Natural Language Processing

CSC413/2516

UNIVERSITY OF TORONTO

Big Picture

- Central to human intelligence.
- Tremendous practical value.
- Colossal developments recently.

Goal of This Tutorial

- Basic NLP literacy.
- Getting up to date with recent developments.
  - Architectures
  - Language tasks
  - Tremendous developments in the field recently!
- Know where to look at if you're starting an NLP project

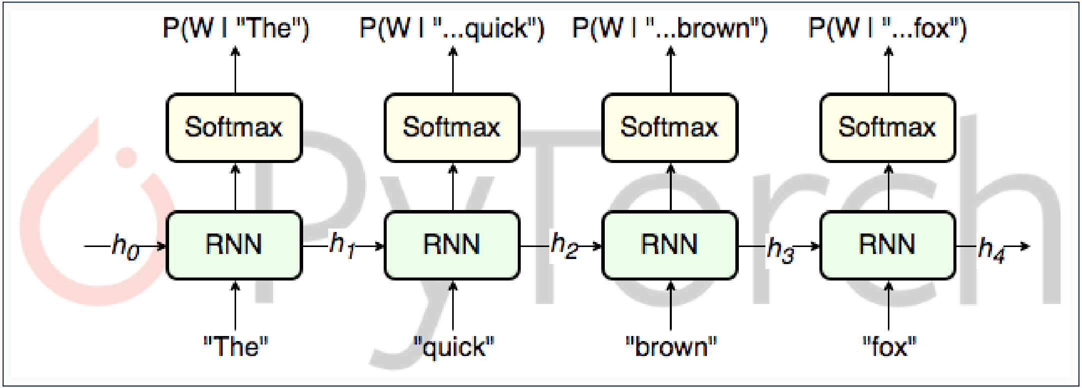
Language Models

- A statistical model that assigns probabilities to the words in a sentences.
- **Most commonly**: Given previous words, what should the next one be?
- **Neural language model**: Model the probability of words given others using neural networks.

Architectures

Which architecture is most suitable?

- We can use recurrent architectures.
- LSTM, GRU ...
- Great for variable length inputs, like sentences.



Recurrent Architectures

What are some of the problems with recurrent architectures?

What are some of the problems with recurrent architectures?

- Not parallelizable across instances.
- Cannot model long dependences.
- Optimization difficulties (vanishing gradients).

We'd like an architectural primitive that is:

- Ideally feed-forward
- Can facilitate between-token interactions
- Can model long dependences easily.

Attention to the rescue!

- There are many forms of attention. Today we'll focus on **scaled dot product attention.**

Attention

- Three inputs: queries, keys and values.
- ”Return a combination of the values based on the similarities between keys and queries”.
- Dimensionalities:
  - $Q \in \mathbb{R}^{n_q x d_{kq}}$
  - $K \in \mathbb{R}^{n_{kv} x d_{kq}}$
  - $V \in \mathbb{R}^{n_{kv} x d_v}$
- $$\mathbb{A}(Q, K, V) = V(softmax(\frac{K^T Q}{d_{kq}})) \tag{1}$$

Quiz!

- True of False: The dimensionality of queries have to match the dimensionality of the keys.

Quiz!

- True of False: The dimensionality of queries have to match the dimensionality of the keys. TRUE
- True or False: The number of keys have to match the number of values.

Quiz!

- True of False: The dimensionality of queries have to match the dimensionality of the keys. TRUE
- True or False: The number of keys have to match the number of values. TRUE
- True or False: The number of keys have to match the number of queries.

Attention

Quiz!

- True of False: The dimensionality of queries have to match the dimensionality of the keys. TRUE
- True or False: The number of keys have to match the number of values. TRUE
- True or False: The number of keys have to match the number of queries. FALSE

Attention

Quiz!

