NLP: Pretraining and Applications

CS60010: Deep Learning

Abir Das

IIT Kharagpur

Mar 30 and 31, 2022

Agenda

- Discussion on unsupervised pretraining towards word embedding
- Discussion on Word2Vec, ELMO, BERT

Resources

§ CS W182 course by Sergey Levine at UC Berkeley. [Link] [Lecture 13]

The Big Idea: Unsupervised Pretraining

- Deep learning works best when we have a lot of data
- **Good news**: there is plenty of text data out there!
- **Bad news**: most of it is unlabeled
- 1,000s of times more data without labels (*i.e.*, valid English text in books, news, web) vs. labeled/paired data (*e.g.*, English/French translations)
- **The big challenge**: how can we use **freely available** and **unlabeled** text data to help us apply deep learning methods to NLP?

Source: CS W182 course, Sergey Levine, UC Berkeley

Start Simple: How do we Represent Words

$$x = \begin{bmatrix} 0 \\ 0 \\ . \\ 0 \\ 1 \\ 0 \\ . \\ 0 \end{bmatrix}$$

- Dimensionality = Number of words in vocabulary
- Not great, not terrible
- Semantic relationship is not preserved

Source: CS W182 course, Sergey Levine, UC Berkeley

Start Simple: How do we Represent Words

$$x = \begin{bmatrix} 0 \\ 0 \\ . \\ 0 \\ 1 \\ 0 \\ . \\ 0 \end{bmatrix}$$

- Dimensionality = Number of words in vocabulary
- Not great, not terrible
- Semantic relationship is not preserved
- The pixels mean something! Not a great metric space, but, still, they mean something

Source: CS W182 course, Sergey Levine, UC Berkeley

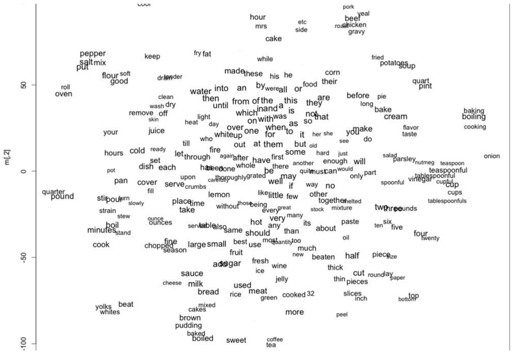
Start Simple: How do we Represent Words

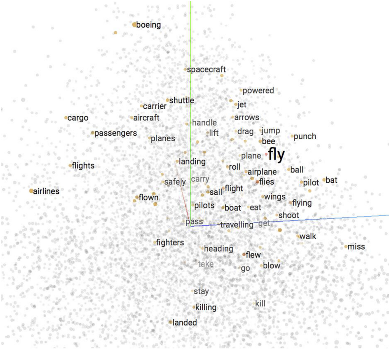
$$x = \begin{bmatrix} 0 \\ 0 \\ . \\ 0 \\ 1 \\ 0 \\ . \\ 0 \end{bmatrix}$$

- Dimensionality = Number of words in vocabulary
- Not great, not terrible
- Semantic relationship is not preserved
- The pixels mean something! Not a great metric space, but, still, they mean something
- Maybe if we had a more meaningful representation of words, then learning downstream tasks would be much easier!
- Meaningful = vectors corresponding to similar words should be close

Source: CS W182 course, Sergey Levine, UC Berkeley

Some Examples of Good Word Embedding

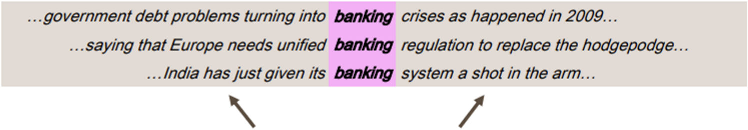




Source: CS W182 course, Sergey Levine, UC Berkeley

How do we learn embeddings?

- **Basic idea**: the meaning of a word is determined by what other words occur in close proximity to it in sentences
- Learn a representation for each word such that its neighbors are "close" under this representation

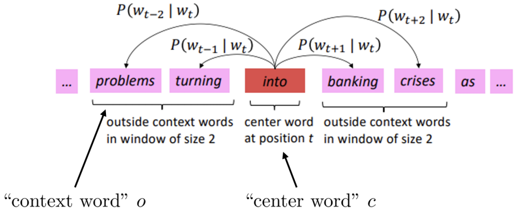


These context words will represent ***banking***

- **Terminology**: The other words which are close to the word in question are known as *context* words. Specifically, context words are words that occur within some distance of the word in question

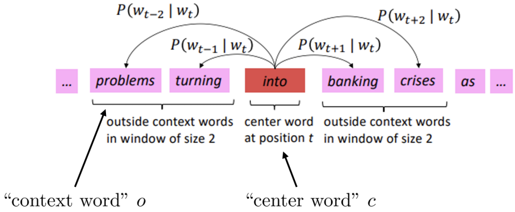
Source: CS W182 course, Sergey Levine, UC Berkeley

More Formally



Source: CS W182 course, Sergey Levine, UC Berkeley

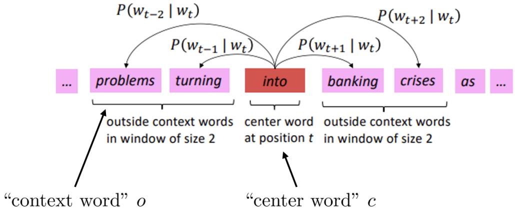
More Formally



- Cast it as a binary classification problem - is this the right context word or not?

