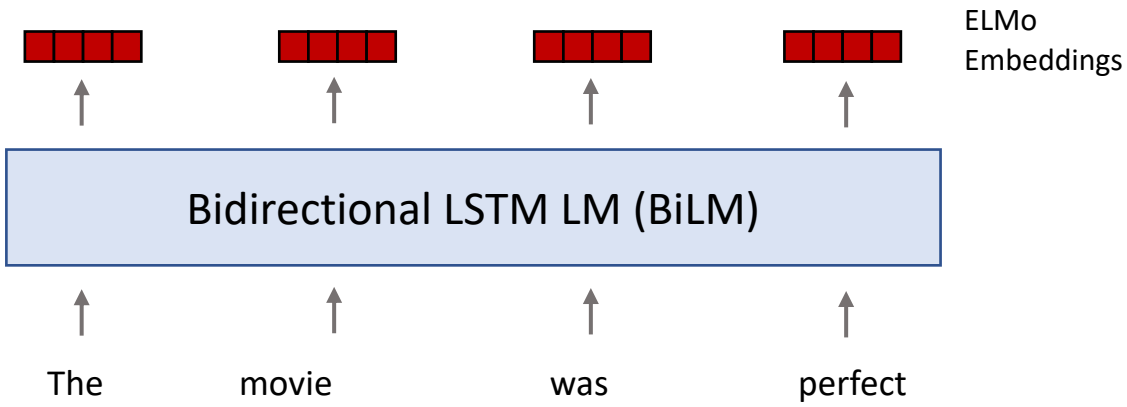
ELMo Usage

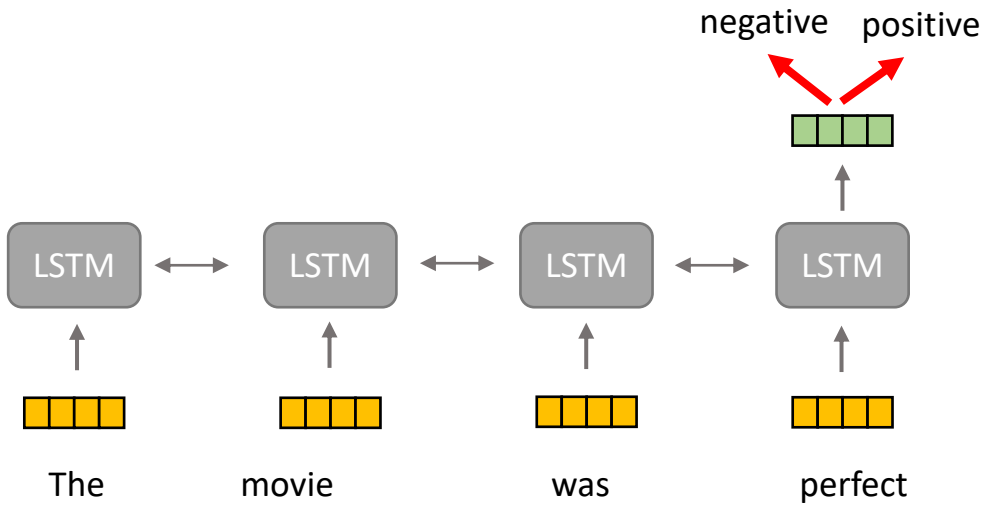
- Run the input through the pretrained Bidirectional LSTM Language Model to get ELMo embeddings



ELMo Usage

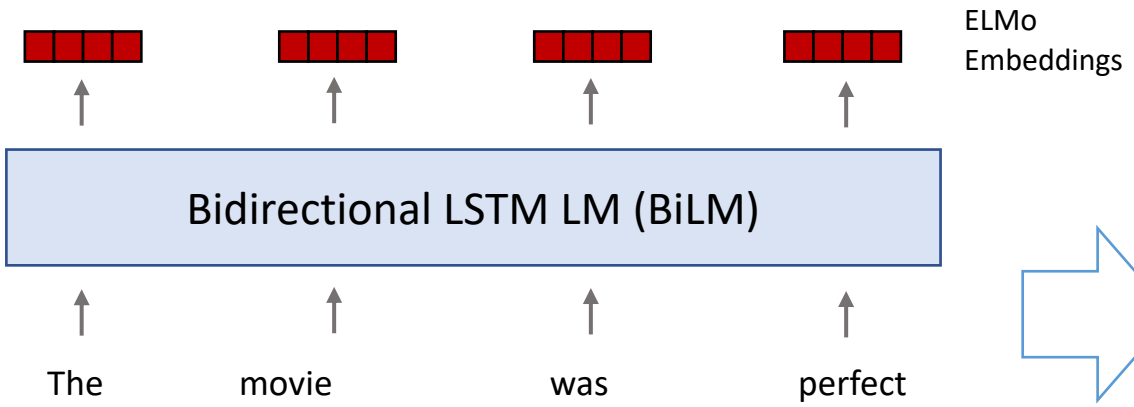
- Simply concatenate input vectors with ELMo vectors
- To use ELMo we freeze the weights of the BiLM
  - Only finetune the λ parameters
- Fine-tune only the task-specific model (e.g., task LSTM)