- True of False: The dimensionality of queries have to match the dimensionality of the keys. TRUE
- True or False: The number of keys have to match the number of values. TRUE
- True or False: The number of keys have to match the number of queries. FALSE
- Dimensionalities:
  - $Q \in \mathbb{R}^{n_q x d_{kq}}$
  - $K \in \mathbb{R}^{n_{kv} x d_{kq}}$
  - $V \in \mathbb{R}^{n_{kv} x d_v}$

Attention

We need to make two central decisions:

- How do we compute similarity?
- How do we ’normalize’ the similarity scores amongst values?

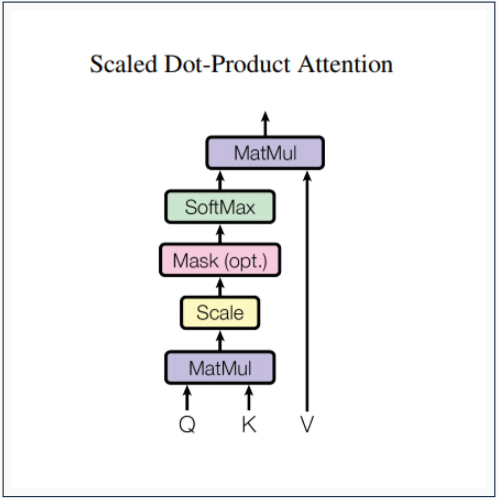
Attention

- **Similarity:** Dot product between keys and queries.
- **Interesting theorem:** In high dimensions, two randomly sampled [1] vectors are almost always approximately perpendicular to each other.
- **Normalization:** Softmax along the keys/values!
- **Result:** Scaled dot product attention.
- We get the following attention mechanism:

$$\mathbb{A}(Q, K, V) = V(softmax(\frac{K^T Q}{d_{kq}})) \tag{2}$$

[1] From, lets say, a isotropic multivariate Gaussian distribution.

Attention


Scaled Dot-Product Attention
MatMul
SoftMax
Mask (opt.)
Scale
MatMul
Q
K
V

Self-Attention

- What is self-attention?

Self-Attention

- What is self-attention?
- Use the same tensor for keys, values and queries!

Self-Attention

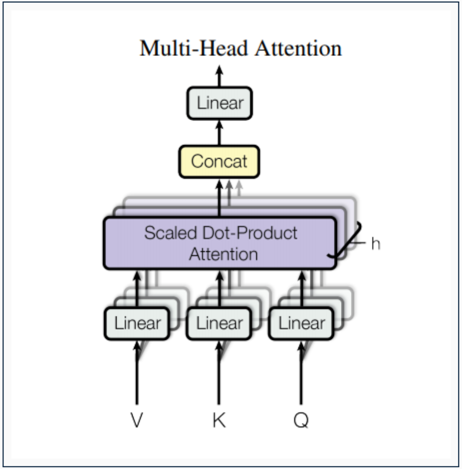
- What is self-attention?
- Use the same tensor for keys, values and queries!
- What are the keys/queries/values in a self attention layer processing sentence

Self-Attention

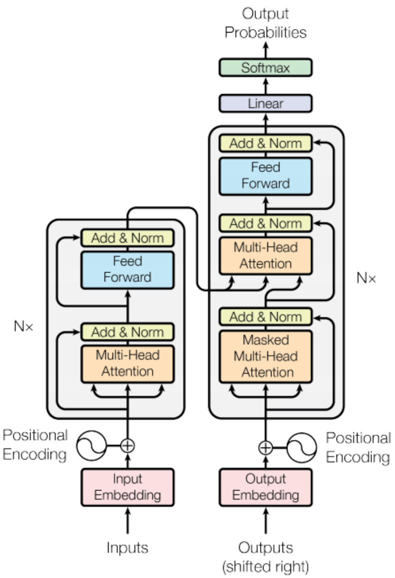
- What is self-attention?
- Use the same tensor for keys, values and queries!
- What are the keys/queries/values in a self attention layer processing sentence
- The features corresponding to each token!

