Deep Learning

Language Models and Transformers

Edgar F. Roman-Rangel.
edgar.roman@itam.mx

Digital Systems Department.
Instituto Tecnológico Autónomo de México, ITAM.

March 12th, 2021.

NLP

Natural Language Processing (NLP).

Discipline dealing with the task of making computers process human language.

Understanding:

- Contents of text.
- Intention/sentiments.
- Languages (translation).

Requires *language models*.

Outline

Word Embedding

Transformers

BERT

Word representation

So far, we commented only on a couple of simple numeric representations of words:

One-hot encoding

E.g., for then 4-th word.

$$[0, 0, 0, 1, 0, 0].$$

Binary vector

E.g., for the 13-th word.

$$[0, 0, 1, 1, 0, 1].$$

Both require:

1. Knowing the total number of unique words in the corpus.
2. Ordering all unique words somehow, e.g., alphabetically.
3. Assigning an integer number to each unique word.
4. Representing integer indices as one-hot encoding or binary vectors.

Word embedding

Neither one-hot encoding nor binary vectors consider semantics.

Word embedding: Numeric representation (vector) of words, in which words with similar meaning result in similar representation.

They allow for vector space models (VSM).

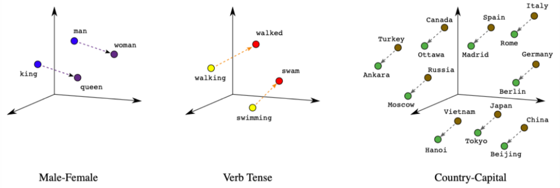


Image Source: (Embeddings: Translating to a Lower-Dimensional Space) by Google.

$$\mathbf{v}_{\text{king}} - \mathbf{v}_{\text{man}} + \mathbf{v}_{\text{woman}} = \mathbf{v}_{\text{queen}}.$$

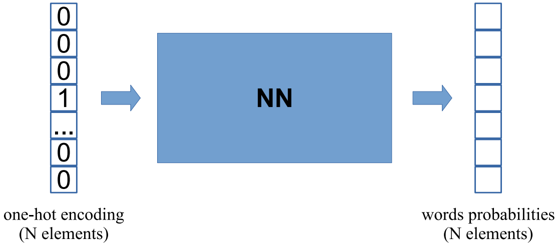
Features for a given word must be extracted from the context of the word itself.

Word2Vec

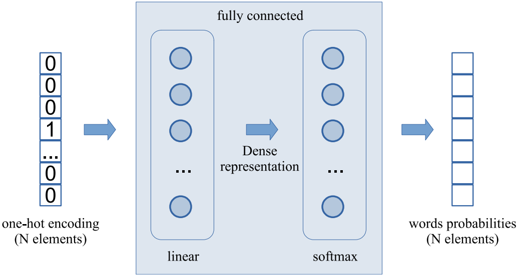
Mikolov et al., 2013. "Efficient Estimation of Word Representations in Vector Space".

Given a set of phrases, let us analyze their words, and for each word learn to predict a probability of the following word. E.g.,

`The quick brown fox __?__ over the lazy dog.`



Word2Vec network



Up to us to define the length of the word embedding.

- Word embedding: $\mathbf{h} = \mathbf{x}^T \mathbf{w}_e$,
- Next word probability: $\mathbf{y} = a(\mathbf{h}^T \mathbf{w}_o)$, where $a(\cdot)$ is the softmax function.

There are two main variants: CBOW and Skip-gram.

Word2Vec variants

CBOW: continuous bag-of-words

Predict target word from context, e.g.,

`The quick brown fox __?__ over the lazy dog.`

Skip-gram

Predict context words from a target, e.g.,

`__?__ __?__ __?__ __?__ jumps __?__ __?__ __?__ __?__.`

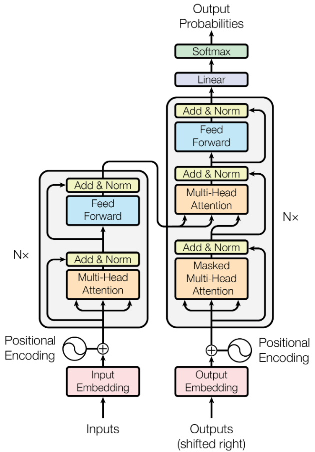
Other models

- GloVe.
- fastText.
- Sentence2Vec.
- Doc2Vec.