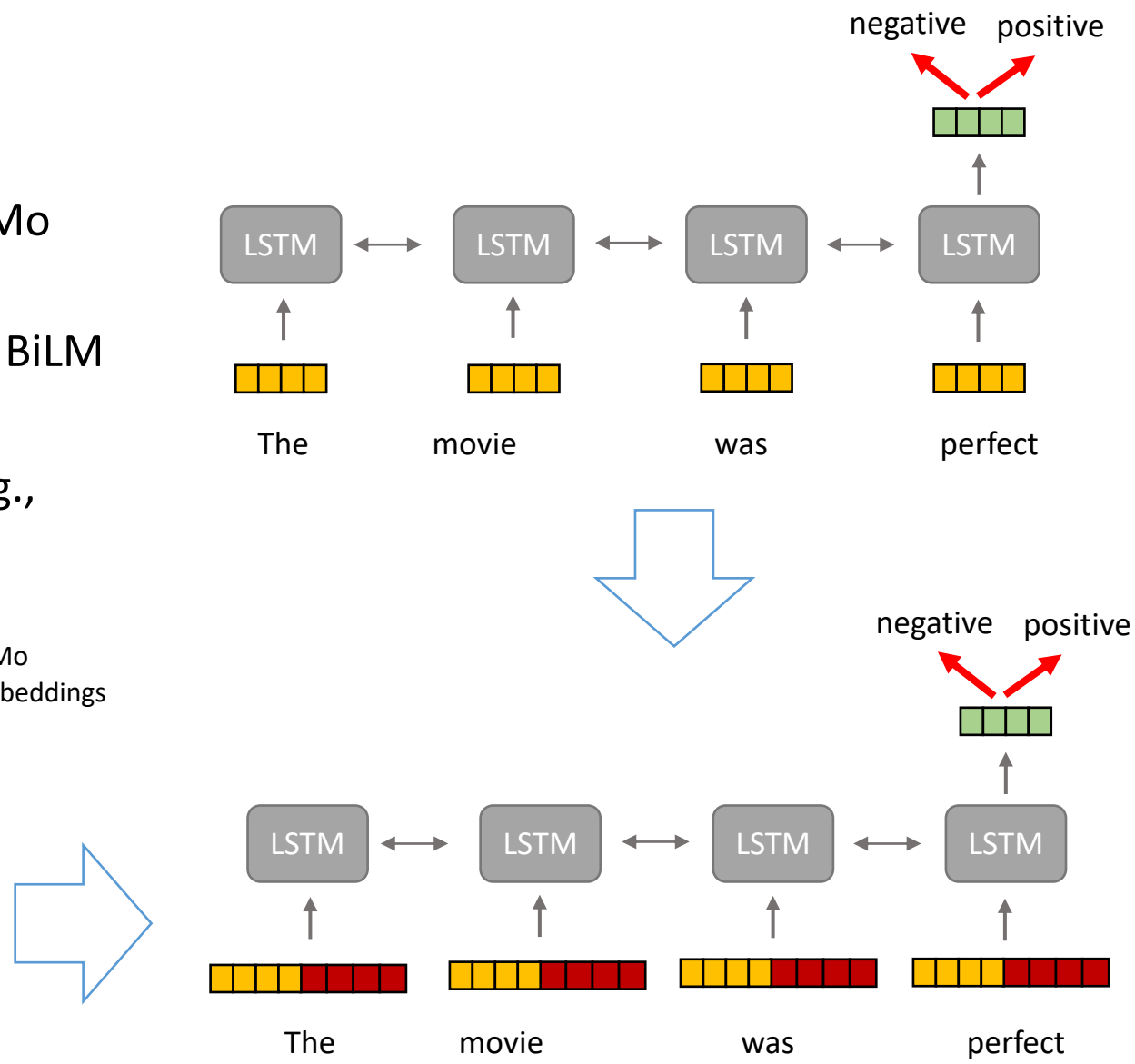




ELMo Usage

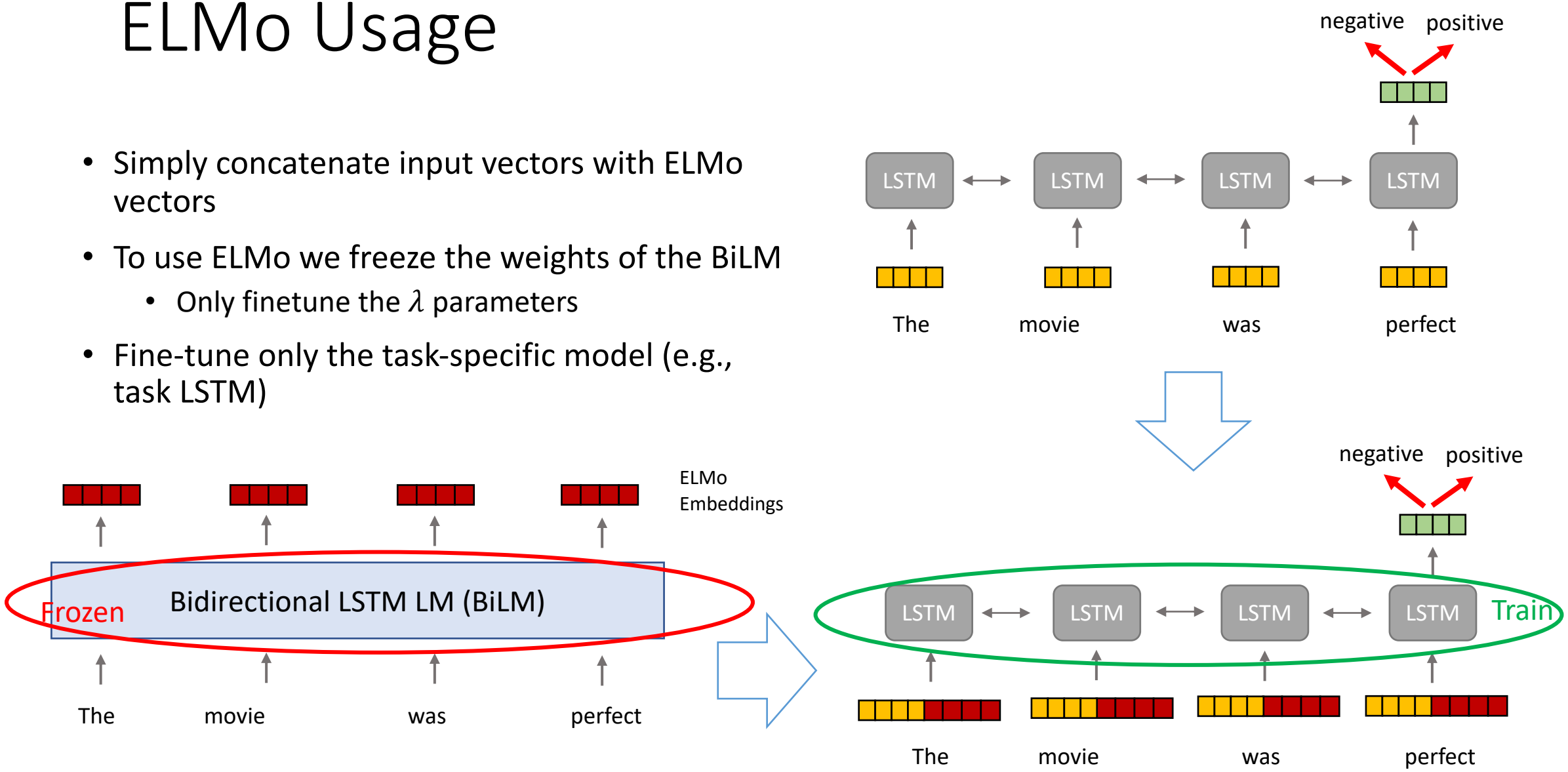
- Simply concatenate input vectors with ELMo vectors
- To use ELMo we freeze the weights of the BiLM
  - Only finetune the λ parameters
- Fine-tune only the task-specific model (e.g., task LSTM)





ELMo Usage

- Simply concatenate input vectors with ELMo vectors
- To use ELMo we freeze the weights of the BiLM
  - Only finetune the λ parameters
- Fine-tune only the task-specific model (e.g., task LSTM)



ELMo implication and results

- When ELMo representations added to downstream tasks they resulted in consistent improvements

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%

ELMo implication and results

	Source	Nearest Neighbors
GloVe	play	playing, game, games, played, players, plays, player, Play, football, multiplayer
biLM	Chico Ruiz made a spectacular play on Alusik 's grounder {...}	Kieffer , the only junior in the group , was commended for his ability to hit in the clutch , as well as his all-round excellent play .
	Olivia De Havilland signed to do a Broadway play for Garson {...}	{...} they were actors who had been handed fat roles in a successful play , and had talent enough to fill the roles competently , with nice understatement .

Questions?

