Multilingual BERT

MICKUS, Timothee

General layout of this lecture :

1. Introduction to contextual embeddings
2. Introduction to the Transformer architecture used by BERT
3. Closeup on BERT's training
4. Training BERT in a multilingual setting

Where do contextual embeddings come from ?

Word Embeddings

A very general timeline

The general idea has always been to turn a word into a dense vector of real value. Theoretical works generally stress a connection with the distributional hypothesis (Firth, 1957)

- stems from information retrieval ('70s)
- first usage of word vectors as "distributional semantics" in the '90s
- first **neural** embeddings in 2003
- wide-spread use of embeddings from 2013 onward
- first **contextualized** neural embeddings 2017

Word Embeddings

The Rise of Contextual Embeddings

Embeddings for words in context. The trend mostly caught on in 2018

- CoVe McCann et al. (2017)
- ELMo Peters et al. (2018)
- OpenAI GPT Radford (2018)
- BERT Devlin et al. (2018)

Explosive gains across multiple NLP tasks

- but we don't *really* know how they work

Contextual embeddings

What changed : from words to sentences

Peters et al. (2018) :

> *Unlike most widely used word embeddings, [...] [contextual] word representations are functions of the entire input sequence*

Contextualized representations guarantee a bijection between sequences of words and sequences of vectors, not between words and vectors individually.

- Has interesting consequences, such as the fact that the sum of all vectors for a sentence is sensitive to order ($\neq$ BoW)

Unlike sentence encoders, which merge together in a single vector all the semantics of the sentence, contextualized embedding algorithm assign to each token a representation that is a function of the entire input sentence.

Contextual embeddings

What changed : fine-tuning vs. feature-based models

Devlin et al. (2018)

- It is now possible to achieve state-of-the-art performance on multiple tasks by simply fine-tuning the embeddings model.
- Contrasts with previous non-contextualized embeddings which were most of the time used as additional features for more complex, often task-specific models (NB : still possible with contextualized representations)

Meet BERT.

The BERT hype

- BERT is a contextualized embedding algorithm designed to assign a sequence of vectors to a sequence of words
- BERT is designed to be used as generally as possible
- BERT is based on the Transformer architecture, which is trendy but pretty much not understood
- BERT is trained on two tasks at once :
  - word-level MLM, derived from a standard psychology test
  - sentence-level Next Sentence Prediction, which allows for sentence relationship awareness
- BERT has dominated many benchmarks.

BERT is a Transformer

BERT (Devlin et al., 2018) is "basically" a simplified encoder from a Transformer (Vaswani et al., 2017)

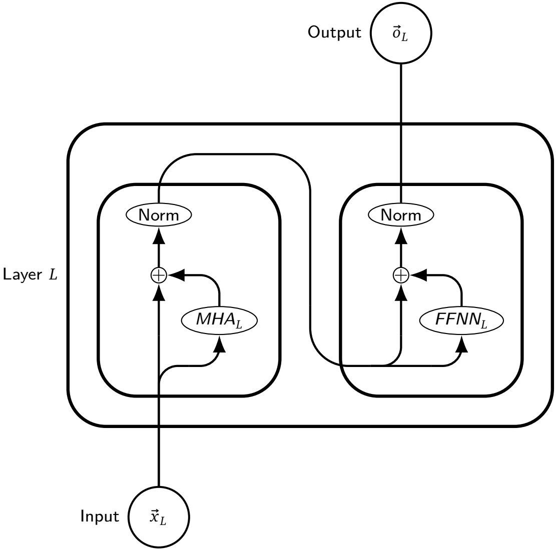
- A Transformer encoder is a stack of L layers divided into two sublayers, each using residual connection and normalisation

  $$\text{SubLayer} = \text{Norm}(x + F(x))$$

  Informally, residual connections allow the upper layers to still retain some information from the input, whereas normalisation ensure that intermediate representations have a similar scale

BERT is a Transformer

Transformer Layer at a glance



BERT is a Transformer

- The first sublayer applies scaled-dot self-attention ; ie. weighting of attended vectors (V) based on a probability distribution (Softmax(...)) of the dot product ($Q \cdot K^T$), taking into the expected standard deviation ($\sqrt{d_K}$) :

$$\text{Attention}(Q, K, V) = \text{Softmax}(\frac{Q \cdot K^T}{\sqrt{d_K}})V$$

- ... combined with multi-head attention, ie. each attention sublayer has A learned linear projections for queries Q, keys K and values V

$$\text{MultiHead}(Q, K, V) = \bigoplus_{a}^{A} \text{Attention}(W_q^a Q, W_k^a K, W_v^a V)$$

  where $\oplus$ denotes concatenation

- Queries Q, keys K and values V correspond (in our case) to the previous layer's output.