- Lingering question: What is learned in an attention layer?
  - The space in which the similarities are computed.
  - The transformations on the values.
- What if we'd like to have different notions of similarity on the same set of tokens?
- **Multi head attention** to the rescue!

Attention


Multi-Head Attention
Linear
Concat
Scaled Dot-Product Attention
h
Linear
Linear
Linear
V
K
Q

Transformers



Properties of the transformer architecture:

- Fully feed forward.
- Equivariance properties of scaled dot product attention (important):
  - How does the output change if we permute the order of queries? (equivariance)
  - How does the output change if we permute the key-value pairs in unison? (invariance)

Exercise: How about self-attention?

Performance Comparison

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

What's next?

What's next?

- Brief intro to self supervised learning
- BERT (i.e. Self-supervised training of language models)

Self Supervised Learning

Three types of learning:

- Supervised learning
- Reinforcement learning
- **Unsupervised/self-supervised learning**:
  - When the label is in the data itself!
  - Possible to make use of large amounts of data with no additional labelling efforts.

Self Supervised Learning

Examples of self-supervised learning:

- Predict next frame in a video.
- Image completion.
- Auto-encoding tasks.
- Rotation prediction.
- **Predicting next word from previous ones**.

Pretraining Language Models

- Can we use large amounts of text data to pretrain language models?

Pretraining Language Models

- Can we use large amounts of text data to pretrain language models?
- Considerations:
  - How can we fuse both left-right and right-left context?
  - How can we facilitate non-trivial interactions between input tokens?

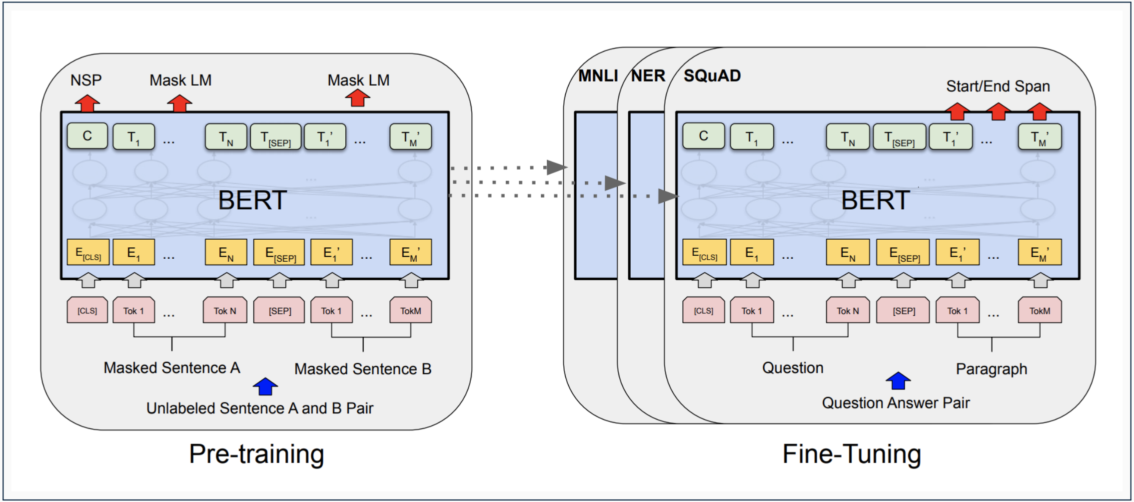
Pretraining Language Models

- Can we use large amounts of text data to pretrain language models?
- Considerations:
  - How can we fuse both left-right and right-left context?
  - How can we facilitate non-trivial interactions between input tokens?
- Previous approaches:
  - ELMO (Peters. et. al., 2017): Bidirectional, but shallow.
  - GPT (Radford et. al., 2018): Deep, but unidirectional.
  - BERT (Devlin et. al., 2018): Deep and bidirectional!