ULMFit: Universal language model fine-tuning

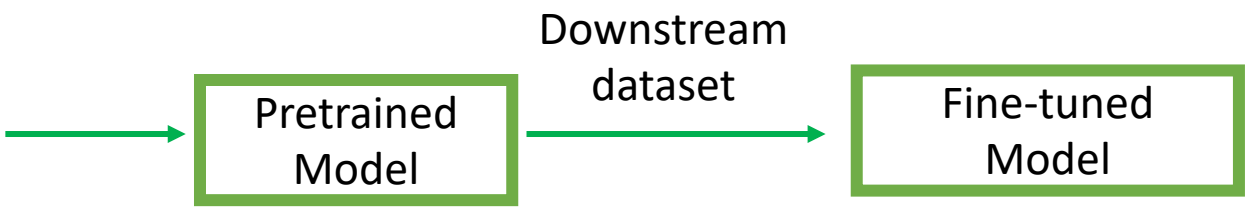
Howard and Ruder, 2018

ULMFit

- Universal Language Model Fine-tuning for Text Classification

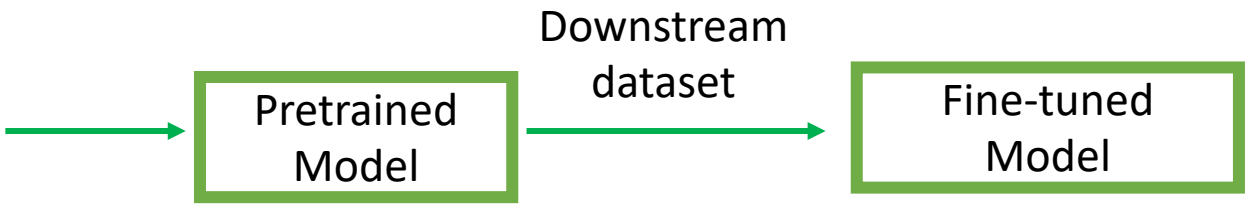
ULMFit

- Takes inspiration from work in computer vision
  - Computer vision models at the time were not trained from scratch



ULMFit

- Takes inspiration from work in computer vision
  - Computer vision models at the time were not trained from scratch

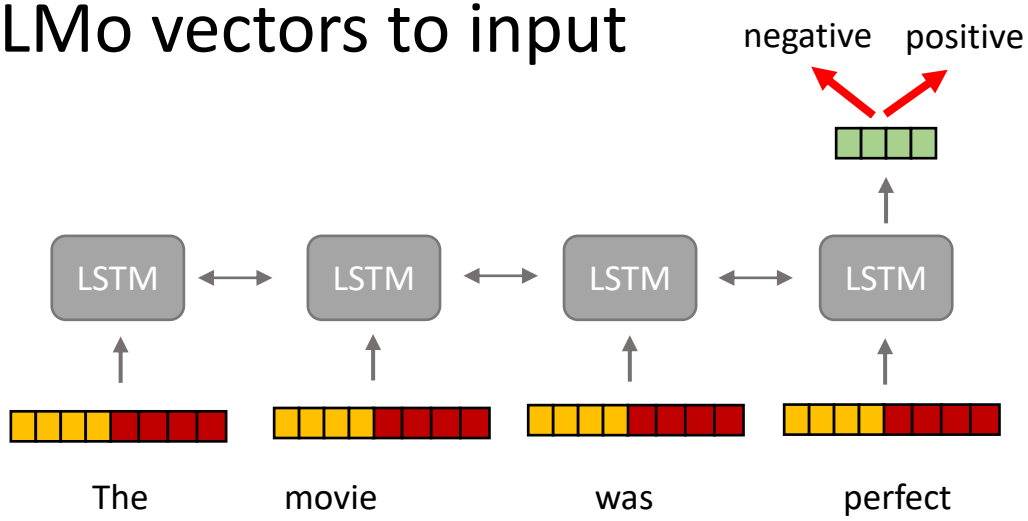


- Most existing NLP models at the time were being trained from scratch for downstream tasks
- **Can we apply the same pretraining/finetuning paradigm to NLP?**

Problem with approaches like ELMO

- One needs to train the task-specific model weights from scratch
- Concatenate the ELMo vectors to input

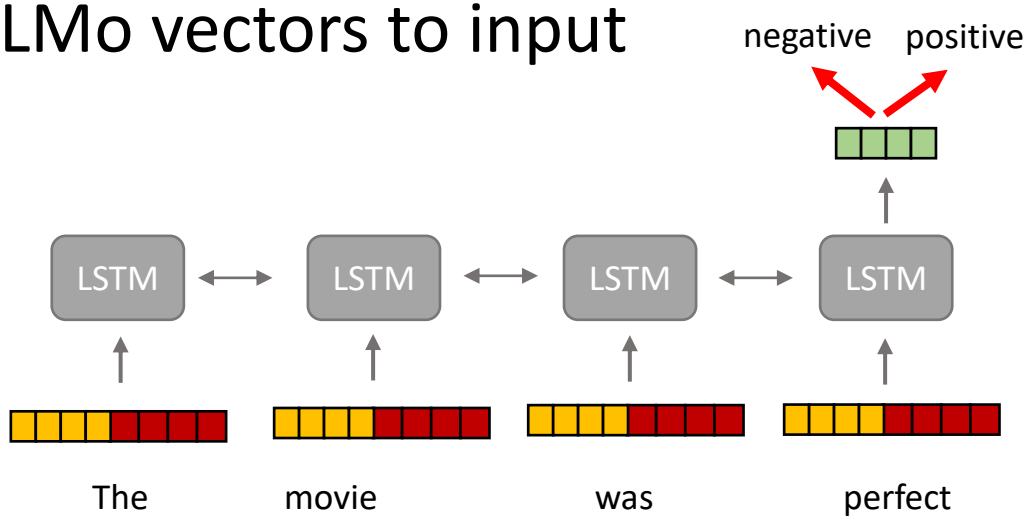
Task model parameters initialized from scratch



Problem with approaches like ELMO

- One needs to train the task-specific model weights from scratch
- Concatenate the ELMo vectors to input

Task model parameters initialized from scratch



- Is there a way to use a “universal” model for all tasks
  - Without the need for initializing parameters from scratch?

ULMFit

- A Universal model: Works across different “classification” tasks
  - With varying document size, and label type

ULMFit

- A Universal model: Works across different “classification” tasks
  - With varying document size, and label type
- Uses a single architecture and training process

ULMFit

- A Universal model: Works across different “classification” tasks
  - With varying document size, and label type
- Uses a single architecture and training process
- Doesn’t require additional feature engineering

ULMFit – Three main stages

Language Model Pretraining

Target Task LM Finetuning

Target Task Classifier Finetuning

1- Language Model Pretraining

- Pretrain the model on a general language corpus
- Allows the model to generalize better
- Specially a useful step for tasks with small training data

2- Target Task LM Finetuning

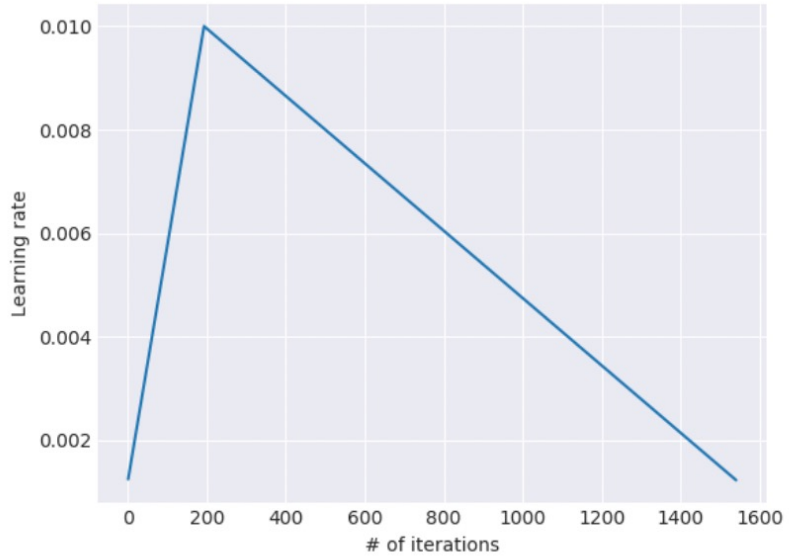
- Finetune the LM on the task data
- Still the language modeling objective
- Two main ideas that made this work:
  - Discriminative finetuning
  - A new type of learning rate scheduler