BERT is a Transformer

- The second sublayer is a feed forward network, composed of two linear transformations with a rectified linear unit activation in between :

$$(x) = \mathrm{ReLU}(xW_1 + b_1)W_2 + b_2$$

- The systems uses learned embeddings to convert the input tokens.
- To provide the model with information relative to the position of a word in a sequence, position encoding vectors are added to the corresponding embeddings :

$$\mathrm{PositionEncoding}(\mathrm{pos}) = \langle \overrightarrow{c(\mathrm{pos}, 1),\ \ldots,\ c(\mathrm{pos}, d_e)} \rangle$$

$$\vec{i}_{\mathrm{w,p}} = e(w) + \mathrm{PositionEncoding}(p)$$

where each component of the position encoding vector is defined using :

$$c(\mathrm{pos}, \mathrm{dim}) = \begin{cases} \sin(\frac{\mathrm{pos}}{10000^{\mathrm{dim}/d_e}}) \text{ if dim} = 2k \\ \cos(\frac{\mathrm{pos}}{10000^{\mathrm{dim}/d_e}}) \text{ otherwise.} \end{cases}$$

In other words the position encoding vectors are **fixed**.

BERT is a Transformer

The Transformer (more precisely its encoder) depends mostly on three hyperparameters :

- L, the number of layers
- A, the number of attention heads
- H, the dimensionality of the hidden representations

Various transformers have various hyperparameters settings :

- the original transformer by Vaswani et al. (2017) was $L = 6$, $H = 512$, $A = 8$
- BERT-Base is $L = 12$, $H = 768$, $A = 12$
- BERT-Large is $L = 24$, $H = 1024$, $A = 16$

Train your own BERT !

How is BERT trained?

Other than dropping the decoder altogether, BERT has very few amendments to the original Transformer algorithm

- the most important change is its learned sentence-specific embeddings (or 'segment' embeddings), which are used for the sentence-level objective (we'll get to it later)
- Some other minor changes involve the **systematic** use of word-piece to tokenize the input text.
- BERT uses GELU rather than ReLU activation

How is BERT trained ?

BERT is trained on two objectives simultaneously

- A word-level objective
- A sentence-level objective

How is BERT trained ?

MLM, aka. Cloze Test

The word-level objective for BERT comes from psychology (Taylor, 1953)

- Introducing the "Cloze Test", also known as "Gap-Fill", "Cloze deletion test", "Fill in the blanks"...
- In a given sentence a word (or a group of words) will be blanked out
- Subjects will then be tasked with filling in said blanks.

It is mostly used as learning exercises to assess **reading proficiency** and **mastery of grammar**. It has also been used jointly with eye-tracking.

How is BERT trained ?

Implementing the Cloze Test as an objective

- The idea behind BERT is to train the Transformer architecture to do well on Cloze Test : if it can find the correct parameters to solve a reading exercise, then it's probably a decent textual representation.
- To do so, we need to formulate the Cloze as a task
- The task will be predict correctly an item that has been 'blanked out'.
- The prediction can be done using a simple softmax layer to which is fed the embedding of the blanked-out item.

This use of the Cloze Test as a training task was dubbed by the authors the 'Masked Language Model' task, or MLM for short.

How is BERT trained ?

MLM, concretely

More concretely :

- The model first randomly selects 15% of the word-pieces, which will be fed to the softmax prediction layer.
- 80% of the randomly selected items (= 12% of the word-pieces in total) will be replaced by a special token `[MASK]` representing a blank
- 10% of the randomly selected word-pieces (= 1.5% of the word-pieces in total) will be replaced by a word at random. This is done to mitigate the mismatch between pre-training and fine-tuning further down the line, since the special token `[MASK]` will never be encountered during fine-tuning.
- 10% of the randomly selected word-pieces (= 1.5% of the word-pieces in total) will be replaced by a word at random. This is done to "bias the representation towards the actual observed word".

How is BERT trained ?

Sentence-level objective

- We mentioned earlier that BERT had two objectives, the second being sentence-level
- This second objective is to predict whether a sentence immediately another in the corpus ; it has been prosaically dubbed the "next sentence prediction" task
- This objective entails that BERT can only be trained on a corpus of coherent documents, and not on corpora composed of shuffled sentences
- This second objective helps a lot on QA and NLI downstream tasks.

How is BERT trained ?

Next sentence prediction

- This is naturally as a binary classification of paired sentences $\langle S_A, S_B \rangle$ between two labels `IsNext` and `NotNext`.
  - The first label `IsNext` corresponds to when S_A is immediately followed by S_B in the training corpus
  - The second label `NotNext` corresponds to when S_A and S_B were just randomly and separately sampled from the corpus and paired together.
- Sentences are presented as a contiguous span of text to the system, using two special tokens `[CLS]` and `[SEP]` as separators.
  More concretely, if $S_A = w_1^A, \ldots, w_n^A$ and $S_B = w_1^B, \ldots, w_m^B$, the system will receive the following sequence as input :
  $\texttt{[CLS]}, w_1^A, \ldots, w_n^A, \texttt{[SEP]}, w_1^B, \ldots, w_m^B, \texttt{[SEP]}$
- To further facilitate the models ability to distinguish two sentences, learned sentences embeddings for S_A and S_B are added respectively to $\texttt{[CLS]}, w_1^A, \ldots, w_n^A, \texttt{[SEP]}$ and to $w_1^B, \ldots, w_m^B, \texttt{[SEP]}$
- Although not specified in the paper, the sentence prediction only uses the `[CLS]` token for its prediction.