Source: CS W182 course, Sergey Levine, UC Berkeley

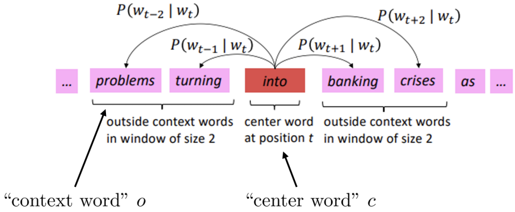
More Formally



- Cast it as a binary classification problem - is this the right context word or not?
- u_o and v_c denote the vector representations of the context word o and the center word c respectively
- For every word in the vocabulary these two vectors are maintained.
- Our goal is to learn them. Once learned, we generally get a single representation of the words by averaging these two

Source: CS W182 course, Sergey Levine, UC Berkeley

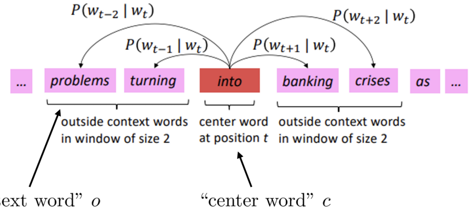
More Formally



- Cast it as a binary classification problem - is this the right context word or not?
- u_o and v_c denote the vector representations of the context word o and the center word c respectively
- For every word in the vocabulary these two vectors are maintained.
- Our goal is to learn them. Once learned, we generally get a single representation of the words by averaging these two
- The idea gave rise to word2vec model by Tomas Mikolov *et al.*

Source: CS W182 course, Sergey Levine, UC Berkeley

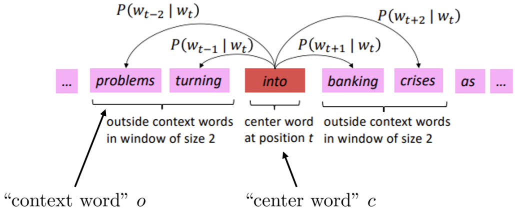
More Formally



- $p(o \text{ is the right word}|c) = \sigma(u_o^T v_c) = \frac{1}{1+\exp -u_o^T v_c}$
- This brings the right context word u_o and right center word v_c close together

Source: CS W182 course, Sergey Levine, UC Berkeley

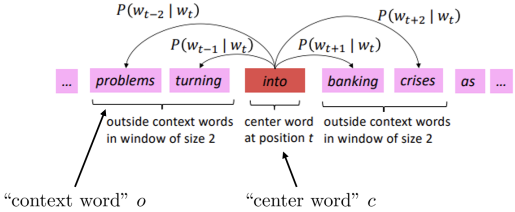
More Formally



- $p(o \text{ is the right word}|c) = \sigma(u_o^T v_c) = \frac{1}{1+\exp -u_o^T v_c}$
- This brings the right context word u_o and right center word v_c close together
- But we need to also provide some negative examples to learn
- $p(o \text{ is the wrong word}|c) = \sigma(-u_o^T v_c) = \frac{1}{1+\exp u_o^T v_c}$
- This will push the wrong pairs of words further apart

Source: CS W182 course, Sergey Levine, UC Berkeley

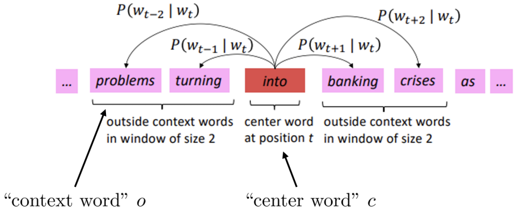
More Formally



- $p(o \text{ is the right word}|c) = \sigma(u_o^T v_c) = \frac{1}{1+\exp -u_o^T v_c}$
- This brings the right context word u_o and right center word v_c close together
- But we need to also provide some negative examples to learn
- $p(o \text{ is the wrong word}|c) = \sigma(-u_o^T v_c) = \frac{1}{1+\exp u_o^T v_c}$
- This will push the wrong pairs of words further apart
- For every center word c and every context word o we will add the "right" log probabilities. Then for some randomly sampled words for the same center word we will add the "wrong" log probabilites

Source: CS W182 course, Sergey Levine, UC Berkeley

More Formally



- This sum is then minimized over the word representations

$$\arg\max_{u_1,\cdots,u_n,v_1,\cdots,v_n} \sum_{c,o} \log p(o \text{ is the right}|c) + \sum_{c,w} \log p(w \text{ is the wrong}|c)$$

Source: CS W182 course, Sergey Levine, UC Berkeley

Word2Vec Summary

- $p(o \text{ is the right word}|c) = \sigma(u_o^T v_c) = \frac{1}{1+\exp{-u_o^T v_c}}$
- $p(w \text{ is the wrong word}|c) = \sigma(-u_w^T v_c) = \frac{1}{1+\exp{u_w^T v_c}}$
- $\underset{u_1,\cdots,u_n,v_1,\cdots,v_n}{\arg\max} \sum_{c,o} \log p(o \text{ is the right}|c) + \sum_{c,w} \log p(w \text{ is the wrong}|c)$
- $\underset{u_1,\cdots,u_n,v_1,\cdots,v_n}{\arg\max} \sum_{c,o} \log \sigma(u_o^T v_c) + \sum_{c,w} \log \sigma(-u_w^T v_c)$

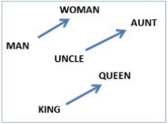
Source: CS W182 course, Sergey Levine, UC Berkeley

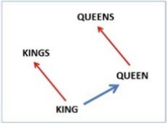
Word2Vec Examples

Algebraic relations:

vec(“woman")−vec(“man") ≃ vec(“aunt")−vec(“uncle")

vec(“woman")−vec(“man") ≃ vec(“queen")−vec(“king")





Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

Source: CS W182 course, Sergey Levine, UC Berkeley

Contextual Representations

- Word embeddings associate a vector with each word. This can make for a much better representation than just a one-hot vector.