Outline

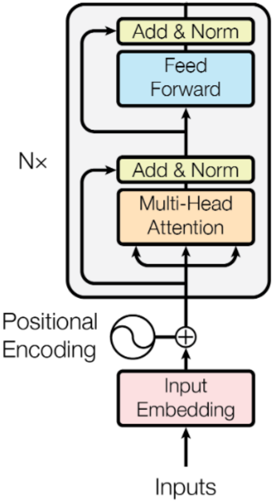
Intro

Vaswani et al., 2017. "Attention Is All You Need".



- DL model for sequence data.
- Encoder-Decoder architecture.
- Avoids recurrence.
- Exploits temporal dependence.
- Parallel transformations.
- Can look ahead in time.
- Uses self-attention mechanisms.
- Designed for seq2seq problems, e.g., text completion or translation.

Encoder



1. Inputs/outputs.
2. Embedding.
3. Positional encoding.
4. Self-attention.
5. Multi-head attention.
6. Residual connections.
7. Feed forward.

Inputs/outputs

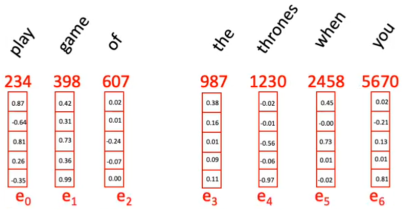
Text split into two parts: x beginning, and y ending.

Beginning:	**Ending:**
A robot must obey the orders	<start> given by human beings <end>
Once you have eliminated the impossible, whatever remains,	<start> however improbable, must be the truth <end>
If he's so smart, how come	<start> he's dead? <end>
When you play the game of thrones	<start> you win or you die <end>

Words are passed as integer indices.

Embedding layer

- Maps integer indices into one-hot encoding vectors.
- Then into dense vector representation, e_t.
- Embedding matrix is a set of weights.
- These weights are also learned during training.



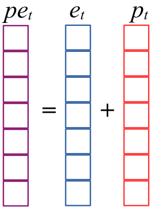
Authors use vectors of length 512.

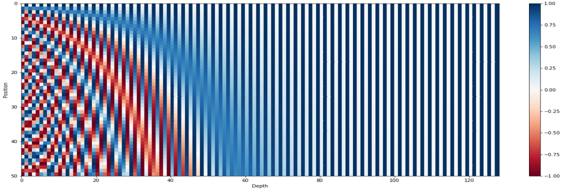
Positional encoding layer

Used to compensate for the lack of recurrence operations. (Transformers process all embeddings at once: parallel).

- Weights embedding vectors e_t with positional encodings p_t.
- Both e_t and p_t have the same length d.
- Position information is used through wave frequencies.

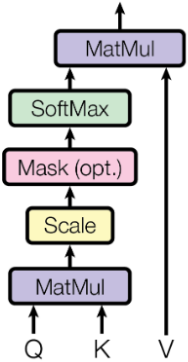
$$p_{(t,2i)} = \sin\left(\frac{t}{10000^{2i/d}}\right), \qquad p_{(t,2i+1)} = \cos\left(\frac{t}{10000^{2i/d}}\right).$$



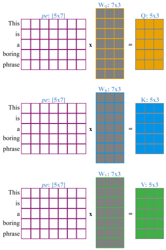


Self-attention

Figuring out how important all the other words in the sentence are w.r.t. to a word of interest.



- Q : query, K : key, V : value. All three are linear transformations of the pe vector (*fc*).
- Can be used to shrink vectors.



Attention filter:

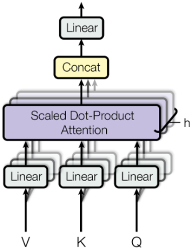
$$\mathbf{A} = \mathrm{s}\left(\frac{\mathbf{QK}^T}{\sqrt{d}}\right).$$

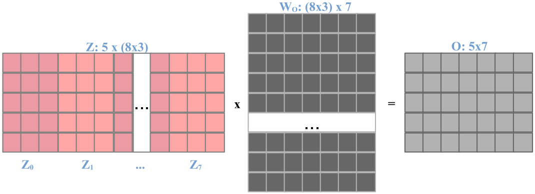
Self-attention score:

$$\mathbf{Z} = \mathbf{AV}.$$

Multi-head attention

Multiple self-attention paths in parallel .