Discriminative finetuning

- Instead of single learning rate for all layers, tune each layer with a different learning rate
  - In practice they use a learning rate η for last layer and $\eta/2.6$ for lower layers
  - The goal is to prevent the base pretrained model to change too much, resulting in more robust finetuning

Learning rate scheduler

- Linear warmup with linear decay
  - First gradually increase the learning rate (warmup step)
    - Prevent the model to change too much too quickly
  - Then gradually decrease the learning rate
    - To allow more fine-grained updates to the model as training progresses
  - This was key point in making the finetuning work well

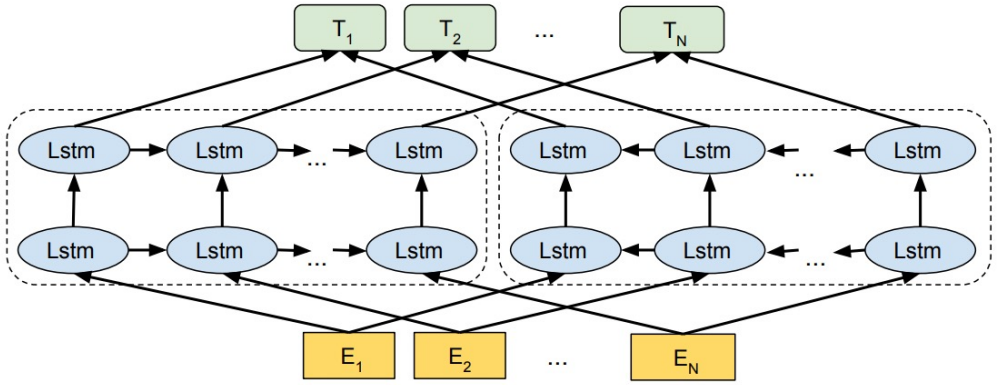


3- Target Task Classifier Finetuning

- Augment the LM with a linear classification layer
  - These are task specific classifiers that are initialized from scratch
  - They are a small subset of the whole network parameters

ULMFit – architecture

- They use bidirectional LSTM as their base model architecture



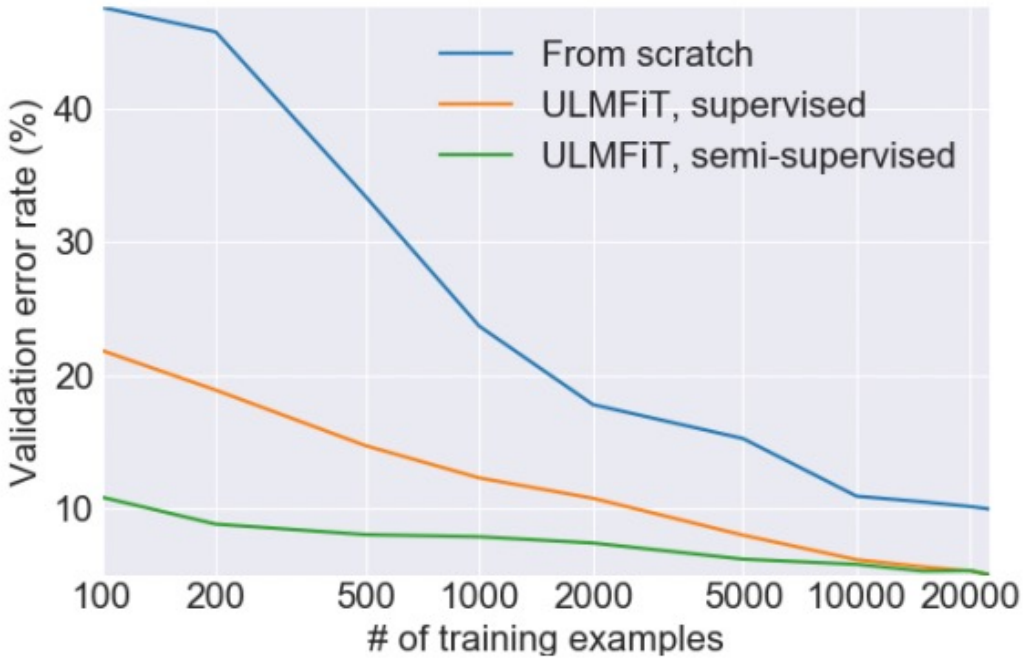
Results

	Model	Test		Model	Test
IMDb	CoVe (McCann et al., 2017)	8.2	TREC-6	CoVe (McCann et al., 2017)	4.2
	oh-LSTM (Johnson and Zhang, 2016)	5.9		TBCNN (Mou et al., 2015)	4.0
	Virtual (Miyato et al., 2016)	5.9		LSTM-CNN (Zhou et al., 2016)	3.9
	ULMFiT (ours)	**4.6**		ULMFiT (ours)	**3.6**

	AG	DBpedia	Yelp-bi	Yelp-full
Char-level CNN (Zhang et al., 2015)	9.51	1.55	4.88	37.95
CNN (Johnson and Zhang, 2016)	6.57	0.84	2.90	32.39
DPCNN (Johnson and Zhang, 2017)	6.87	0.88	2.64	30.58
ULMFiT (ours)	**5.01**	**0.80**	**2.16**	**29.98**

Test error rates (%) , lower is better

Impact of different stages



Impact of the LR schedule and discriminative finetuning

LM fine-tuning	IMDb	TREC-6	AG
No LM fine-tuning	6.99	6.38	6.09
Full	5.86	6.54	5.61
Full + discr	5.55	6.36	5.47
Full + discr + stlr	**5.00**	**5.69**	**5.38**

ULMFit

- One of the first attempts towards universal language models that can be used for a variety of tasks
- Eliminated the need for starting major model parameters from scratch
- Technical contributions such as learning rate schedule was key to make this work

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

Background: OpenAI GPT (this is GPT1)

- GPT (Radford et al., 2018)
  - Showed that transformers can replace LSTMs for language modeling
  - They used a unidirectional left-to-right transformer
  - They applied they same principles of ULMFit
    - Pretraining and finetuning a single model (without the middle step of LM task finetuning)
    - They update all parameters in finetuning
  - A language model that supported generation tasks

BERT

- Argues that unidirectional training limits the power of LMs
  - Can result in decreased performance on tasks such as QA

BERT main ideas

1- Similar to UMLFit, have a universal LM that works for all tasks and that doesn't need initializing from scratch

2- Similar to GPT, replace LSTM with Transformer

3- Make the transformer bidirectional by introducing a novel language modeling task