Source: CS W182 course, Sergey Levine, UC Berkeley

Contextual Representations

- Word embeddings associate a vector with each word. This can make for a much better representation than just a one-hot vector.
- The vector does not change if the word is used in different ways.
  - Let's play baseball
  - I saw a play yesterday
- Same word2vec representation, even though they mean different.

Source: CS W182 course, Sergey Levine, UC Berkeley

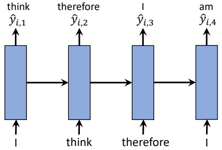
Contextual Representations

- Word embeddings associate a vector with each word. This can make for a much better representation than just a one-hot vector.
- The vector does not change if the word is used in different ways.
  - Let's play baseball
  - I saw a play yesterday
- Same word2vec representation, even though they mean different.
- Can we learn word representations that **depend on context**?

Source: CS W182 course, Sergey Levine, UC Berkeley

Contextual Representations

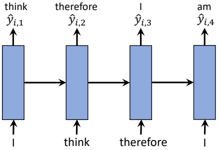
- Word embeddings associate a vector with each word. This can make for a much better representation than just a one-hot vector.
- The vector does not change if the word is used in different ways.
  - Let's play baseball
  - I saw a play yesterday
- Same word2vec representation, even though they mean different.
- Can we learn word representations that **depend on context**?
- High level idea:
  - Train a language model
  - Run it on a sentence
  - Use its hidden state



Source: CS W182 course, Sergey Levine, UC Berkeley

Contextual Representations

- Word embeddings associate a vector with each word. This can make for a much better representation than just a one-hot vector.
- The vector does not change if the word is used in different ways.
  - Let's play baseball
  - I saw a play yesterday
- Same word2vec representation, even though they mean different.
- Can we learn word representations that **depend on context**?
- High level idea:
  - Train a language model
  - Run it on a sentence
  - Use its hidden state



- Question 1: How to train the best language model for this?
- Question 2: How to use this language model for downstream tasks?

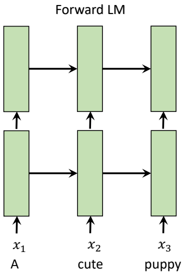
Source: CS W182 course, Sergey Levine, UC Berkeley

Contextual Representations

- **ELMO**: **E**mbedding from **L**anguage **Mo**dels
  Peters *et al.* "Deep Contextualized Word Representations", NAACL 2018.
  Bidirectional LSTM model used for context-dependent embeddings

- **BERT**: **B**idirectional **E**ncoder **R**epresentations from **T**ransformers
  Devlin *et al.* "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", NAACL 2019.
  Transformer language model used for context-dependent embeddings

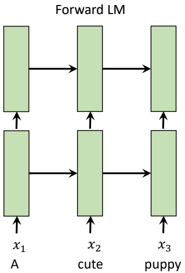
ELMO



§ ELMO, in essence, is a language model (recurrent) producing the next word given the words so far in the sentence

Source: CS W182 course, Sergey Levine, UC Berkeley

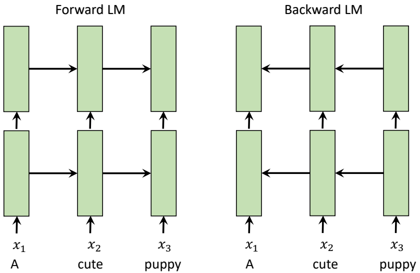
ELMO



- ELMO, in essence, is a language model (recurrent) producing the next word given the words so far in the sentence
- Problem with this basic approach is that the representation of a word in a sentence will depend only on the previous words - not on the entire sentence
- Compare this with word2vec. It used context words both before and after the center word

Source: CS W182 course, Sergey Levine, UC Berkeley

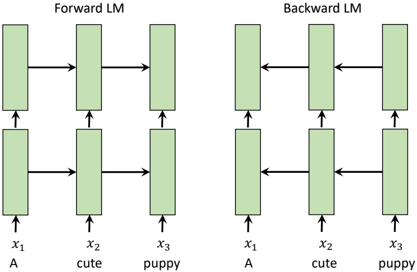
ELMO



- ELMO, in essence, is a language model (recurrent) producing the next word given the words so far in the sentence
- Problem with this basic approach is that the representation of a word in a sentence will depend only on the previous words - not on the entire sentence
- Compare this with word2vec. It used context words both before and after the center word
- There can be many ways to resolve this. ELMO uses two separate language models

Source: CS W182 course, Sergey Levine, UC Berkeley

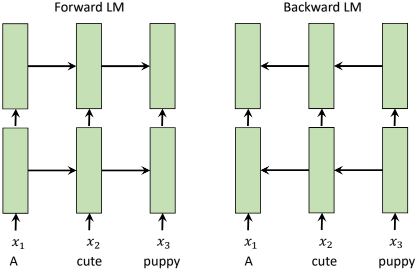
ELMO



- The backward LM runs over the sequence in reverse, predicting the previous word given the future
- In practice, the two models share parameters from the initial embedding layer and last fc layer. The LSTMs of the two models do not share parameters

Source: CS W182 course, Sergey Levine, UC Berkeley

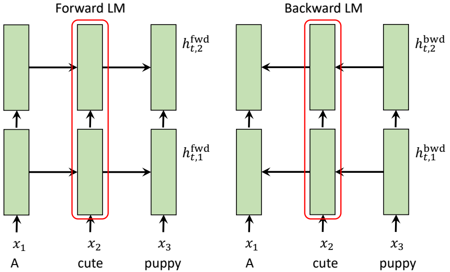
ELMO



- The backward LM runs over the sequence in reverse, predicting the previous word given the future
- In practice, the two models share parameters from the initial embedding layer and last fc layer. The LSTMs of the two models do not share parameters
- Now if you have representations of the same word from both the models, it will contain both forward and backward information