Residual connections

Add & Norm layer

1. Adds $pz = pe + Z$ representations.
2. Gaussian normalization per row.

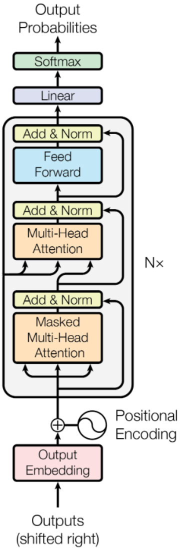
$$o_i(t) = \frac{pz_i(t) - \mu(t)}{\sqrt{\sigma^2(t) + \epsilon}},$$

- (t) represents time step (word), i.e., row.
- i is the index for the feature in the embedding space.
- $\mu(t)$ and $\sigma(t)$: mean and standard deviation of the t-th row.

Final feed forward

Stack of linear combinations plus ReLU activation functions.

Decoder



- Looks quite similar to the encoder.
- Takes two inputs: o and $\hat{y}(t)$:
  encoder's output and decoder's output up to t.
- Multi-head attention.
- Masked multi-head attention.

1. Receives one word at a time (*ending* phrase).
2. Starts with token $<$start$>$.
3. Every time, its input $\hat{y}(t)$ grows by one row.

Decoder main path

Multi-head attention

Encoder's output o is split into Q and K for the decoder's input.

Masked multi-head attention

Tokens go through embedding and masked multi-head attention layers to generate a value representation V.

More processing

Q, K and V are combined the same way as previously.

Final linear combination

- Flattens the hidden representation.
- Passes to a set of fully connected perceptrons (as many as the number of words in the corpus).
- Softmax-normalize the output: probability of next word.

Masked multi-head attention

- Attention filter of the *ending* phrase is masked during training.
- Forces the model to consider only past words.

	<start>	I	am	no	man	<end>
<start>	1	0	0	0	0	0
I	0.01	0.99	0	0	0	0
am	0.001	0.004	0.995	0	0	0
no	0.003	0.004	0.003	0.99	0	0
man	0.003	0.003	0.04	0.02	0.93	0
<end>	0.001	0.001	0.001	0.001	0.001	0.995

Variants

- Performer.
- Linformer.
- Reformer.
- BERT.
- BERTa.
- RoBERTa.
- ELMO.
- GPT2 and GPT3.

Outline

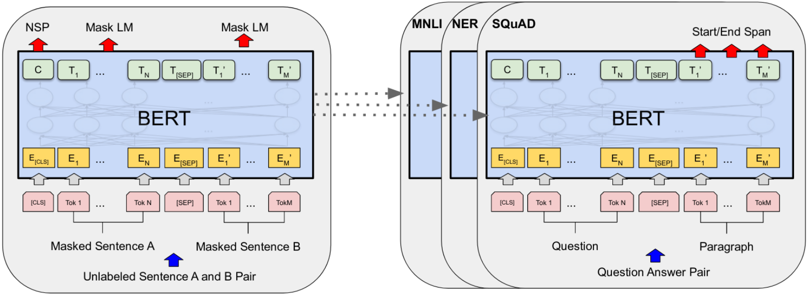
Intro

Devlin et al., 2019. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding".

BERT: Bidirectional Encoder Representations from Transformers.

- Pretrained transformer for language models.
- Only the transformer's decoder part.
- Bidirectional (no masking of future words).
- Instead, masking words out of the input sentences.
- Trained on a large corpus.
- Transfer learning - model released to github and tensorflow.
- State-of-the-art for NLP applications (16,821 citations).

Model training



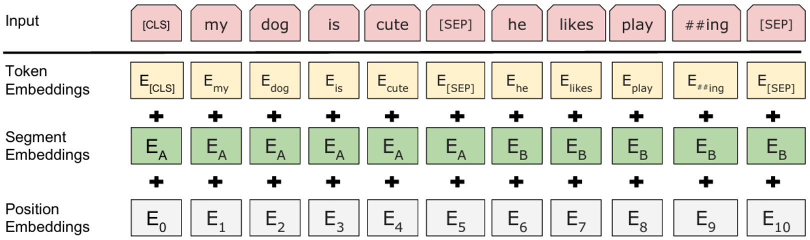
Pre-training Fine-Tuning

Two tasks.

- Masked language model (pdf over masked words).
- Next sentence prediction (binary classification).

Input and embedding

Combines three types of embedding: input embedding, segment embedding, and positional embedding.



Q&A

Thank you!

edgar.roman@itam.mx