Foundations for Natural Language Processing

Transfer learning

Ivan Titov
(with graphics/materials from Elena Voita)

School of informatics

Transfer learning

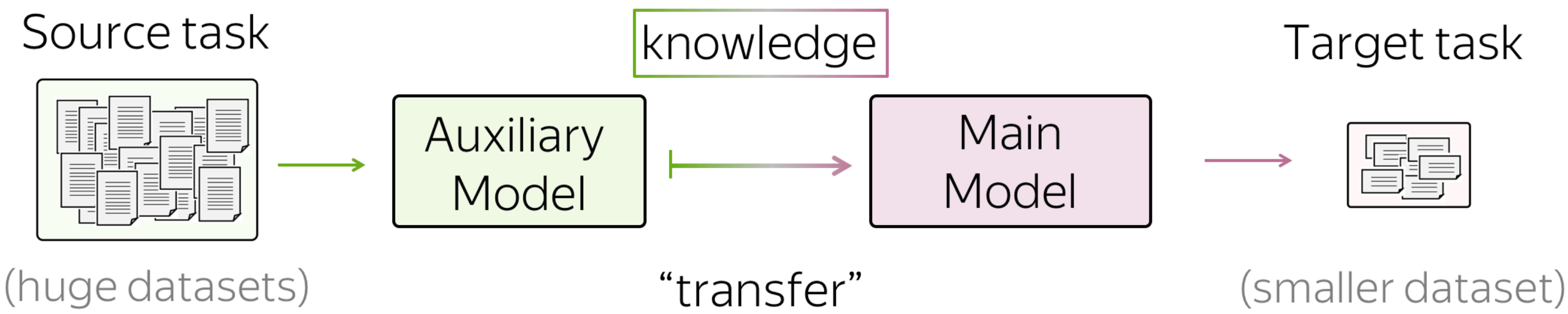
Typically, we do not have enough training data to estimate an accurate model on the data for the target task

Consider question answering:

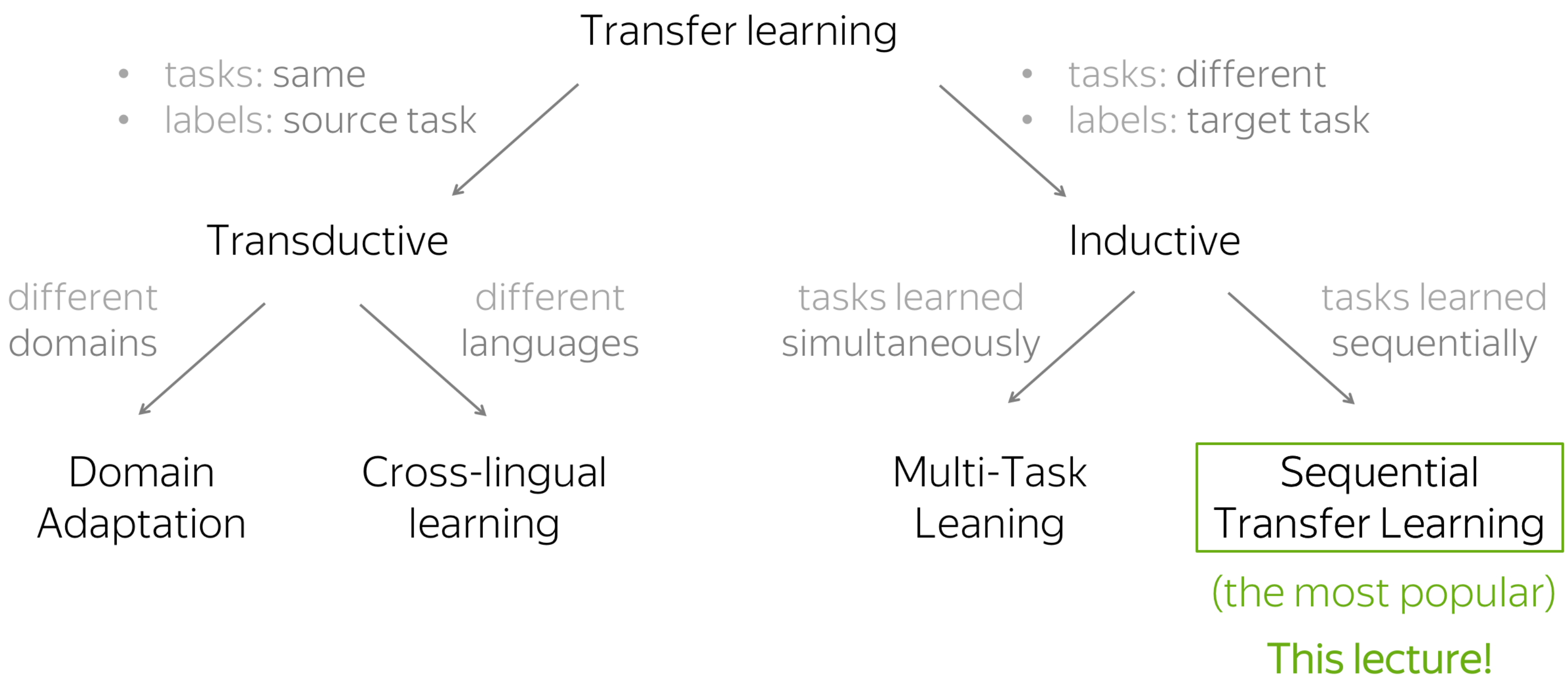
It is impossible to maintain a large and up-to-date collection of question-answer pairs for all possible questions, domains, languages, cultures…

How can we benefit from data for other tasks? (including tasks for which data occurs 'naturally' such as language modeling = predicting next word)