Source: CS W182 course, Sergey Levine, UC Berkeley

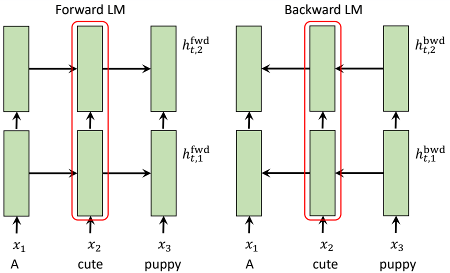
ELMO



- "Together" all these hidden states form a representaiton of the word 'cute'

Source: CS W182 course, Sergey Levine, UC Berkeley

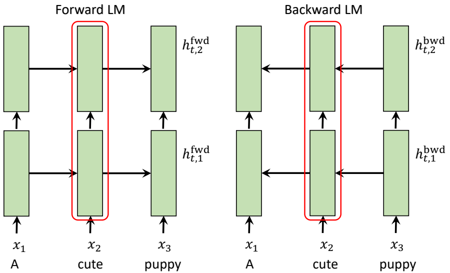
ELMO



- "Together" all these hidden states form a representaiton of the word 'cute'
- Simple option: $\text{ELMO}_t = [h_{t,2}^{\text{fwd}}, h_{t,2}^{\text{bwd}}]$

Source: CS W182 course, Sergey Levine, UC Berkeley

ELMO

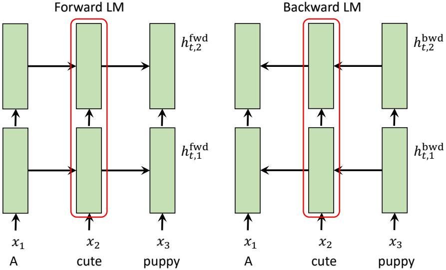


- "Together" all these hidden states form a representaiton of the word 'cute'
- Simple option: $\text{ELMO}_t = [h_{t,2}^{\text{fwd}}, h_{t,2}^{\text{bwd}}]$
- Complex option: $\text{ELMO}_t = \gamma \sum_{i=1}^{L} w_i [h_{t,i}^{\text{fwd}}, h_{t,i}^{\text{bwd}}]$

  w_i are softmax-normalized weights and γ allows the task specific model to scale the entire ELMo vector

Source: CS W182 course, Sergey Levine, UC Berkeley

ELMO



- w_i and γ are learned.
- After taking hidden representations from an ELMO model pretrained on large amount of text data, w_i and γ are learned for the particular downstream task
- ELMO$_t$ is concatenated with other word representations (*e.g.*, word2vec) and passed thorugh the model for the task
- Model parameters along with w_i and γ are also learned

Source: CS W182 course, Sergey Levine, UC Berkeley

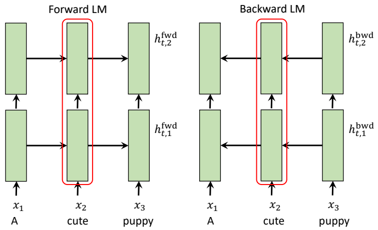
ELMO

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 shows improved performance in six downstream tasks

- Question answering
- Textual entailment
- Semantic role labeling
- Coreference resolution
- Named entity extraction
- Sentiment analysis

ELMO Summary



- Train **forward** and **backward** language models on a large corpus of **unlabeled** text data
- Use the (concatenated) forward and backward LSTM states to represent the word **in context**
- Concatenate the ELMo embedding to the word embedding (or one-hot vector) as an **input** into a downstream task-specific sequence model
- This provides a context specific and semantically meaningful representation of each token

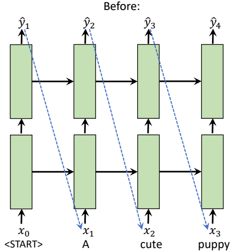
Source: CS W182 course, Sergey Levine, UC Berkeley

BERT

- BERT is transformer based and either the classic BERT or its variants are de facto standard, now-a-days, for word embedding

Source: CS W182 course, Sergey Levine, UC Berkeley

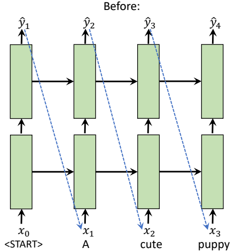
BERT



- BERT is transformer based and either the classic BERT or its variants are de facto standard, now-a-days, for word embedding
- What if we would like to naively replace LSTM with transformer
- ELMO was trained as a language model. So we could try to train transformer as a language model

Source: CS W182 course, Sergey Levine, UC Berkeley

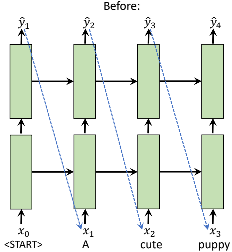
BERT



- BERT is transformer based and either the classic BERT or its variants are de facto standard, now-a-days, for word embedding
- What if we would like to naively replace LSTM with transformer
- ELMO was trained as a language model. So we could try to train transformer as a language model
- Before we used transformer in seq-to-seq model where the language model is the decoder and the encoder provides the ‘condition’
- All we have to do to get an unconditional language model is to use the same decoder but remove the condition