4- Eliminate the intermediate step of LM **task** finetuning of ULMFit

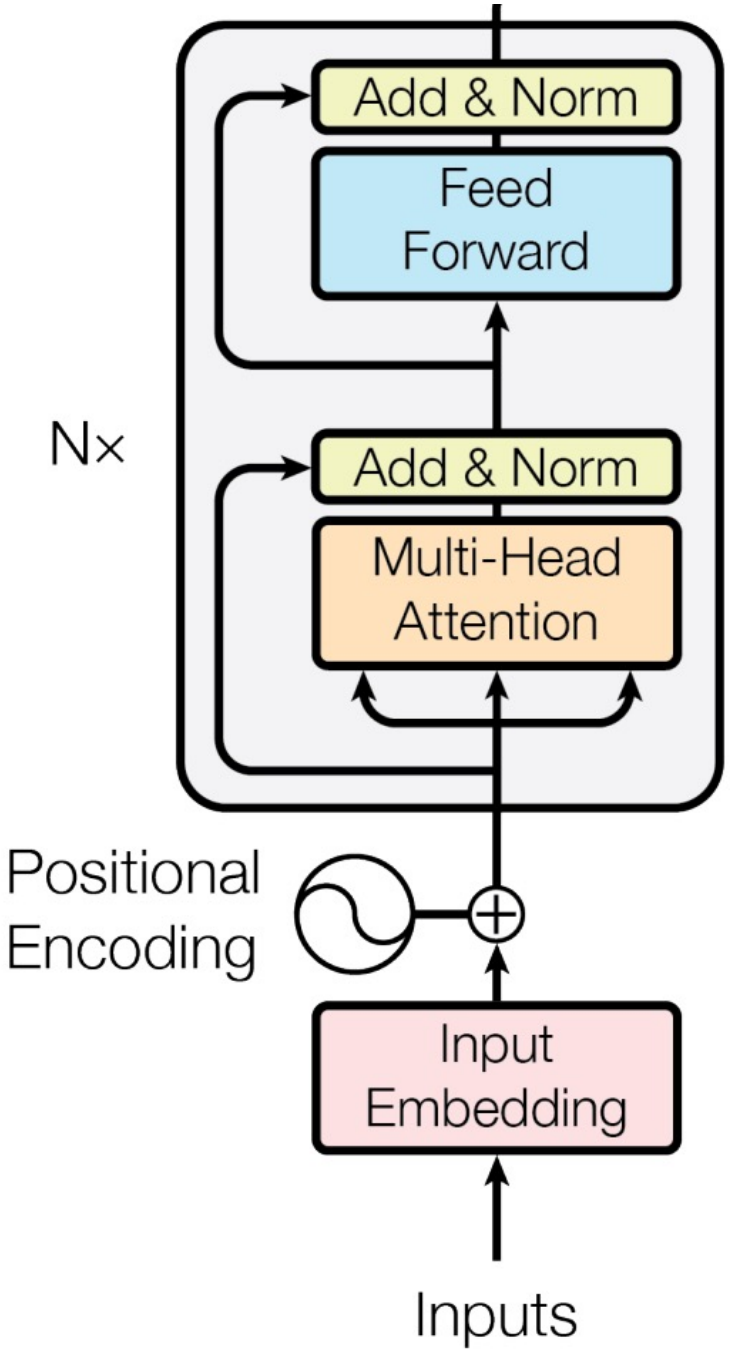
5- Follow tricks that were used in ULMFit in making finetuning robust

6- Scale up the pretraining

BERT model architecture

- BERT uses Transformer
  - Two sizes: Base and Large

	Base	Large
Number of layers	12	24
Number of attention heads	12	16
Dimension of the input/output	768	1024
Total number of parameters	110M	340M



BERT input/output format

- Vocabulary
  - Size 30,000
  - Using WordPieces to tokenize the input text
    - If a token is not in the vocabulary, it will be broken down to smaller “word-pieces” that are in vocabulary

Fig from: https://ai.googleblog.com/2021/12/a-fast-wordpiece-tokenization-system.html

BERT input format

- BERT was designed to address a variety of tasks
  - Single sentence classification (e.g., sentiment classification)
  - Sentence pair classification

Sentence 1	Sentence 2	Label (entail, contradict, neutral)
A man inspects the uniform of a figure	The man is sleeping.	contradict

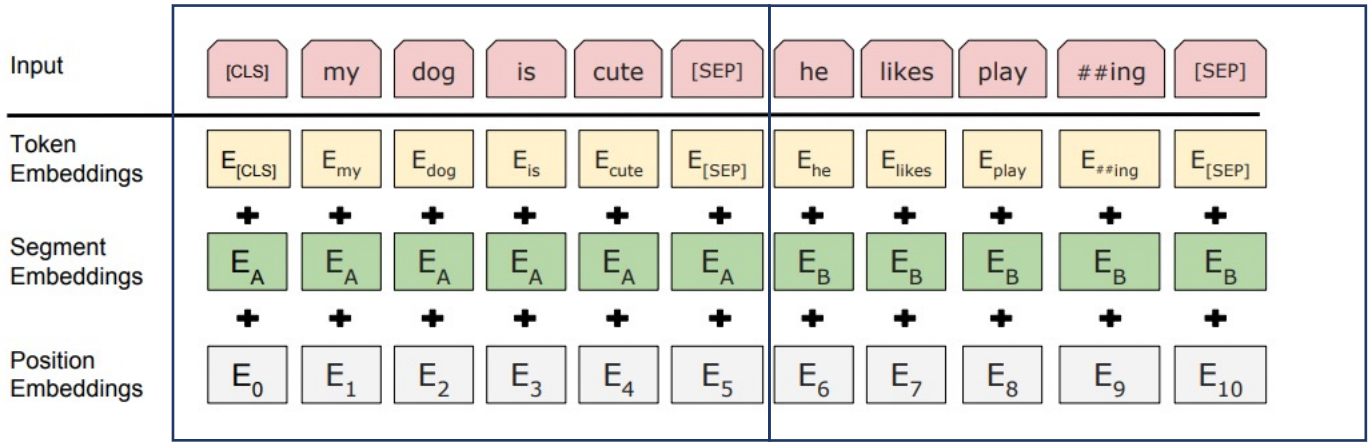
  - Question answering

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under **gravity**. The main forms of precipitation include drizzle, rain, sleet, snow, **graupel** and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals **within a cloud**. Short, intense periods of rain in scattered locations are called "showers".

What causes precipitation to fall?
gravity

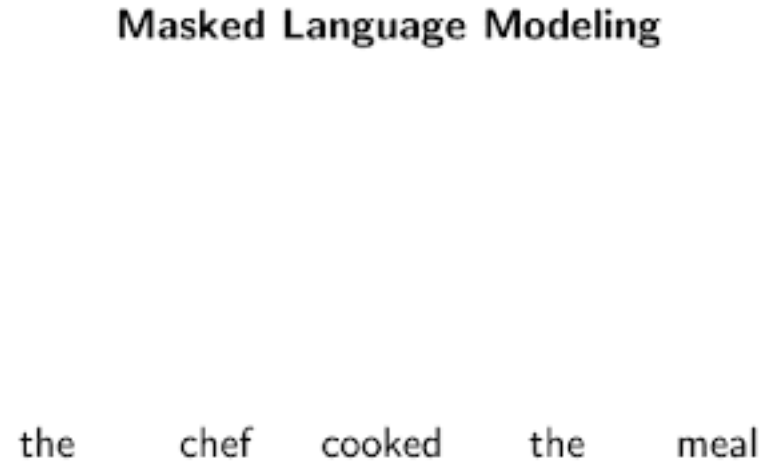
BERT input format

- Always append a special **[CLS]** token before the sentence and append the sentence with a **[SEP]** token
- If there are two sequences use a special **[SEP]** tokens between them
- To let the model better handle 2 sentence tasks, BERT uses an additional segment embedding that is added to the input



Pretraining objectives

- Masked Language Modeling
  - This is designed to train the deep bidirectional representation
  - ~15% of the input tokens are masked



Pretraining objectives

- Masking rate (15%):
  - If mask too little: computationally expensive
  - Too much masking: not enough info for the model to recover the masked tokens
- How are these chosen? Uniformly sampled
  - Later work shows that more principled masking could benefit downstream task performance and result in faster training

  PMI Masking (Levine et al., 2021) https://arxiv.org/pdf/2010.01825.pdf
  SpanBERT (Joshi et al., 2020) https://arxiv.org/pdf/1907.10529.pdf

Details of their MLM masking

- Select 15% of the tokens randomly
  - Take the position corresponding to these tokens
  - 80% of the time replace the word at that position with **[MASK]**
  - 10% of the time the word is replaced with a random word in vocab
  - 10% of the time they keep it unchanged
  - Use the original word at that position as target of the prediction

Details of their MLM masking

- Select 15% of the tokens randomly
  - Take the position corresponding to these tokens
  - 80% of the time replace the word at that position with **[MASK]**
  - 10% of the time the word is replaced with a random word in vocab
  - 10% of the time they keep it unchanged
  - Use the original word at that position as target of the prediction

- Why?
  - Pretraining - finetuning mismatch
    - [MASK] tokens are not present during finetuning

Details of their MLM masking

- Select 15% of the tokens randomly
  - Take the position corresponding to these tokens
  - 80% of the time replace the word at that position with **[MASK]**
  - 10% of the time the word is replaced with a random word in vocab
  - 10% of the time they keep it unchanged
  - Use the original word at that position as target of the prediction

- Why?