Transfer learning

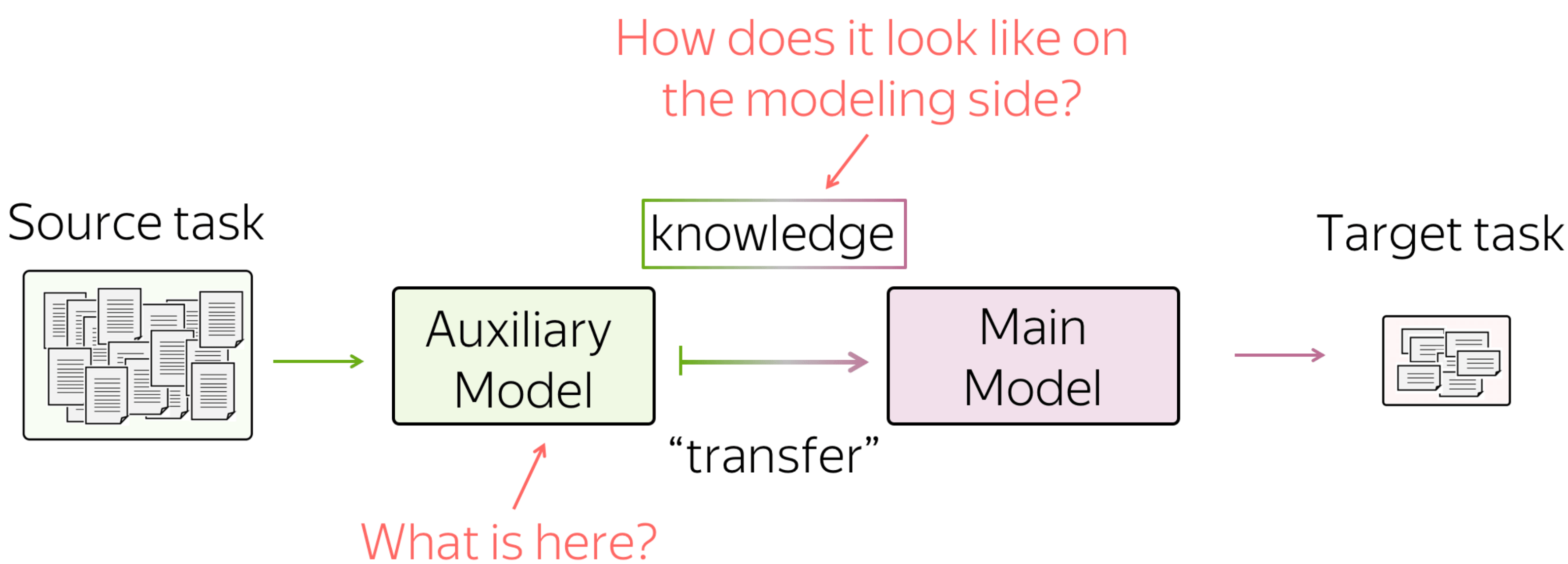


Transfer learning is a broad area



From Sebastian Ruder

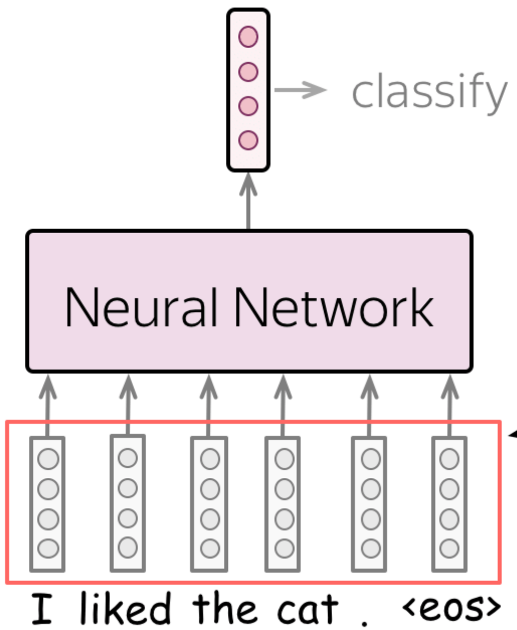
Transfer learning is a broad area



How do we define the auxiliary model?
How do we represent the 'knowledge'?
How do we integrate it in the target-task model?

Word Embeddings for Transfer Learning

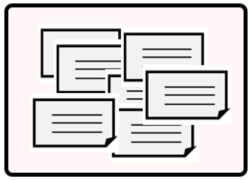
Neural Text Classification



Input word embeddings:

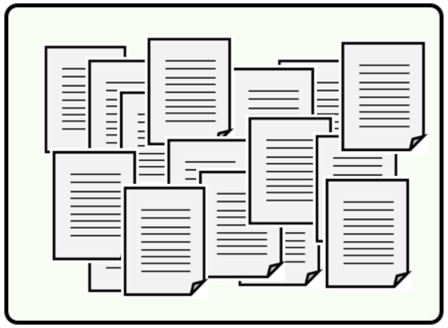
- Train from scratch
- Take pretrained (Word2Vec, GloVe)
- Initialize with pretrained, then fine-tune

Training from scratch vs using pretrained



Training data for text classification (labeled)

- Not huge, or not diverse, or both
- Domain: task-specific



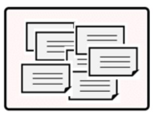
Training data for word embeddings (unlabeled)

- Huge diverse corpus (e.g., Wikipedia)
- Domain: general

Pretrain and fine-tune

- Train from scratch

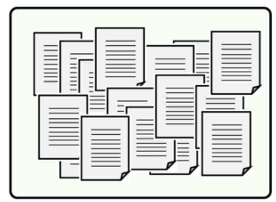
What they will know:



May be not enough to learn relationships between words

- Take pretrained (Word2Vec, GloVe)

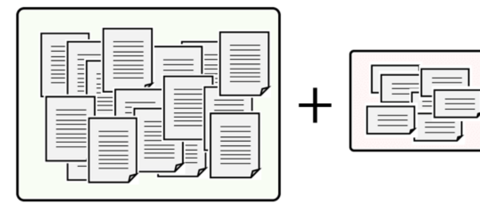
What they will know:



Know relationships between words, but are **not** specific to the task

- **Initialize with pretrained, then fine-tune**

What they will know:

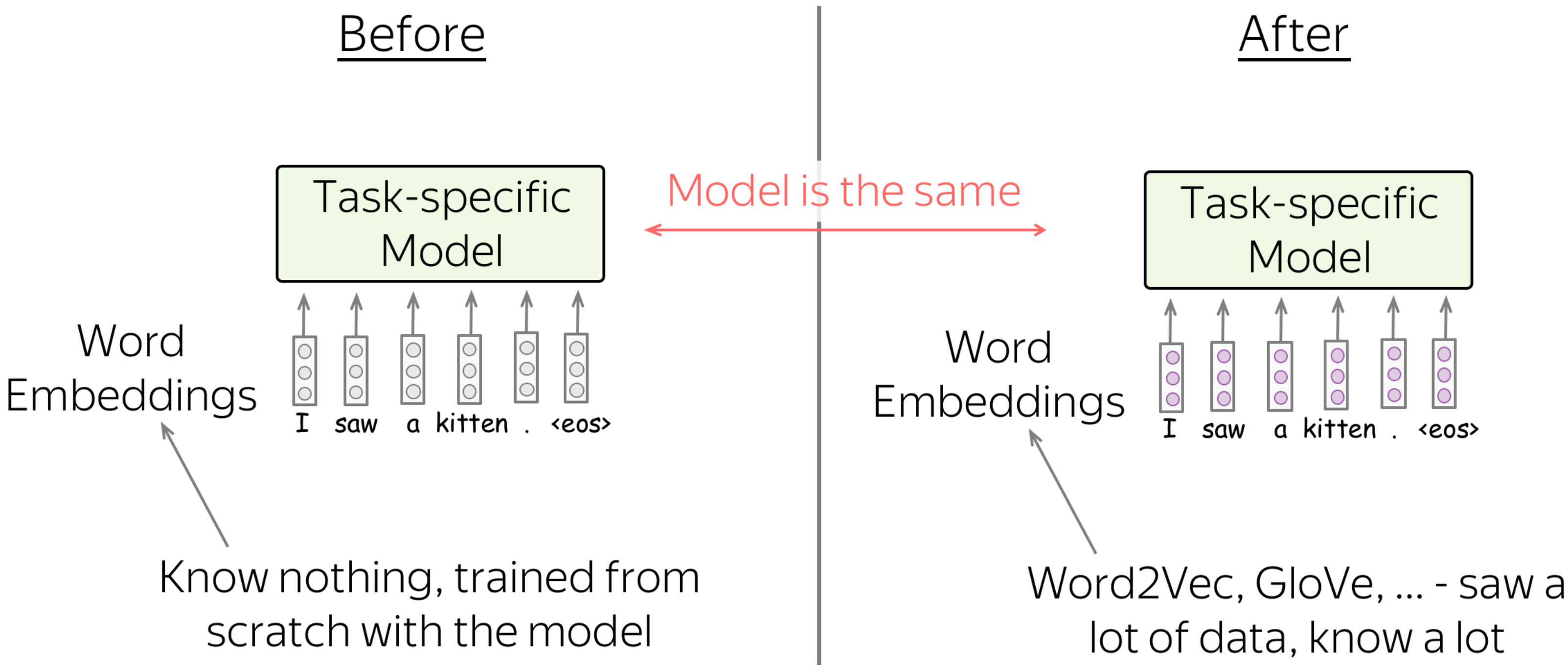


Know relationships between words and adapted for the task

"**Transfer**" knowledge from a huge unlabeled corpus to your task-specific model

Starting with pretrained embeddings and then fine-tuning (= training) them on the specific task enables the transfer of knowledge from a vast dataset, while specializing the embedding to the target task and domain

Transfer through word embeddings



Limitations of transfer with word embeddings

- Word embeddings encode word meanings without considering context, merging all senses of a word

- They also do not represent larger linguistic units (phrases, sentences, paragraphs).

The target model, thus, has to learn disambiguation and composition from limited task-specific data.*

Even more limiting for text generation tasks – the model needs to learn, e.g., how to achieve fluency on the target-task data alone

*however, for some tasks like topic classification, these processes are less critical, so transfer with embeddings can work well

Using a model to acquire the knowledge and transferring the model to the target task