Source: CS W182 course, Sergey Levine, UC Berkeley

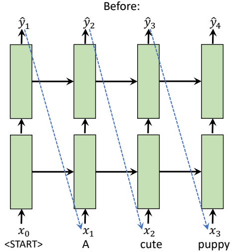
BERT

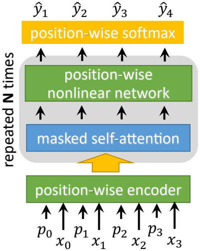


- Cross-attention was responsible for the condition and we remove it from the transformer decoder to get a language model

Source: CS W182 course, Sergey Levine, UC Berkeley

BERT

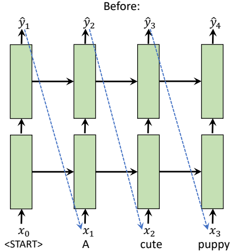


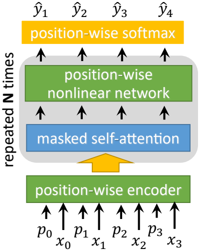


- Cross-attention was responsible for the condition and we remove it from the transformer decoder to get a language model
- We have masked self-attention as the transformer decoder has it and it prevents the circular dependency on future words
- But we don't have the cross-attention anymore

Source: CS W182 course, Sergey Levine, UC Berkeley

BERT

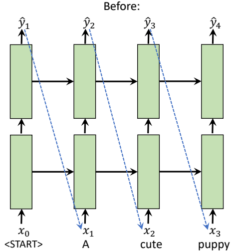


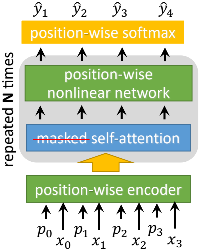


- Cross-attention was responsible for the condition and we remove it from the transformer decoder to get a language model
- We have masked self-attention as the transformer decoder has it and it prevents the circular dependency on future words
- But we don't have the cross-attention anymore
- This direct way of repalcing LSTM in ELMO with transformer decoder is not bidirectional though
- We could train two transformers and make "transformer ELMO"

Source: CS W182 course, Sergey Levine, UC Berkeley

BERT

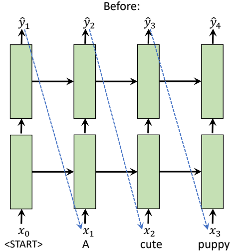


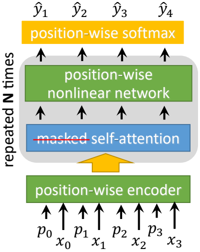


§ But is there a better way? Can we simply remove the mask in self-attention and have one transformer?

Source: CS W182 course, Sergey Levine, UC Berkeley

BERT

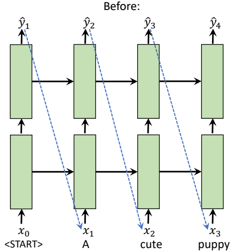


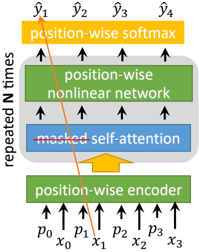


- But is there a better way? Can we simply remove the mask in self-attention and have one transformer?
- What could go wrong?

Source: CS W182 course, Sergey Levine, UC Berkeley

BERT

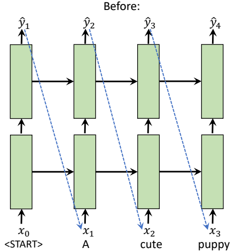


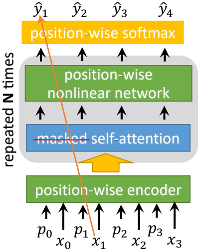


- But is there a better way? Can we simply remove the mask in self-attention and have one transformer?
- What could go wrong?
- The model can very easily learn a shortcut to get the right answer. The "right answer" at time t is same as the input at time $t+1$!

Source: CS W182 course, Sergey Levine, UC Berkeley

BERT

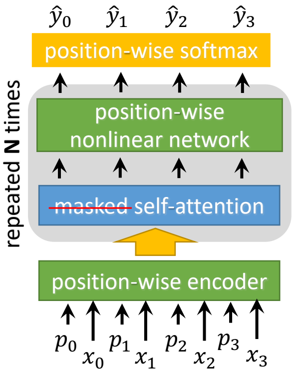




- But is there a better way? Can we simply remove the mask in self-attention and have one transformer?
- What could go wrong?
- The model can very easily learn a shortcut to get the right answer. The "right answer" at time t is same as the input at time $t+1$!
- BERT has to modify the training procedure slightly to avoid this trivial solution

Source: CS W182 course, Sergey Levine, UC Berkeley

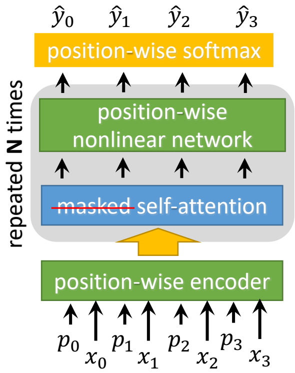
BERT



- The first thing is that there is no shifting in output. The output at timestep t is exactly same as the input at timestep t

Source: CS W182 course, Sergey Levine, UC Berkeley

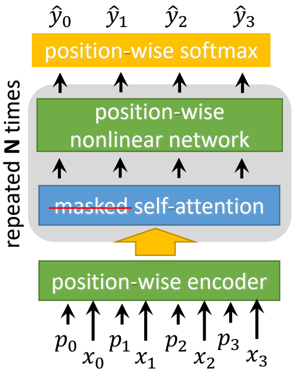
BERT



- The first thing is that there is no shifting in output. The output at timestep t is exactly same as the input at timestep t
- But the input is modified a little bit to make the task harder for the decoder
- Randomly mask out some input tokens where 'masking' means replacing the token with a special token denoted as [MASK]
- However, the output reamins the same