How is BERT trained ?

Recap : BERT input format

BERT is a Transformer, trained on the "next sentence prediction" objective (viz. '*does the 2nd sentence of the input follow the 1st ?*')

1. tokens are first embedded
2. 'positional encodings' $\vec{p(i)}$ mark the position i of the token
3. 'Segment encodings' $\vec{\mathtt{seg}}_A$, $\vec{\mathtt{seg}}_B$ mark which sentence tokens belong to
4. 2 special tokens : `[SEP]` for sentence boundaries, `[CLS]` for performing the actual prediction

How is BERT trained ?

Recap : BERT input format illustrated

Given the example "*My dog barks. It is a pooch.*", the actual input would be :

$$
\begin{aligned}
\vec{\texttt{[CLS]}} + \vec{p(0)} + \vec{\texttt{seg}}_A, &\quad \vec{My} + \vec{p(1)} + \vec{\texttt{seg}}_A, \\
\vec{dog} + \vec{p(2)} + \vec{\texttt{seg}}_A, &\quad \vec{barks} + \vec{p(3)} + \vec{\texttt{seg}}_A, \\
\vec{.} + \vec{p(4)} + \vec{\texttt{seg}}_A, &\quad \vec{\texttt{[SEP]}} + \vec{p(5)} + \vec{\texttt{seg}}_A, \\
\vec{It} + \vec{p(6)} + \vec{\texttt{seg}}_B, &\quad \vec{is} + \vec{p(7)} + \vec{\texttt{seg}}_B, \\
\vec{a} + \vec{p(8)} + \vec{\texttt{seg}}_B, &\quad \vec{pooch} + \vec{p(9)} + \vec{\texttt{seg}}_B, \\
\vec{.} + \vec{p(10)} + \vec{\texttt{seg}}_B, &\quad \vec{\texttt{[SEP]}} + \vec{p(11)} + \vec{\texttt{seg}}_B
\end{aligned}
$$

BERT, speaking in tongues

Many languages at once

BERT can be trained on multiple language at once.

- May help if there's no BERT for the specific language
- May be less useful than language-specific BERTs (ZH, FR, FI, DE, EN...)

"Multilingual BERT" model was trained on the 100 largest wikipedia dumps

Many languages at once

BPE

Input in BERT is tokenized using Byte-Pair Encoding (BPE) word-pieces.

To compute BPE :

1. start with all characters as possible tokens T
2. tokenize the text according to T and whitespaces
3. find the most frequent pair of tokens $\langle t_1, t_2 \rangle$
4. add their concatenation $t_1 t_2$ to T
5. restart from 2. ; loop until T reaches a certain size

Many languages at once

BPE in Multilingual BERT

In BERT multilingual, all data from all languages are merged into a single corpus, and a vocabulary of 110K BPE word-pieces is then computed.

- Chinese characters in multilingual BERT are considered as distinct tokens, since Chinese doesn't use whitespaces.
- all other language characters are lowercased and accent diacritics are removed
- to avoid over- or under-representing languages when training the model and computing BPE, a sampling ratio per language is defined :

$$\hat{P}(L) = \frac{P(L)^S}{\sum_{L'} P(L')^S}$$

  with $P(L)$ the original magnitude of the language L in the corpus, and S a smoothing factor (in the case of multilingual BERT, $S = 0.7$). The result is that more frequent languages are sampled less, whereas less frequent languages are sampled more.

References I

Devlin, Jacob et al. (2018). « BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding ». In : *CoRR* abs/1810.04805. arXiv : 1810.04805. url : http://arxiv.org/abs/1810.04805.

Firth, J. R. (1957). « A synopsis of linguistic theory 1930-55. ». In : *Studies in Linguistic Analysis (special volume of the Philological Society)* 1952-59, p. 1-32.

McCann, Bryan et al. (2017). « Learned in Translation : Contextualized Word Vectors ». In : *CoRR* abs/1708.00107. arXiv : 1708.00107. url : http://arxiv.org/abs/1708.00107.

Peters, Matthew E. et al. (2018). « Deep contextualized word representations ». In : *CoRR* abs/1802.05365. arXiv : 1802.05365. url : http://arxiv.org/abs/1802.05365.

Radford, Alec (2018). « Improving Language Understanding by Generative Pre-Training ». In :

Taylor, Wilson (1953). « Cloze Procedure : A New Tool for Measuring Readability ». In : *Journalism Quarterly* 30.

Vaswani, Ashish et al. (2017). « Attention Is All You Need ». In : *CoRR* abs/1706.03762. arXiv : 1706.03762. url : http://arxiv.org/abs/1706.03762.