Key ideas

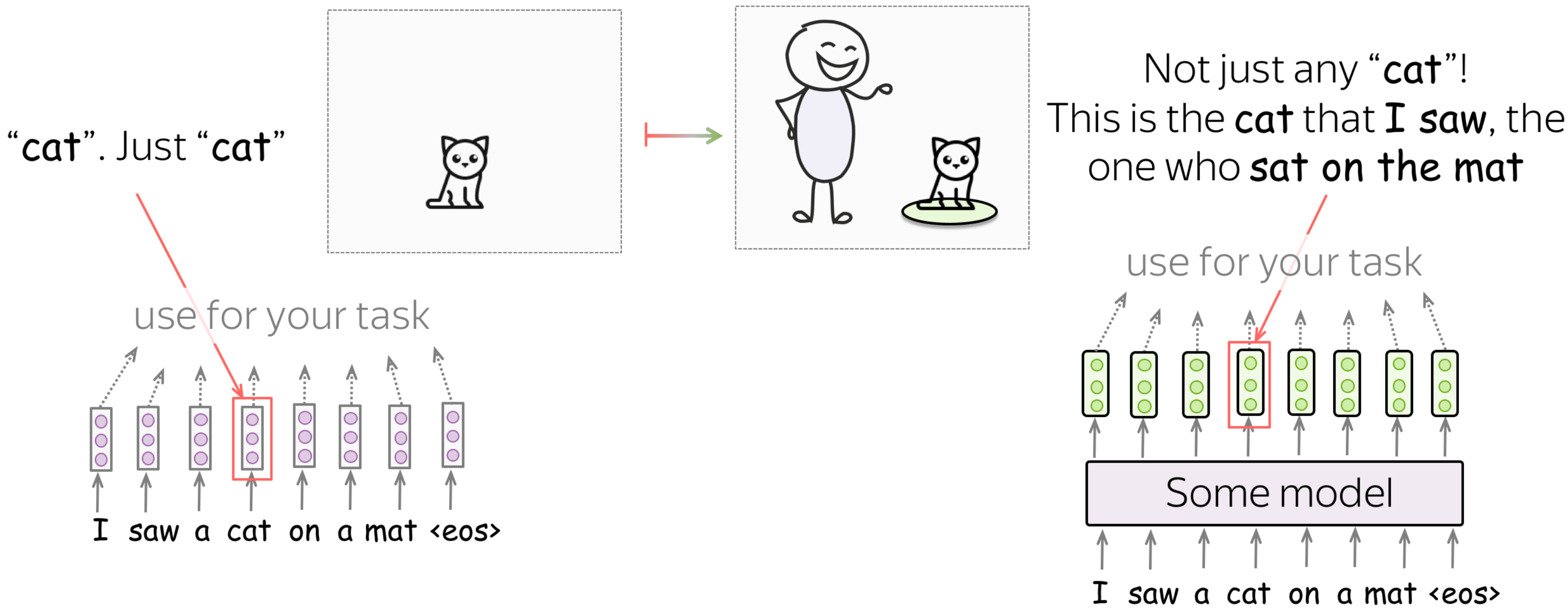
	GloVe, Word2Vec →	CoVe, ELMo →	GPT, BERT
	Great idea 1		
What is encoded	words →	words in context	words in context
How it is used for downstream tasks	Input for task-specific models	Input for task-specific models →	Instead of task-specific models
		Great idea 2	

The two great ideas:

- <u>what is encoded</u>: from words to words-in-context (the transition from Word2Vec to ELMo)

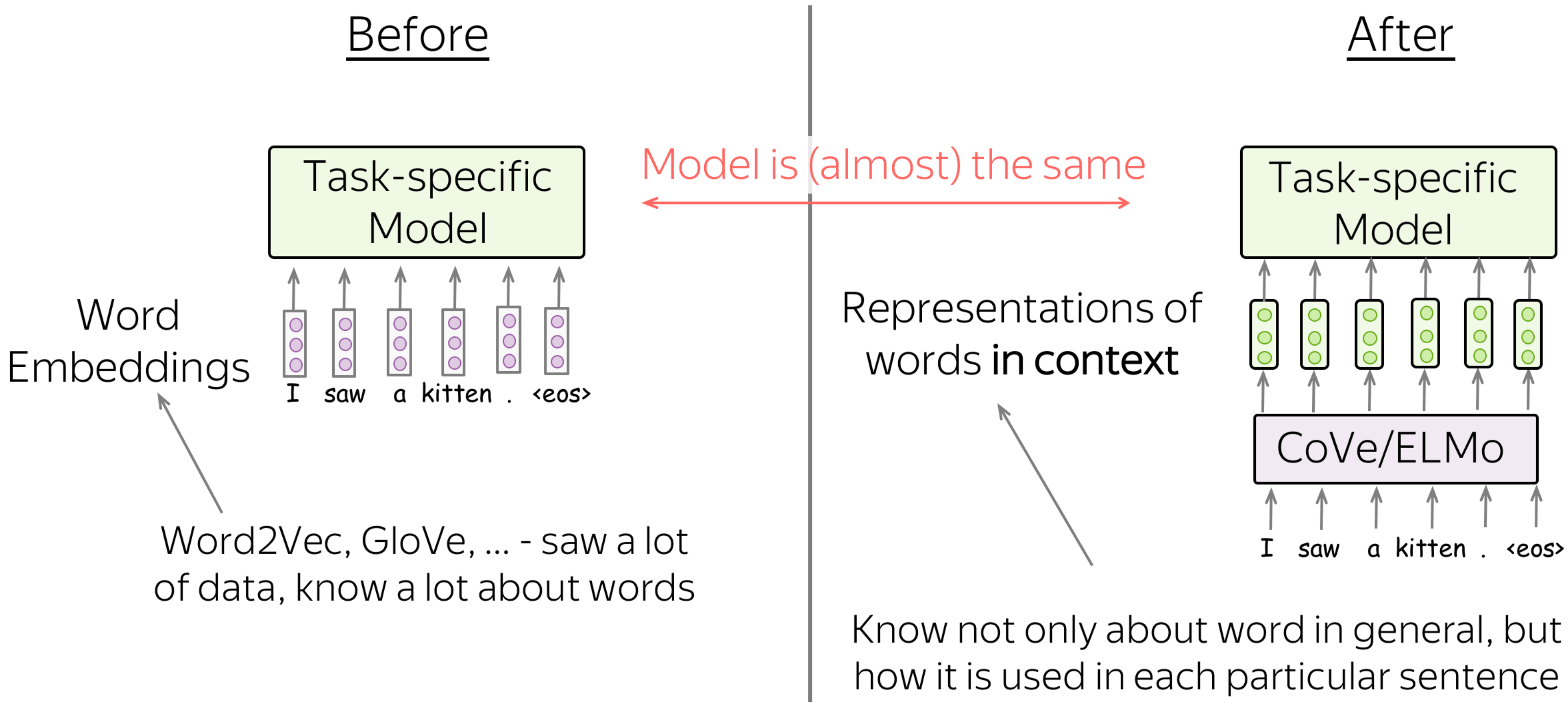
- <u>usage for downstream tasks</u>: from replacing only word embeddings in task-specific models to replacing entire task-specific models (the transition from ELMo to GPT/BERT)

Transferring words-in-context



If successful, the representation of the word will both disambiguate the term and also specialize it for the given context

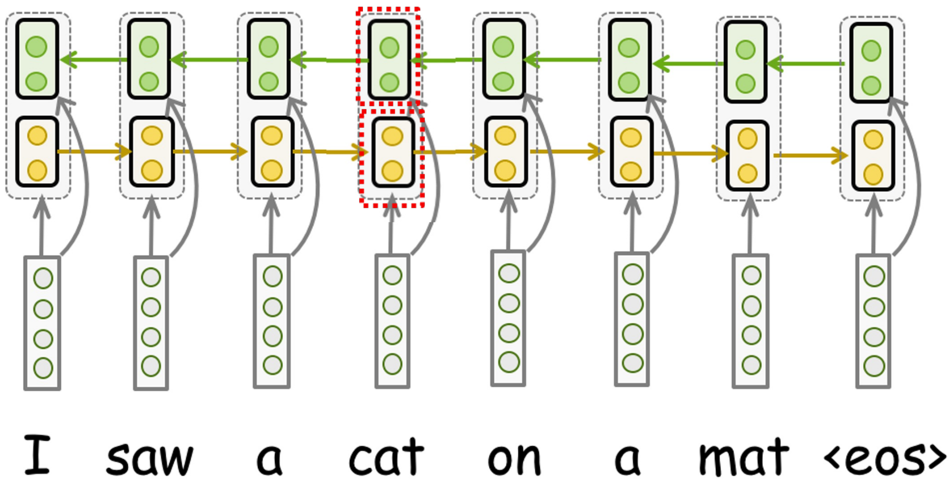
Transferring words-in-context



What is ELMo (or CoVe)?