Source: CS W182 course, Sergey Levine, UC Berkeley

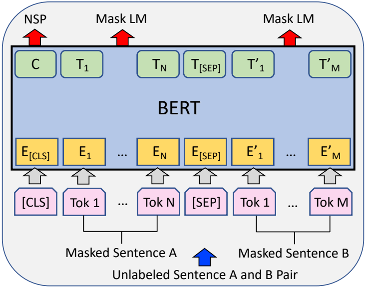
BERT



- The first thing is that there is no shifting in output. The output at timestep t is exactly same as the input at timestep t
- But the input is modified a little bit to make the task harder for the decoder
- Randomly mask out some input tokens where 'masking' means replacing the token with a special token denoted as [MASK]
- However, the output reamins the same
- **Input**: I [MASK] therefore I [MASK]
  Output: I think therefore I am
- This "fill in the blanks" task forces the model to *work hard* to learn a good representation
- At the same time, the absence of masked self-attention makes it **bidirectional**

Source: CS W182 course, Sergey Levine, UC Berkeley

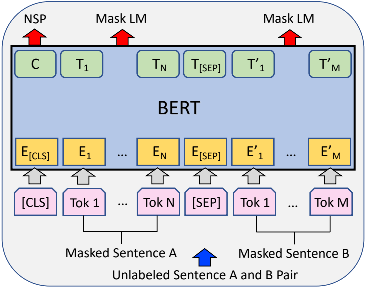
Training BERT



- BERT is trained with pairs of sentences
- The first sentence starts with the [CLS] token and second with [SEP] token

Source: CS W182 course, Sergey Levine, UC Berkeley

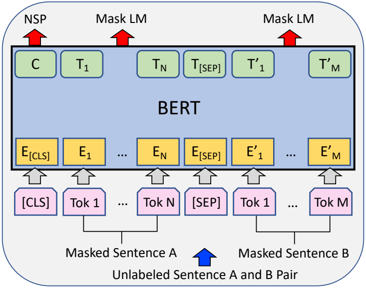
Training BERT



- BERT is trained with pairs of sentences
- The first sentence starts with the [CLS] token and second with [SEP] token
- Many NLP tasks involve two sentences - *e.g.*, question and answer, paragraph and summary *etc*. The idea is language model is accustomed with seeing such input pairs

Source: CS W182 course, Sergey Levine, UC Berkeley

Training BERT

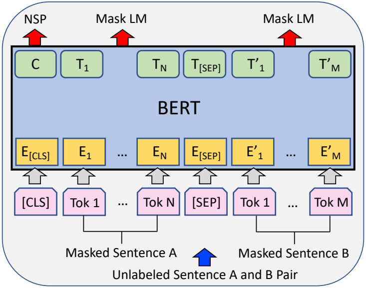


- BERT is trained with pairs of sentences
- The first sentence starts with the [CLS] token and second with [SEP] token
- Many NLP tasks involve two sentences - *e.g.*, question and answer, paragraph and summary *etc*. The idea is language model is accustomed with seeing such input pairs

- Input sentence pairs are transformed in two ways
  - Randomly replace 15% of the tokens with [MASK]
  - Randomly swap the order of the sentences 50% of the time

Source: CS W182 course, Sergey Levine, UC Berkeley

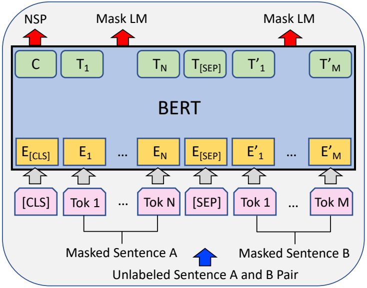
Training BERT



- BERT is trained with pairs of sentences
- The first sentence starts with the [CLS] token and second with [SEP] token
- Many NLP tasks involve two sentences - *e.g.*, question and answer, paragraph and summary *etc*. The idea is language model is accustomed with seeing such input pairs
- Input sentence pairs are transformed in two ways
  - Randomly replace 15% of the tokens with [MASK]
  - Randomly swap the order of the sentences 50% of the time
- The first input and first output are also special
  - The first input token is a special token [CLS]
  - The final hidden state corresponding to [CLS] is [NSP]. It predicts whether first sentence follows the second or vice-versa. It provides different ways to use BERT

Source: CS W182 course, Sergey Levine, UC Berkeley

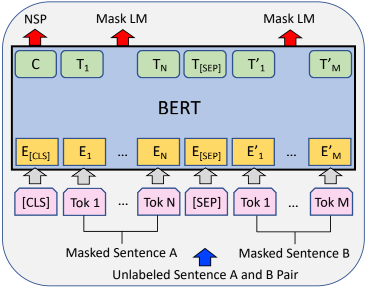
Using BERT



- If you have NLP tasks requiring the whole sentence representation, taking output from this [NSP] and replacing with task specific classifier does better job
- Some such examples are: Entailment classification, semantic equivalence, Sentiment classification *etc*.