BERT Workflow

- The BERT workflow includes:
  - Pretrain on generic, self-supervised tasks, using large amounts of data (like all of Wikipedia)
  - Fine-tune on specific tasks with limited, labelled data.
- The pretraining tasks (will talk about this in more detail later):
  - Masked Language Modelling (to learn contextualized token representations)
  - Next Sentence Prediction (summary vector for the whole input)

BERT Architecture



BERT Architecture

Properties

- Two input sequences.
  - Many NLP tasks have two inputs (question answering, paraphrase detection, entailment detection etc.)
- Computes embeddings
  - Both token, position and segment embeddings.
  - Special start and separation tokens.
- Architecture
  - Basically the same as transformer encoder.
- Outputs:
  - Contextualized token representations.
  - Special tokens for context.

BERT Embeddings

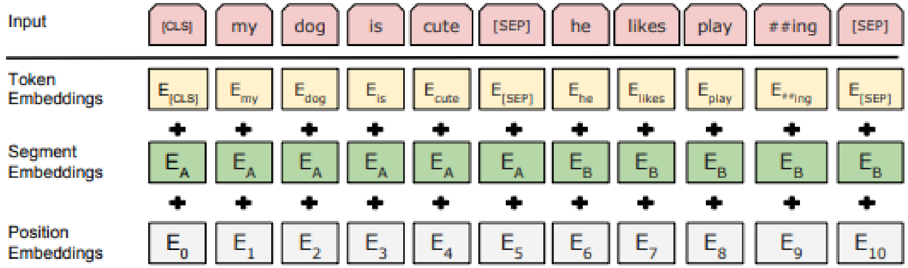


Figure 2: BERT input representation. The input embeddings is the sum of the token embeddings, the segmentation embeddings and the position embeddings.

- How we tokenize the inputs is very important!
- BERT uses the WordPiece tokenizer (Wu et. al. 2016)

(Aside) Tokenizers

- Tokenizers have to balance the following:
  - Being comprehensive (rare words? translation to different languages)
  - Total number of tokens
  - How semantically meaningful each token is.
- This is an activate area of research.

Pretraining tasks

- Masked Language Modelling (i.e. Cloze Task (Taylor, 1953)
- Next sentence prediction

Masked Language Modelling

- Mask 15% of the input tokens. (i.e. replace with a dummy masking token)
- Run the model, obtain the embeddings for the masked tokens.
- Using these embeddings, try to predict the missing token.
- ”I love to eat peanut ___ and jam. ” Can you guess what’s missing?

Masked Language Modelling

This procedure forces the model to encode context information in the features of all of the tokens.

Next Sentence Prediction

- Goal is to summarize the complete context (i.e. the two segments) in a single feature vector.
- Procedure for generating data
  - Pick a sentence from the training corpus and feed it as ”segment A”.
  - With 50% probability, pick the following sentence and feed that as ”segment B”.
  - With 50% probability, pick the a random sentence and feed it as ”segment B”.
- Using the features for the context token, predict whether segment B is the following sentence of segment A.
- Turns out to be a very effective pretraining technique!

Fine Tuning

Procedure:

- Add a final layer on top of BERT representations.
- Train the whole network on the fine-tuning dataset.
- Pre-training time: In the order of days on TPUs.
- Fine tuning task: Takes only a few hours max.

Fine Tuning

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 "Average" column is slightly different than the official GLUE score, since we exclude the problematic WNLI set.[8] BERT and OpenAI GPT are single-model, single task. F1 scores are reported for QQP and MRPC, Spearman correlations are reported for STS-B, and accuracy scores are reported for the other tasks. We exclude entries that use BERT as one of their components.

Pointers

The transformer section of this tutorial is influenced by the fantastic talk by Lukasz Kaiser on transformers: https://www.youtube.com/watch?v=rBCqOTEfxvgt=1704s