Word representation with a bidirectional RNN

This is the cat on the mat
and that is the cat that I saw



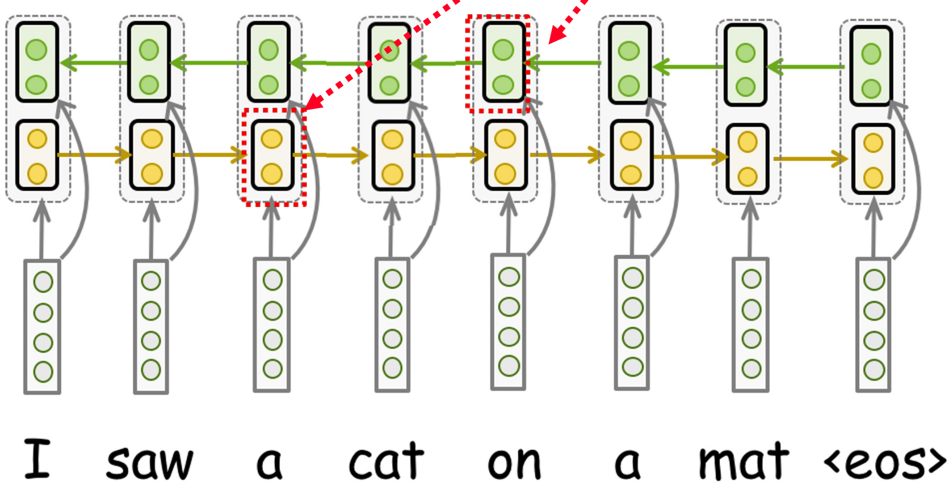
If the RNNs are successfully trained (how?), the marked two states together will represent this information

The pair of states can be used as word-in-context representation

Training RNNs to predict a 'masked' word

"Embeddings from Language Models" (ELMo;
NAACL 2018 best paper award)

This X was on a mat
and this is X that I saw



The two states do not carry information about the token 'cat'

We can train the RNNs to predict the left-out word relying these two states! (masked language modeling)

Why masking words is a good task?

Why training a model to predict a missing word a good pretraining task?

- Because it makes the model acquire a range of capabilities, which can be useful for a range of different target tasks

The University of Edinburgh is located in __________, UK. [**trivia**]

I put __________ fork down on the table. [**syntax**]

The woman walked across the street, checking for traffic over __________ shoulder. [**coreference**]

I went to the ocean to see the fish, turtles, seals, and __________. [**lexical semantics/topic**]

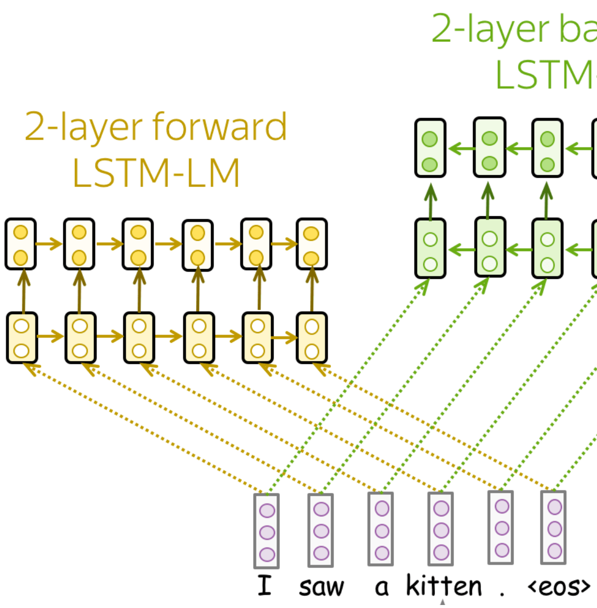
Overall, the value I got from the two hours watching it was the sum total of the popcorn and the drink. The movie was __________. [**sentiment**]

John went into the kitchen to make some tea. Standing next to John, Jake pondered his destiny. Jake left the __________. [**some degree of reasoning**]

I was thinking about the sequence that goes 1, 1, 2, 3, 5, 8, 13, 21, __________ [**some arithmetic reasoning**]

Examples by Pasquale Minervini

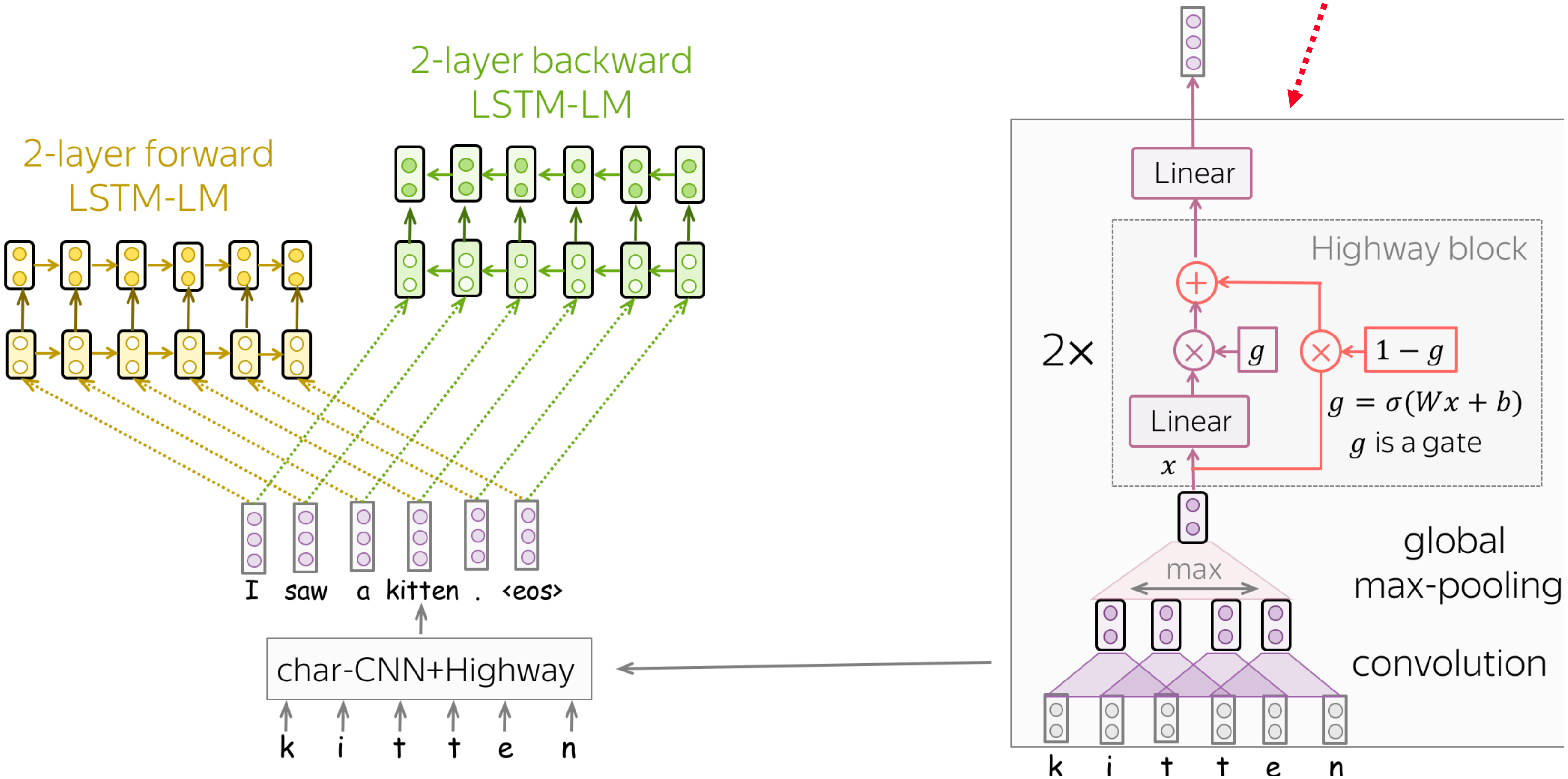
ELMo is a bit more complicated



Having more than one layer is crucial, it is not only about the expressivity!
(ask me why but don’t expect a short answer)