Source: CS W182 course, Sergey Levine, UC Berkeley

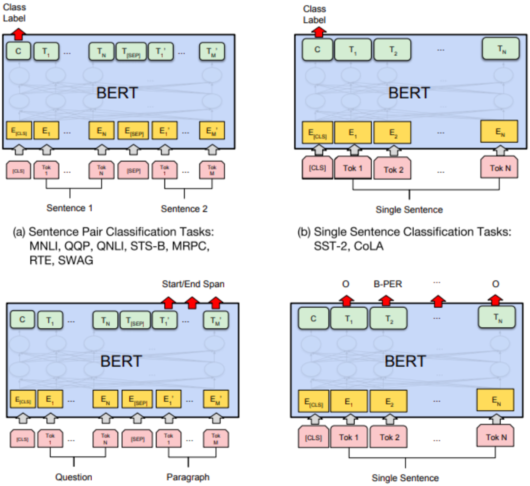
Using BERT



- If you have NLP tasks requiring the whole sentence representation, taking output from this [NSP] and replacing with task specific classifier does better job
- Some such examples are: Entailment classification, semantic equivalence, Sentiment classification *etc*.
  - Train BERT normally with huge corpus of unlabeled text data
  - Put a crossentropy loss on only the first output (replaces the sentence order classifier)
  - Finetune whole model end-to-end on the new task

Source: CS W182 course, Sergey Levine, UC Berkeley

Using BERT



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

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

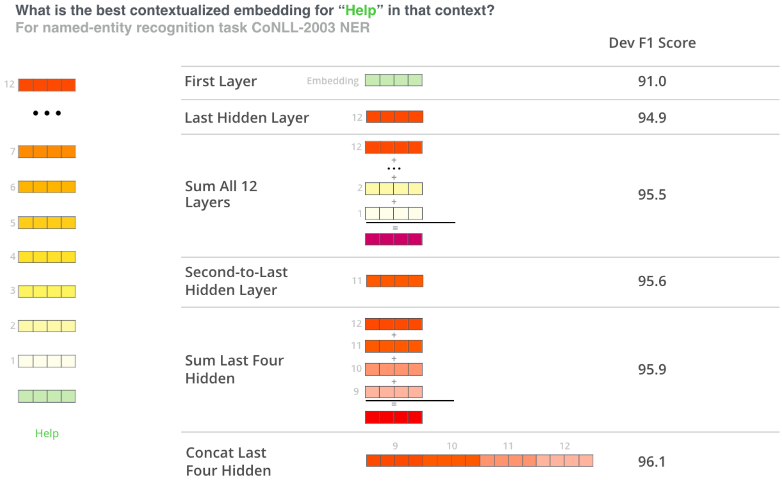
(c) Question Answering Tasks: SQuAD v1.1

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

Source: https://jalammar.github.io/illustrated-bert/

Using BERT

We can also pull out features, just like with ELMo!



Source: https://jalammar.github.io/illustrated-bert/

Using BERT

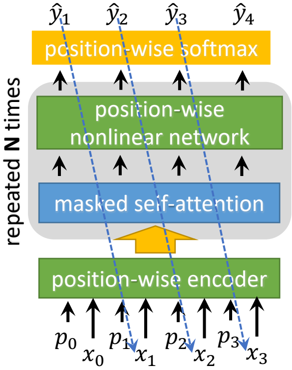
System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	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 General Language Understanding Evaluation (GLUE) benchmark is a collection of diverse natural language understanding tasks
- $\text{BERT}_{\text{BASE}}$ is 12 layers and $\text{BERT}_{\text{LARGE}}$ is 24 layers
- Since its inception, BERT has been applied to many NLP tasks and that often makes a huge difference in performance

Source: BERT Paper

GPT et al.



- People have also used one-directional (forward) type transformer models. It does have one big advantage over BERT
- Generation is not really possible with BERT, but a forward (masked attention) model can do it!
- GPT (GPT-2, GPT-3 *etc.*) is a classic example of this

Source: CS W182 course, Sergey Levine, UC Berkeley

Pretrained Language Models Summary

§ BERT

- BERT is a 'bidirectional' transformer
- Trained with masked out tokens as a fill-in-the-blank task
- + Great representations
- - Can't generate texts

Source: CS W182 course, Sergey Levine, UC Berkeley

Pretrained Language Models Summary

§ BERT

- BERT is a 'bidirectional' transformer
- Trained with masked out tokens as a fill-in-the-blank task
- + Great representations
- - Can't generate texts

§ OpenAI GPT

- GPT is an one dimensional transformer
- Transformer decoder without cross-attention and with masked self-attention
- + Can generate texts
- - Ok representations

Source: CS W182 course, Sergey Levine, UC Berkeley

Pretrained Language Models Summary

§ BERT

- BERT is a ‘bidirectional’ transformer
- Trained with masked out tokens as a fill-in-the-blank task
- + Great representations
- - Can’t generate texts

§ OpenAI GPT

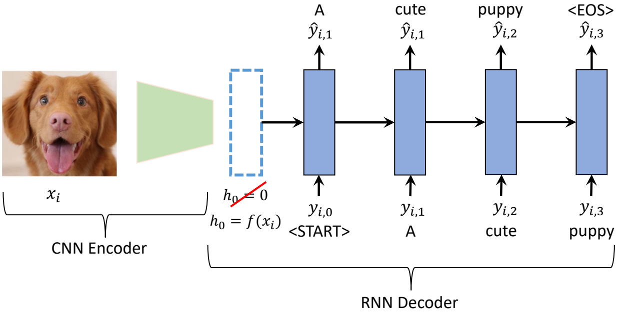
- GPT is an one dimensional transformer
- Transformer decoder without cross-attention and with masked self-attention
- + Can generate texts
- - Ok representations

§ ELMO

- Bidirectional LSTMs
- ELMO trains two separate LSTM language models
- - Ok representations
- Largely supplanted by BERT