Second pretraining objective

- Next Sentence Prediction
  - Take a text that has two sentences
  - Break down to two
  - 50% of the time replace the second sentence with a random sentence
  - Ask the model to predict if the sentences are naturally after each other or not

[CLS] the cat sat on the mat [SEP] It looked very cozy [SEP] → **YES**

[CLS] the cat sat on the mat [SEP] Tomorrow is 4th of July. [SEP] → **NO**

Details of pretraining

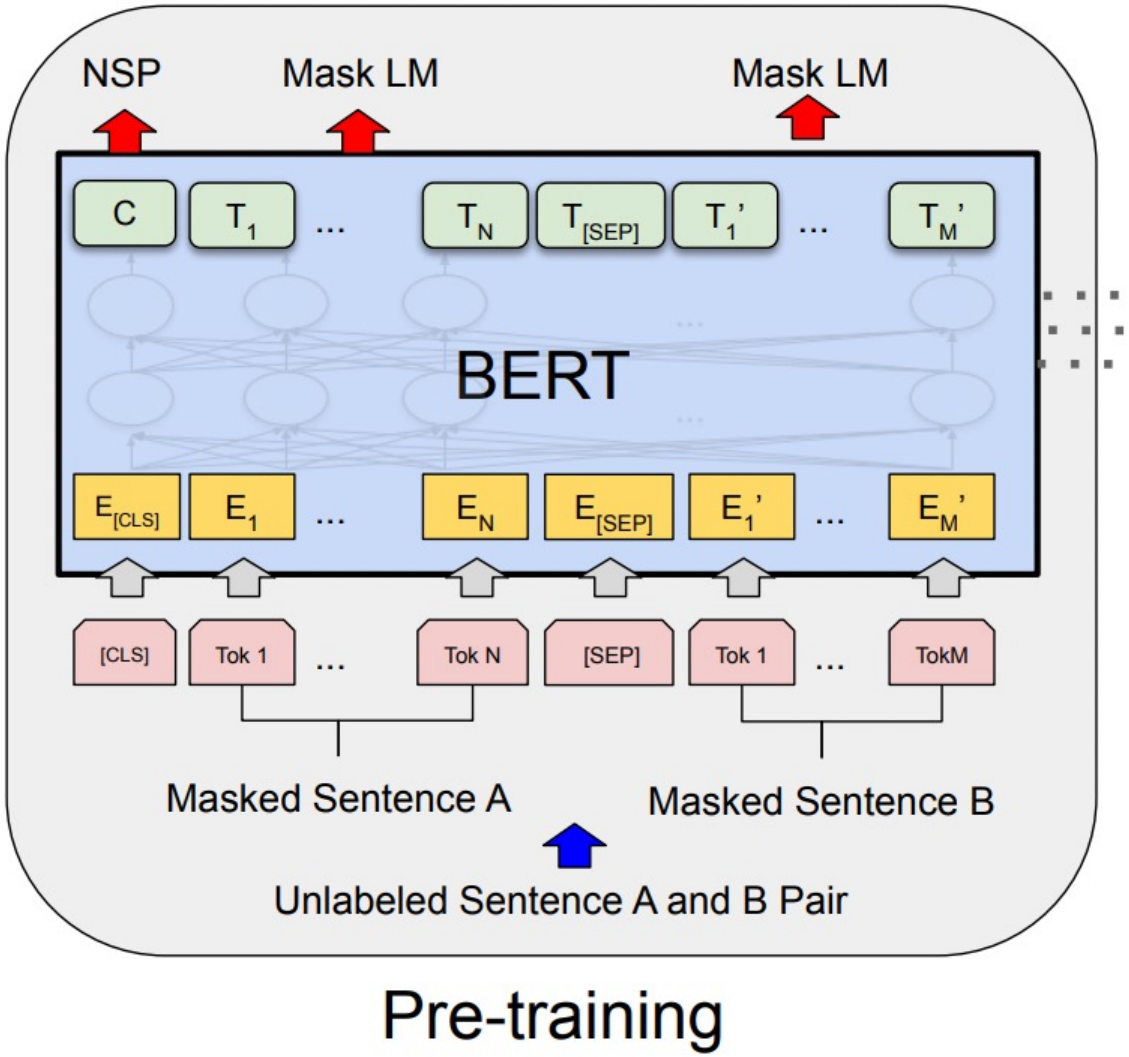
- Larger pretraining data size compared with prior work
  - Books corpus (~0.8B tokens)
    - OpenAI GPT1 was trained on this only
  - Wikipedia (2.5B tokens)

- Sequence length:
  - 512 tokens

- Trained for 1M steps, batch size 256
  - Batch size in terms of tokens: 256 ×512 = 128,000 tokens
- Training time:
  - base model: 16 TPU chips for 4 days training cost today using cloud tpu: ~$1.3K
  - Large model: 64 TPU chips for 4 days training cost today using cloud tpu : ~$5K

Pretraining (summary)

- The two pretraining tasks are performed jointly

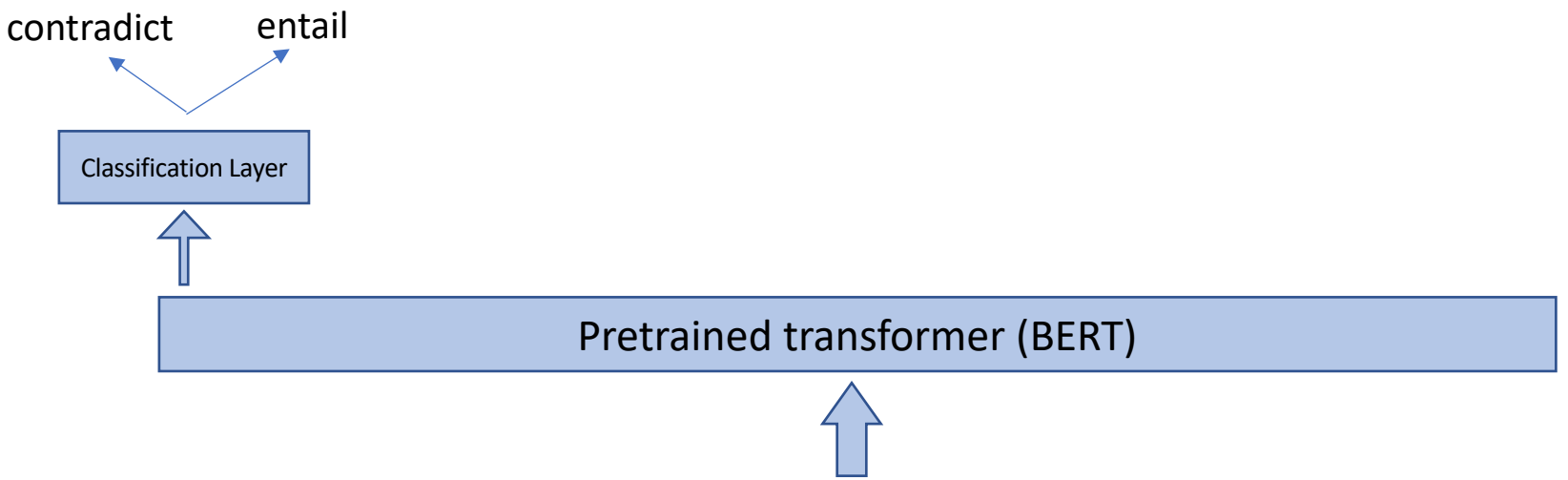


Finetuning

- BERT follows ULMFit
  - Pretraining allows having a single model that can be finetuned quickly for different tasks
  - Expands to a wider range of classification tasks
    - NLI, QA, Sequence tagging