ELMo is a bit more complicated

Let's not worry about these details too much

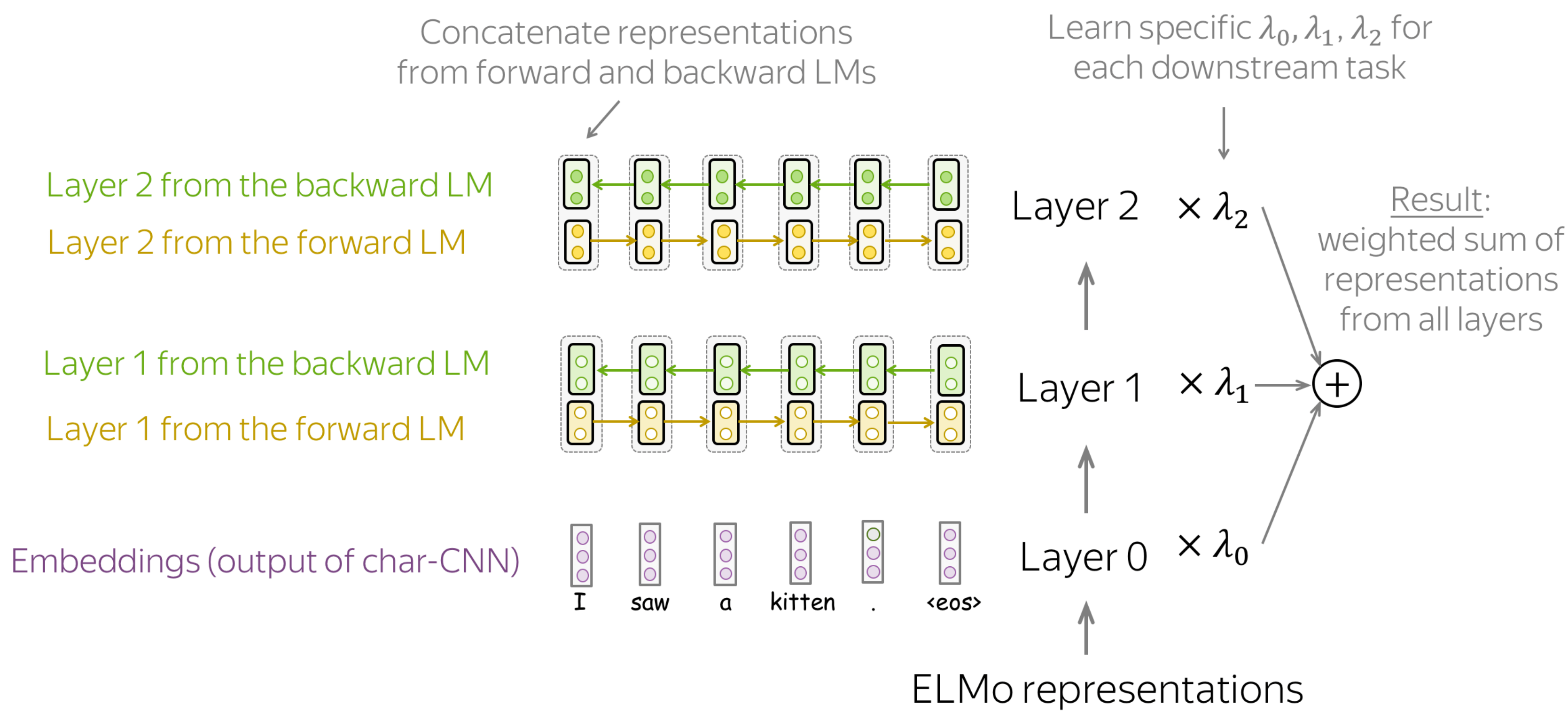


The word embeddings are computed from character embeddings

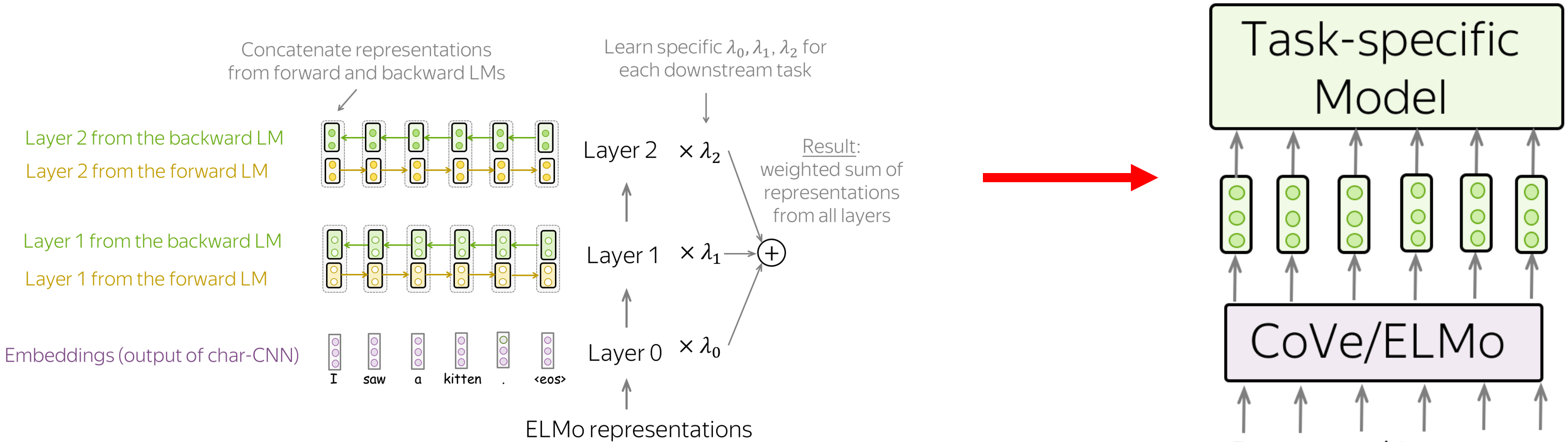
– makes it possible to handle 'unseen' words [an alternative to more common subword tokenization, from lecture 26]

This provides an inductive bias that words with similar spellings often have similar meaning (e.g., running, run, runner, runs)

How do we actually use ELMo embeddings



Word representation with a bidirectional RNN



On top of weighted sum of ELMo layers, you then train a task-specific model

ELMo was a very important model, resulting in big progress over 'static' word embeddings across a range of tasks

Where are we in the lecture?

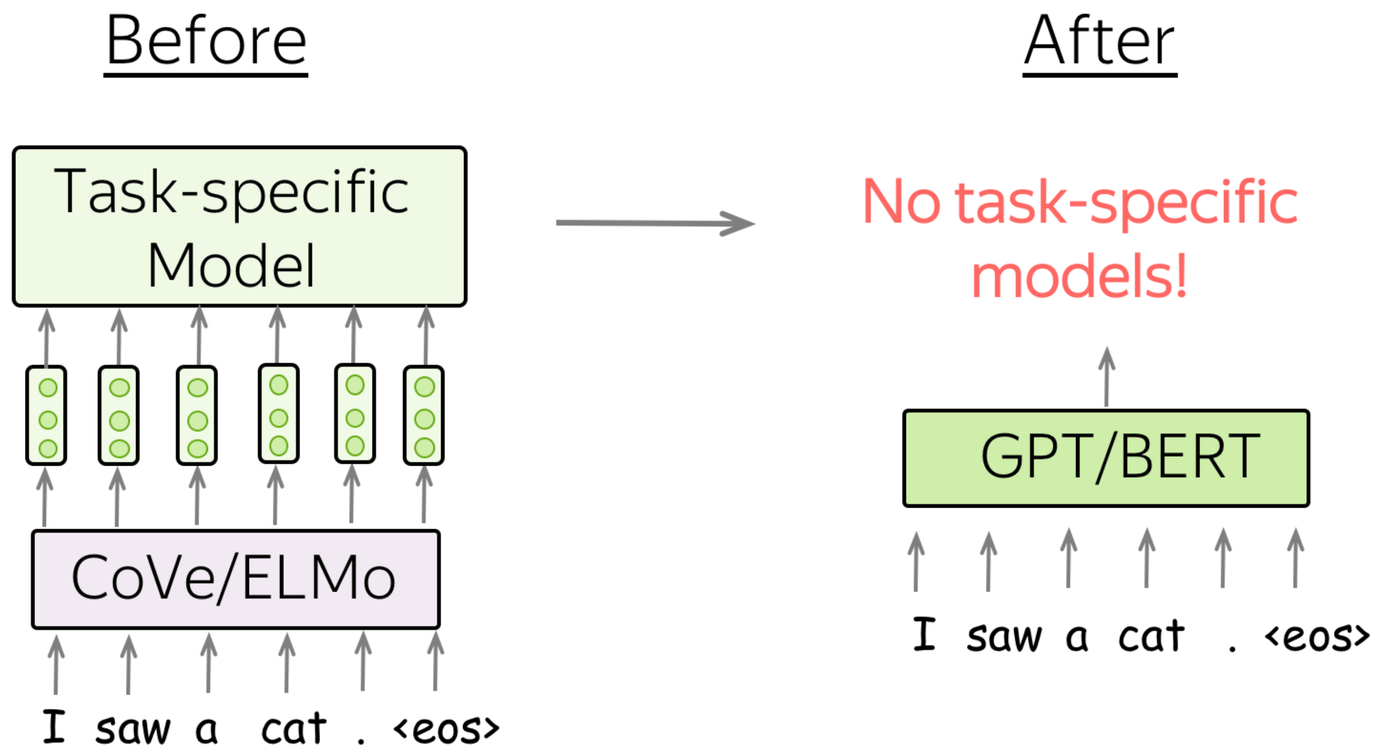
	GloVe, Word2Vec →	CoVe, ELMo →	GPT, BERT
		Great idea 1	
What is encoded	words →	words in context	words in context
How it is used for downstream tasks	Input for task-specific models	Input for task-specific models →	Instead of task-specific models
			Great idea 2