Source: CS W182 course, Sergey Levine, UC Berkeley

Image Captioning



Source: CS W182 course, Sergey Levine, UC Berkeley

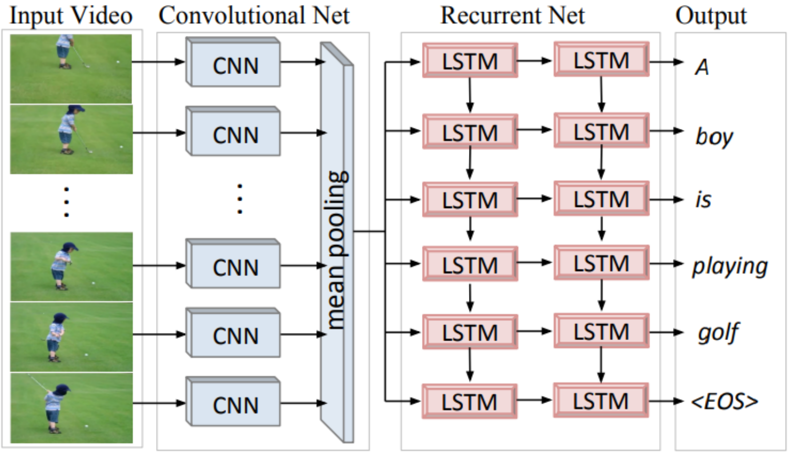
Video Captioning

§ Example from MSR-VTT Dataset

Figure 1. Examples of the clips and labeled sentences in our MSR-VTT dataset. We give six samples, with each containing four frames to represent the video clip and five human-labeled sentences.

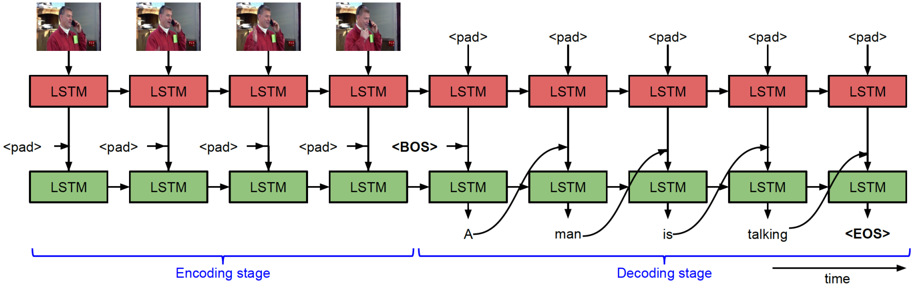
Source: J Xu, T Mei, T Yao and Y Rui, 'MSR-VTT: A Large Video Description Dataset for Bridging Video and Language', CVPR 2016

Video Captioning



Source: S Venugopalan *et al.* 'Translating Videos to Natural Language Using Deep Recurrent Neural Networks', NAACL 2015

Video Captioning



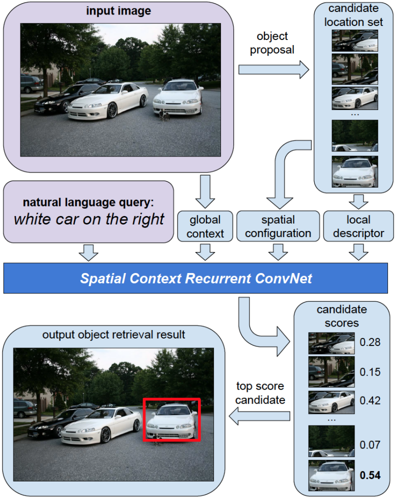
Source: S Venugopalan *et al.* 'Sequence to Sequence – Video to Text', ICCV 2015

Natural Language Object Retrieval

query=*'man in middle with blue shirt and blue shorts'*

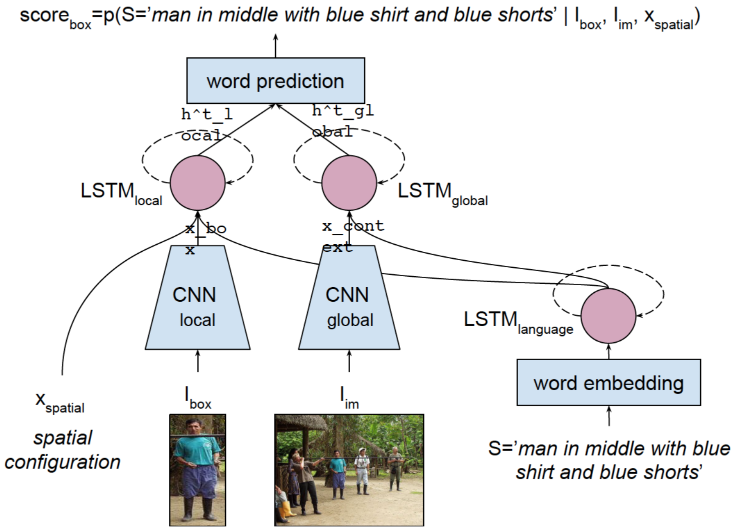
Source: R Hu *et al.* 'Natural Language Object Retrieval', CVPR 2016

Natural Language Object Retrieval



Source: R Hu *et al.* 'Natural Language Object Retrieval', CVPR 2016

Natural Language Object Retrieval



Source: R Hu *et al.* 'Natural Language Object Retrieval', CVPR 2016

Natural Language Object Retrieval

At test time, given an input image I, a query text S and a set of candidate bounding boxes $\{b_i\}$, the query text S is scored on i-th candidate box using the likelihood of the query text sequence conditioned on the local image region, the whole image and the spatial configuration of the box, computed as

$$\begin{aligned} s &= p(S|I_{box}, I_{im}, x_{spatial}) && (8) \\ &= \prod_{w_t \in S} p(w_t|w_{t-1}, \cdots, w_1, I_{box}, I_{im}, x_{spatial}) && (9) \end{aligned}$$

and the highest scoring candidate boxes are retrieved.

Source: R Hu *et al.* 'Natural Language Object Retrieval', CVPR 2016

Natural Language Object Retrieval

Source: R Hu *et al.* 'Natural Language Object Retrieval', CVPR 2016