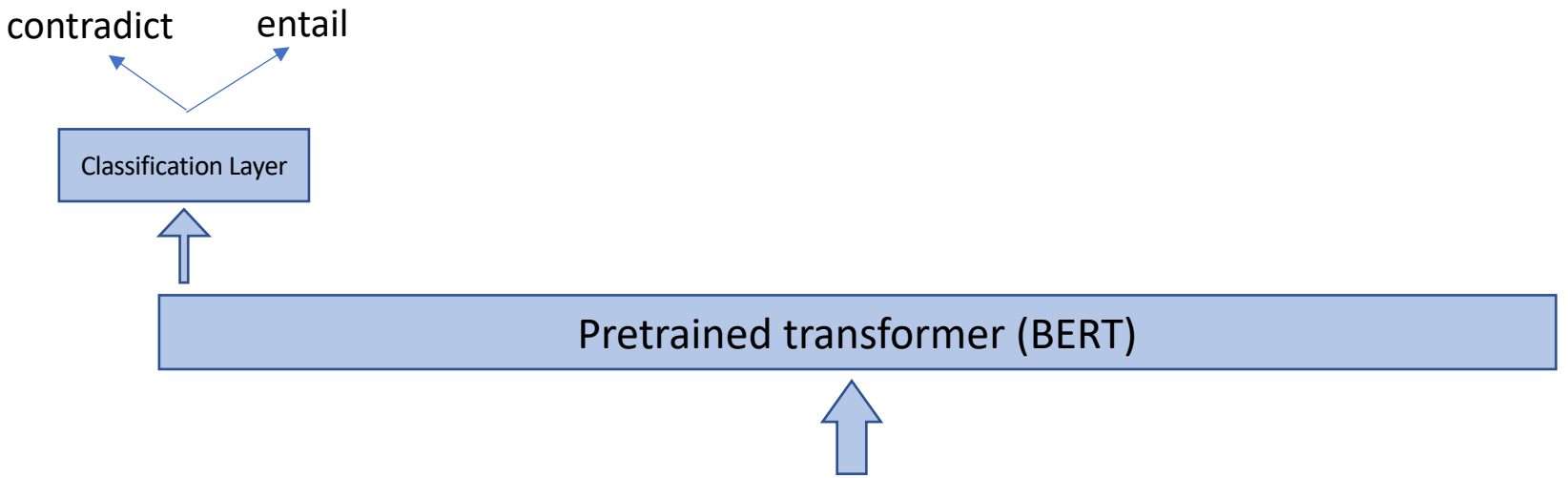
Finetuning

- Classification (single sentence or sentence pair)
  - Take the representation at the [CLS] token
  - Perform classification through a linear layer (also called **classification head**)
    - Similar to ULMFit
    - Initialized from scratch



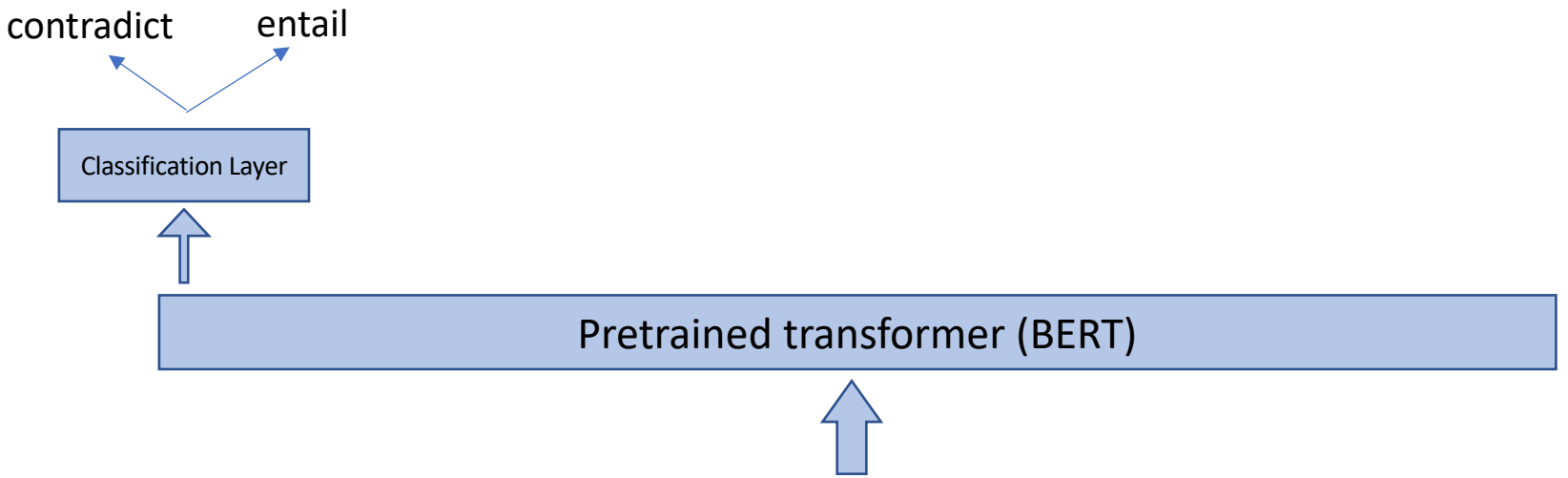
Finetuning

- Classification (single sentence or sentence pair)
  - Take the representation at the [CLS] token
  - Perform classification through a linear layer (also called **classification head**)
    - Similar to ULMFit
    - Initialized from scratch
  - **How many new parameters does this introduce?**