The two great ideas:

- <u>what is encoded</u>: from words to words-in-context (the transition from Word2Vec to ELMo)

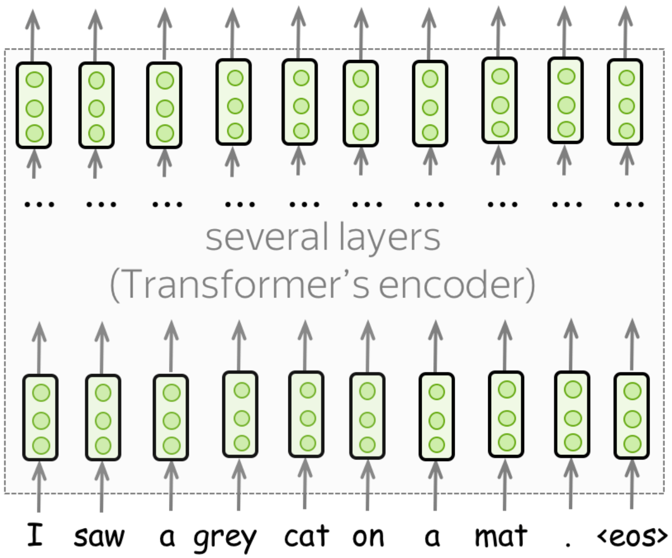
- <u>usage for downstream tasks</u>: from replacing only word embeddings in task-specific models to replacing entire task-specific models (the transition from ELMo to GPT/BERT)

Pre-training a model and fine-tuning this model on the target task



BERT: Bidirectional Encoder Representations from Transformers

NAACL 2019 best paper award



Model architecture:

- Transformer's encoder

What is special about it:

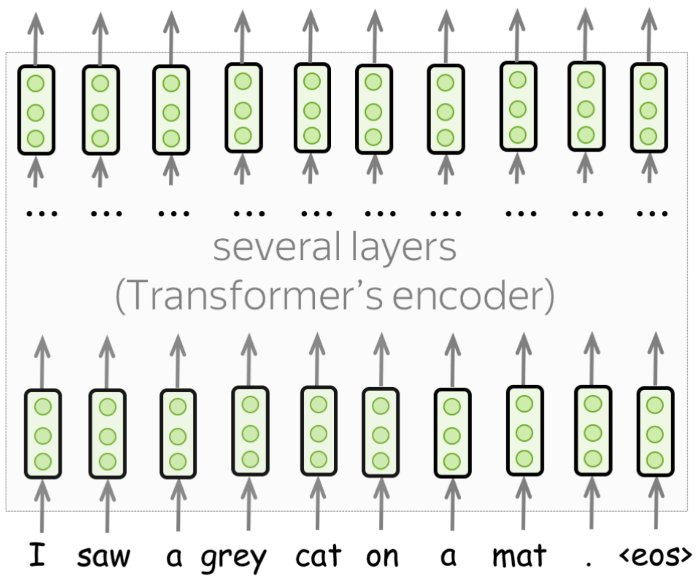
- Training objectives
  - MLM: Masked language modeling
  - NSP: Next sentence prediction
- The way it is used
  - No task-specific models

We will ignore NSP as later studies have demonstrated that it was not particulary useful (the subsequent RoBERTa model ditched NSP)

Once trained, we can use BERT as a usual Transformer encoder, but let's understand how to train it first

Masked language modeling objective

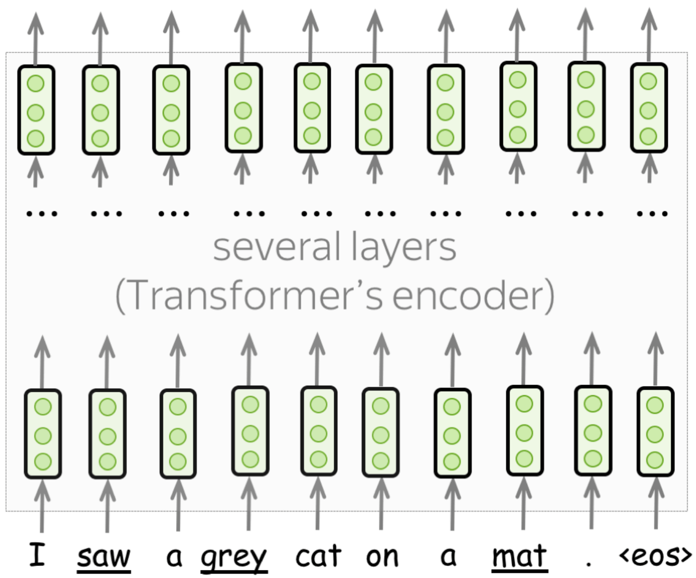
Similar to ELMo's objective, but the RNN state-selection trick cannot be used with Transformers



At each training step:

Masked language modeling objective

Similar to ELMo’s objective, but the RNN state-selection trick cannot be used with Transformers

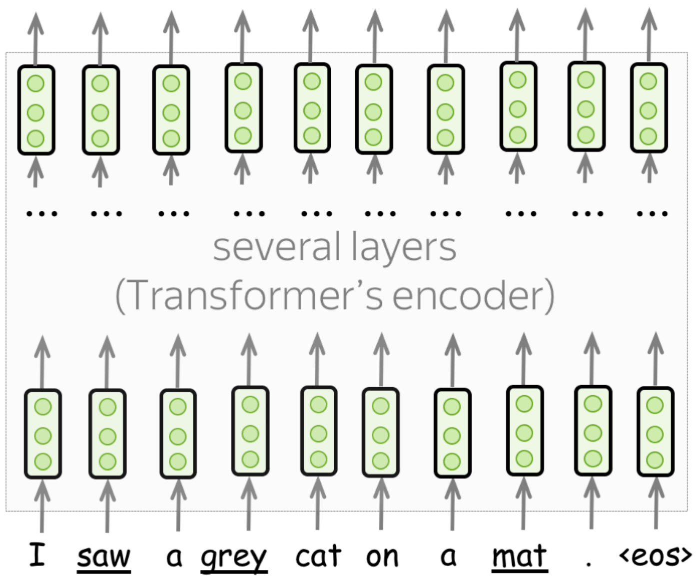


At each training step:

- pick randomly 15% of tokens

Masked language modeling objective

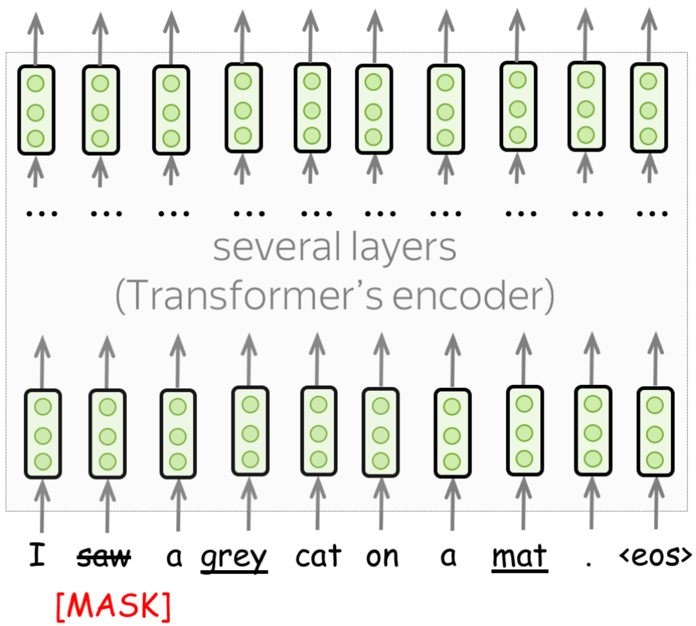
Similar to ELMo's objective, but the RNN state-selection trick cannot be used with Transformers



At each training step:

- pick randomly 15% of tokens
- replace each of the chosen tokens with something

Masked language modeling objective

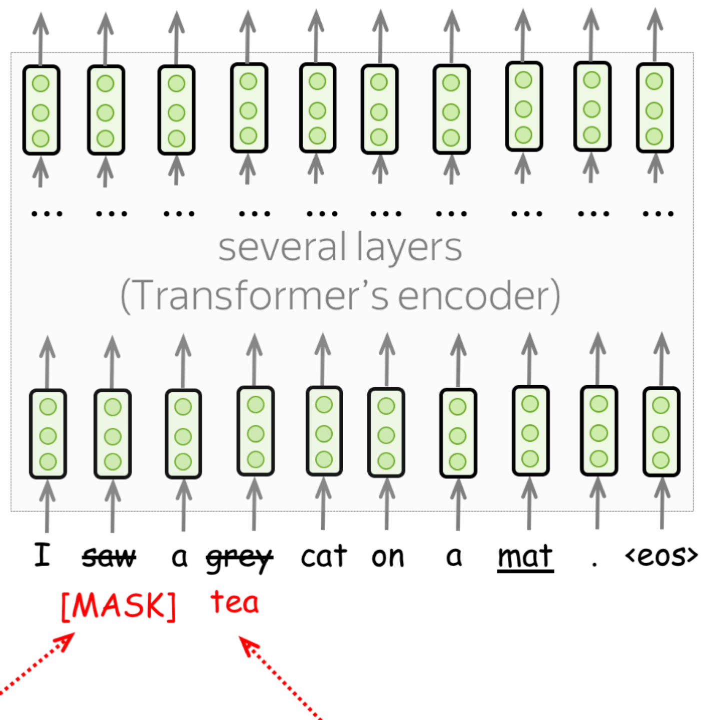


At each training step:

- pick randomly 15% of tokens
- replace each of the chosen tokens with something

  - [MASK], with p = 80%

Masked language modeling objective



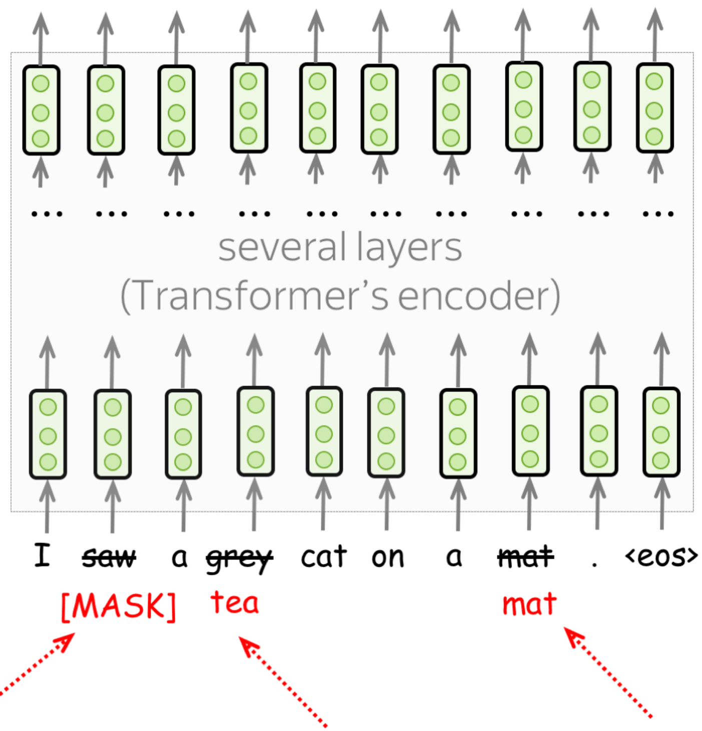
At each training step:

- pick randomly 15% of tokens
- replace each of the chosen tokens with something

- [MASK], with p = 80%
- Random token, with p = 10%

Masked language modeling objective

Similar to ELMo's objective, but the state-selection trick cannot be used with Transformers

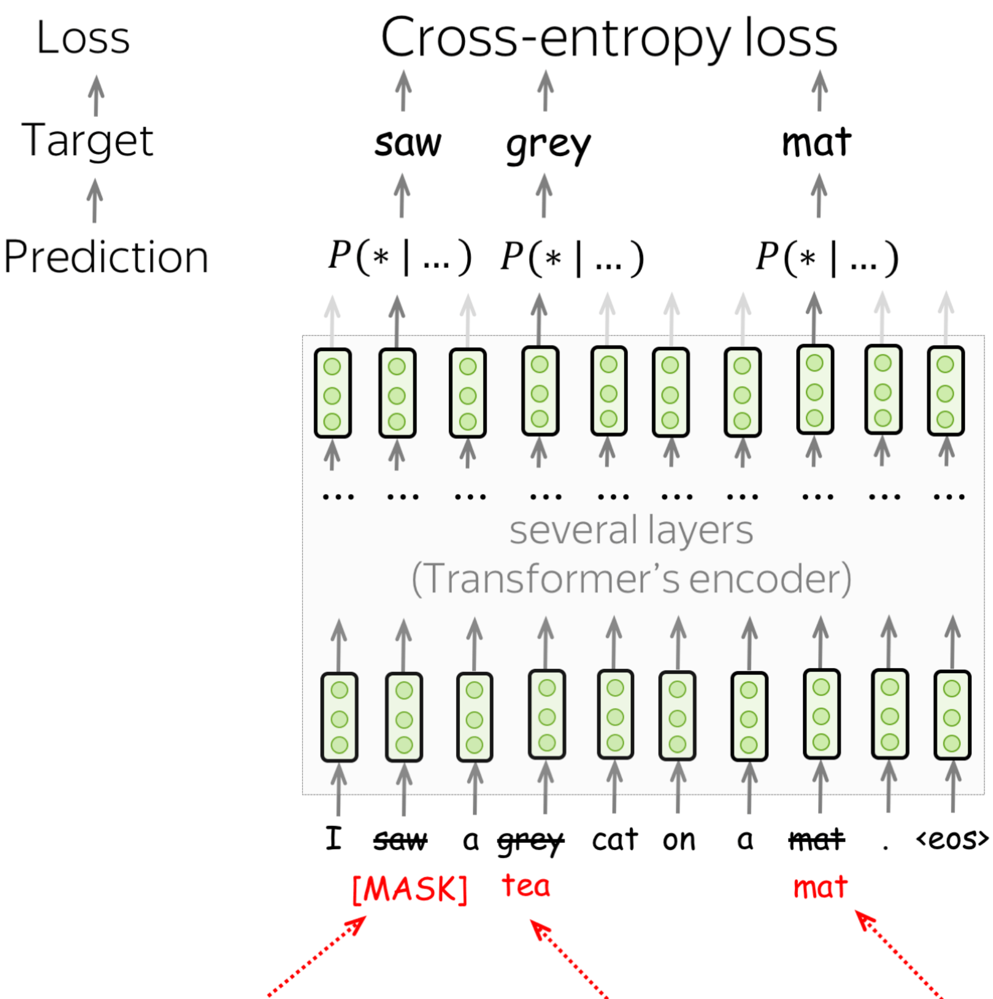


At each training step:

- pick randomly 15% of tokens
- replace each of the chosen tokens with something

  - **[MASK]**, with p = 80%
  - Random token, with p = 10%
  - Original token, with p = 10%

Masked language modeling objective



At each training step:

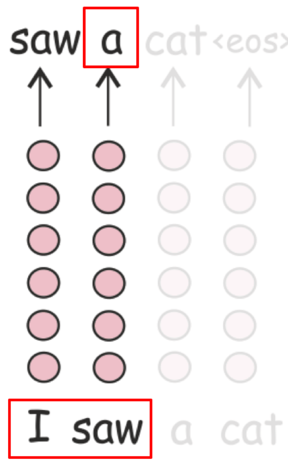
- pick randomly 15% of tokens
- replace each of the chosen tokens with something
- predict original chosen tokens

  - **[MASK]**, with p = 80%
  - Random token, with p = 10%
  - Original token, with p = 10%