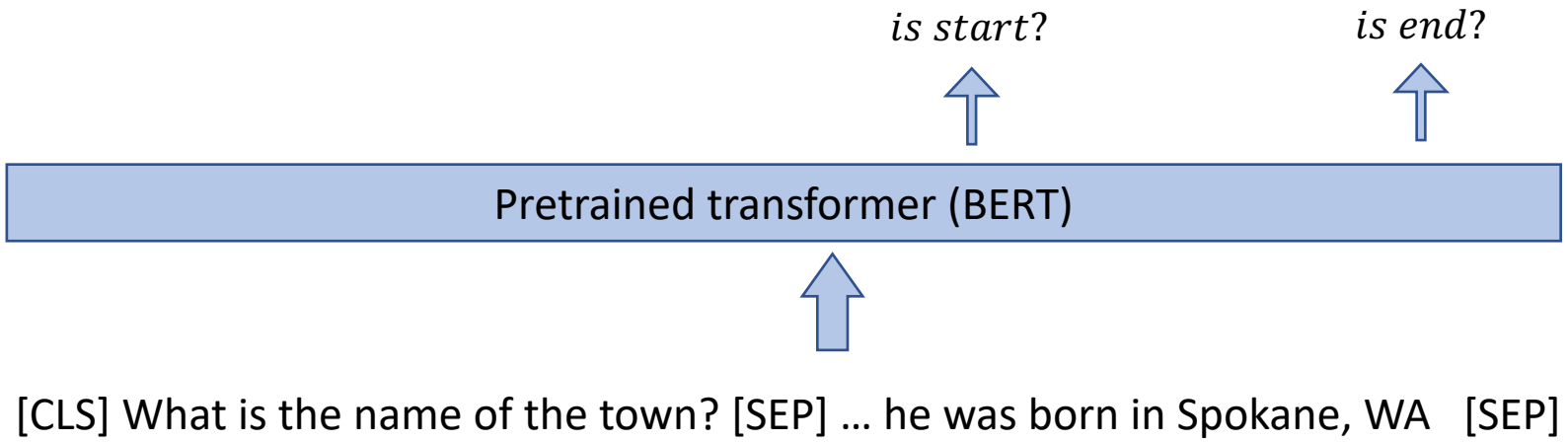
Finetuning

- Classification (single sentence or sentence pair)
  - Take the representation at the [CLS] token
  - Perform classification through a linear layer (also called **classification head**)
    - Similar to ULMFit
    - Initialized from scratch
  - **How many new parameters does this introduce? (hidden size × num labels)**



Finetuning

- Question answering
  - Classify if each span can be start or end of the answer span



Some technical notes

- The pretraining learning rate was much larger $1e-4$
  - Using the same learning rate schedule as ULMFit
    - Linear warmup for 10K steps then linear decay
- Finetuning learning rate is typically $1e-5\ to\ 5e-5$
  - Finetuning also usually works better with learning rate warmup

Results

- GLUE: Sentence/Sentence pair classification benchmark

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**

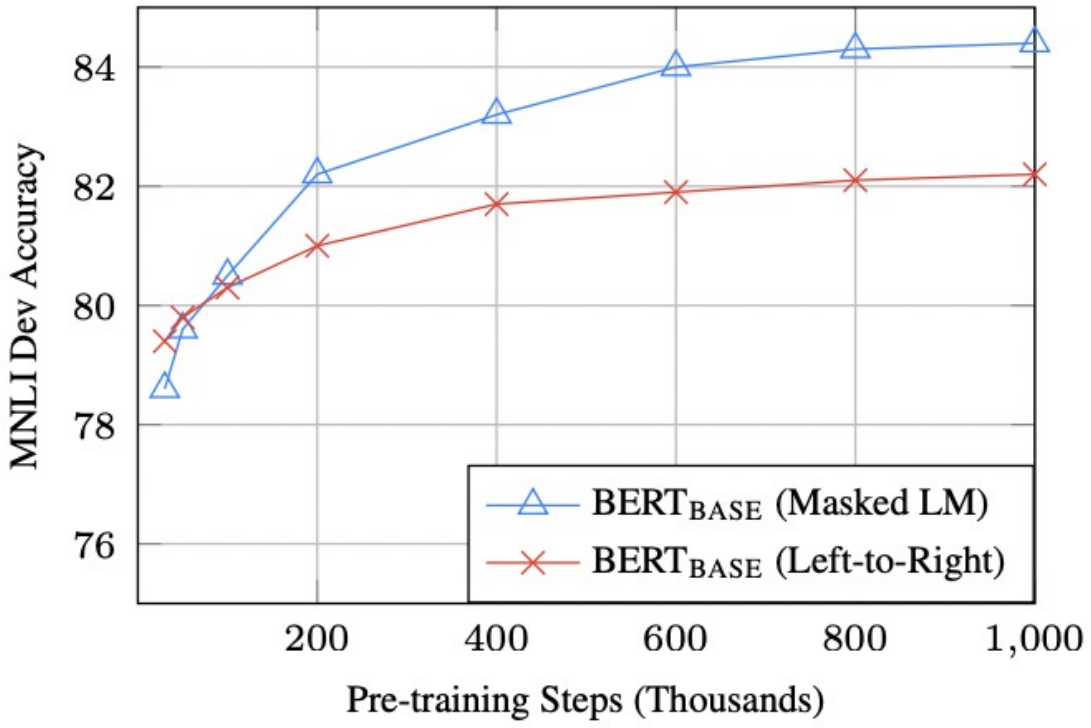
Results - QA

System	Dev		Test	
	EM	F1	EM	F1
Top Leaderboard Systems (Dec 10th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
Published				
BiDAF+ELMo (Single)	-	85.6	-	85.8
R.M. Reader (Ensemble)	81.2	87.9	82.3	88.5
Ours				
$\text{BERT}_{\text{BASE}}$ (Single)	80.8	88.5	-	-
$\text{BERT}_{\text{LARGE}}$ (Single)	84.1	90.9	-	-
$\text{BERT}_{\text{LARGE}}$ (Ensemble)	85.8	91.8	-	-
$\text{BERT}_{\text{LARGE}}$ (Sgl.+TriviaQA)	**84.2**	**91.1**	**85.1**	**91.8**
$\text{BERT}_{\text{LARGE}}$ (Ens.+TriviaQA)	**86.2**	**92.2**	**87.4**	**93.2**

System	Dev		Test	
	EM	F1	EM	F1
Top Leaderboard Systems (Dec 10th, 2018)				
Human	86.3	89.0	86.9	89.5
#1 Single - MIR-MRC (F-Net)	-	-	74.8	78.0
#2 Single - nlnet	-	-	74.2	77.1
Published				
unet (Ensemble)	-	-	71.4	74.9
SLQA+ (Single)	-		71.4	74.4
Ours				
$\text{BERT}_{\text{LARGE}}$ (Single)	78.7	81.9	80.0	83.1

How much bidirectional context matters?

- MLM converges slower
- But it outperforms left-to-right LM



Major impacts of BERT

- In order to have state-of-the-art performance on different tasks, there is no need for coming up with a novel model architecture
  - End of task model architecture engineering

- As we will see in next lectures, larger scales, better approaches for language modeling and transfer learning are the key for future performance improvements

Questions?

Logistics - FQA

- How many papers in total do I need to present throughout the semester?

Logistics - FQA

- How many papers in total do I need to present throughout the semester?
  - Depends on the number of students that will end up taking the class
    - But it would be maximum 4 papers for the entire semester for each student
    - Can be less if enrollment is high

Logistics - FAQ

- What if I am presenting but having trouble understanding some parts of the paper? Will I get penalized?

Logistics - FAQ

- What if I am presenting but having trouble understanding some parts of the paper? Will I get penalized?
  - You are not the author of the paper. It is okay if you don’t completely understand every detail!
  - We will try to understand the details in discussions
  - Also feel free to reach out to ask questions

Logistics - FAQ

- What is the expected outcome of the project? Do you expect conference submissions?

Logistics - FAQ

- What is the expected outcome of the project? Do you expect conference submissions?
  - Not at all. This is a class project and should have a more limited scope than a conference paper!
  - Usually, good class projects have a small, focused contribution or study a focused problem, task, or phenomena
  - For some cases, the project may show potential to become a conference submission
    - But it is up to your team if you are interested in making it conference worthy
  - Negative results are okay and won't be penalized. The important thing is to provide sufficient analysis that can explain the results

Next time

GPT-2, Language Models are Unsupervised Multi-task Learners
https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

T5: Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer
https://arxiv.org/pdf/1910.10683.pdf

Any volunteers?