Masked language modeling vs language modeling

Language Modeling

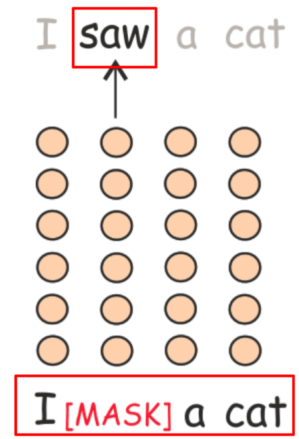
- Target: next token
- Prediction: $P(* \mid \text{I saw})$



left-to-right, does not see future

Masked Language Modeling

- Target: current token (the true one)
- Prediction: $P(* \mid \text{I [MASK] a cat})$



sees the whole text, but something is corrupted

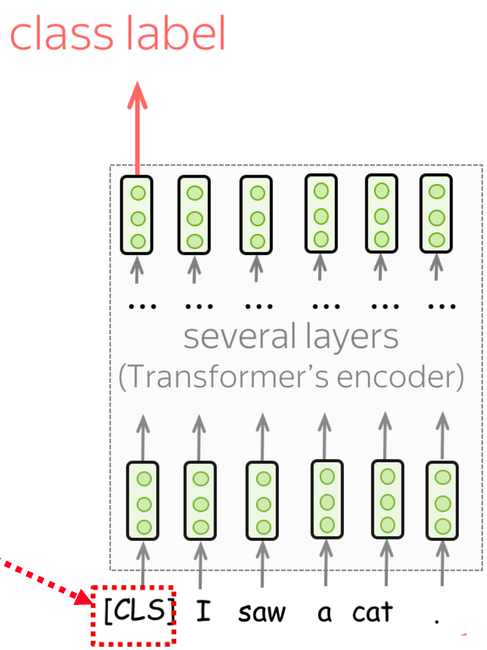
MLM is still language modeling: the goal is to predict some tokens in a sentence/text based on some part of this text (strictly speaking it is called pseudo-likelihood)

The advantage of LM is that we do not need annotated data, this is a task which both can instantiated on unlabeled data and requires (at least for a subset of predictions) acquiring complex and diverse text 'understanding'

Using BERT for downstream tasks

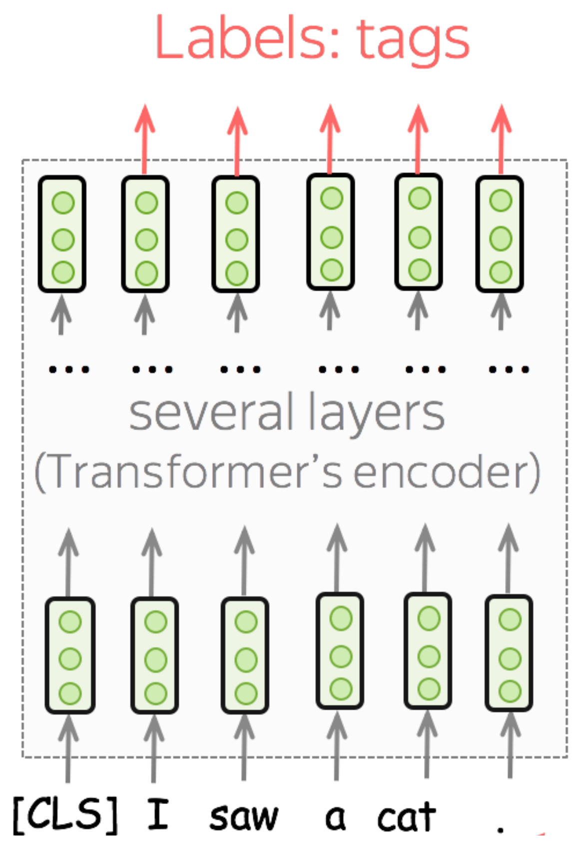
<u>Classification:</u> take pretained BERT, add a classifier on top and fine-tune (i.e. optimize the parameters) for the target task

This is just a special token. With BERT, it was used within NSP objective (see earlier). In practice, it can be just any extra token



Using BERT for downstream tasks

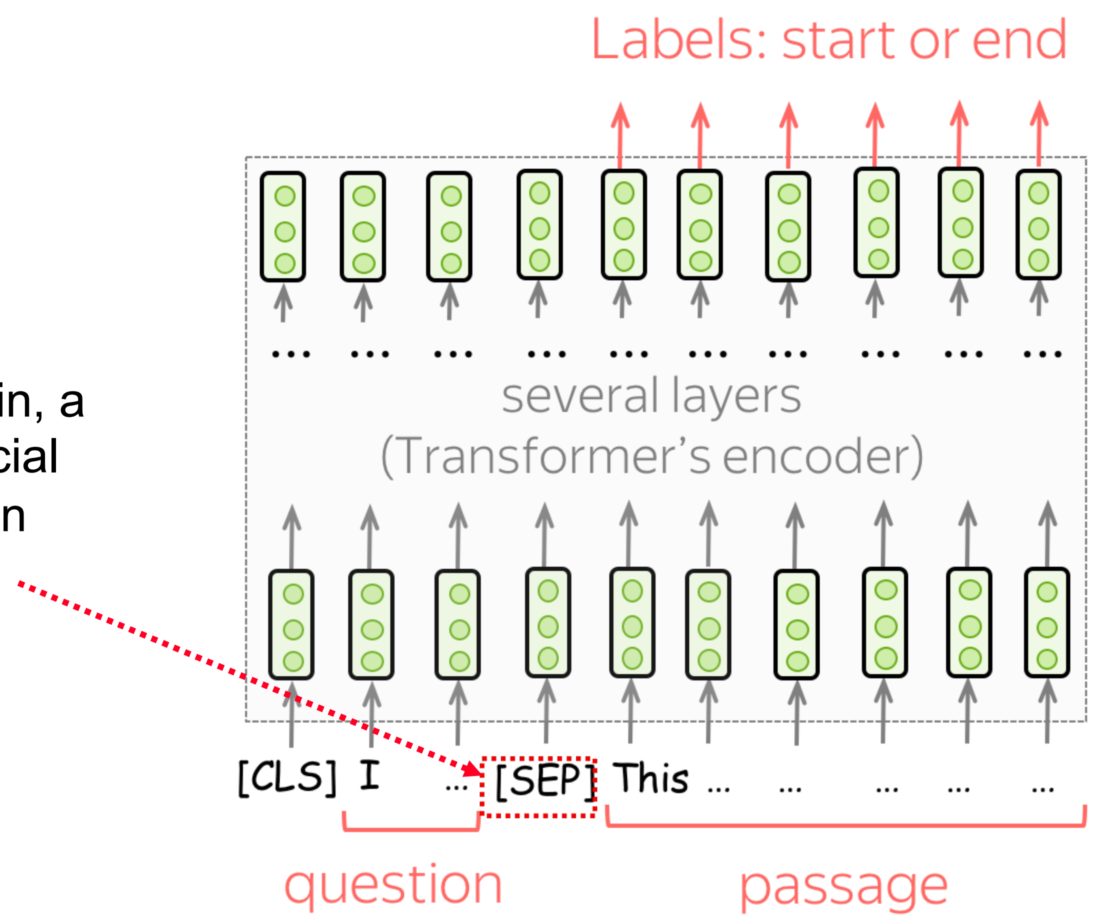
<u>Sequence tagging:</u> PoS tagging or named entity recognition



Using BERT for downstream tasks

<u>Question answering</u> on a basis of text passage: mark the position where the answer starts and ends

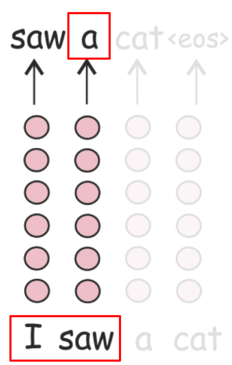
Again, a special token



Using BERT for downstream tasks

The key limitation of BERT – it is (almost) impossible to use BERT to generate text, so what do you do if you downstream task is a generation task (e.g., summarization, report generation, chat or translation)?

<u>The solution:</u> pretrain on the standard language modelling objective rather than masked language modelling (GPT, GPT-2, ...)



left-to-right, does not see future

(or pretrain encoder-decoder models, such as Google's T5)

Current trends and hot research topics

- Very large models, trained on lots of data (running out of text created by humanity...)

- Emerging abilities:
  - e.g., in context learning – a training set is provided as input to the model in the prompt

- Making models follow instructions / complex prompts
  - through fine-tuning on data for various instruction-following tasks and using human feedback

- Effective ways of fine-tuning these huge models

- Multimodal models (images, video, speech, …)

- Interpretability

- Using language models in other domains (e.g., for planning in robotics)

- Risks and biases
  - attempts to make large language models safe(r)

Take aways

- Transfer learning is behind current successes in NLP
- Word embeddings provide a simple but somewhat limited way of achieving the transfer
- Word-in-context embeddings
- Pre-training with (masked) language modeling
- Pre-train and fine-tune methodology