Embeddings in Natural Language Processing

How to Represent Text for Neural Networks

Part A: Embeddings Overview

Ester Hlav

ECBM E6040 Neural Networks and Deep Learning
Columbia University

August 2019

Natural Language Processing with Deep Learning

Intro: **Pre-Deep** Learning NLP (No Word Embeddings)

I. **Static Embeddings** (*MLP*) → embed words
   A. Neural Probabilistic Language Model (**NPLM**)
   B. **Word2Vec** (Word to Vector)
   C. **GloVe** (Global Vectors for Word Representation)
   D. **Character** Embeddings

II. **Contextualized Embeddings** (*Attention with LSTM or Transformers*) → embed sentences
   A. **CoVe** (Contextualized Word Vectors)
   B. **ELMo** (Embeddings from Language Models)
   C. **BERT** (Bidirectional Encoder Representations from Transformers)
   D. **XLM** (Cross-lingual Language Model Pretraining)
   E. **XLNet** (Generalized Autoregressive Pretraining for Language Understanding)
   F. **RoBERTa** (Robustly Optimized BERT Pretraining Approach)

Embeddings Research Papers References

NPLM:	**A Neural Probabilistic Language Model** by Bengio *et al.* (2003)
Word2Vec:	**Distributed Representations of Words and Phrases and their Compositionality** by Mikolov *et al.* (2013)
GloVe:	**Global Vectors for Word Representation** by Pennington *et al.* (2014)
Characters:	**Character-Aware Neural Language Models** by Kim *et al.* (2015)
CoVe:	**Learned in Translation: Contextualized Word Vectors**, McCann *et al.* (2017)
ELMo:	**Deep contextualized word representations**, Peters *et al.* (2018)
BERT:	**Pre-training of Deep Bidirectional Transformers for Language Understanding**, Devlin *et al.* (2018)
XLM:	**Cross-lingual Language Model Pre-training**, Devlin *et al.* (2019)
XLNet:	**Generalized Autoregressive Pre-training for Language Understanding**, Yang *et al.* (2019)
RoBERTa:	**A Robustly Optimized BERT Pre-training Approach**, Liu *et al.* (2019)

Intro: Pre-Deep Learning NLP (No Word Embeddings)

Natural Language Processing (NLP) tasks:

- *Easy*: Spell Checking, Keyword Search, Finding Synonyms, *etc*.
- *Medium*: Parsing Information from Websites or Documents, *etc*.
- *Hard*: Machine Translation, Semantic Analysis, Question Answering, *etc*.

How to represent words? → A computer cannot understand strings, but *does* understand **vectors.**

Two traditional vectorization methods:

i. **Word Vectors**: one-hot encoding

ii. **Singular Value Decomposition** (SVD) methods:
   - Word-Document Matrix
   - Window based Co-occurrence Matrix

Intro: Pre-Deep Learning NLP (No Word Embeddings)

i. **One-hot encoding:**

- V words in vocabulary → V-dimensional vector
- All words assumed independent: $\forall i \neq j, (w^i)^T w^j = 0$
- Can reduce the size of the space by reducing redundancy and sparsity (i.e. embeddings, making V entities represented within a space of size $<$ V)

$$w^{aardvark} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, w^{a} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, w^{at} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \cdots w^{zebra} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

ii. **SVD methods:**

- **Word Document Matrix**
  - Matrix V by M documents
  - Track words in document by 1
- **Window based Co-occurrence Matrix**
  - Matrix V by V
  - Counts co-occurrences in corpus

Applying SVD to X:

$$\begin{array}{cc} & |V| \\ |V| & \begin{bmatrix} & & \\ & X & \\ & & \end{bmatrix} \end{array} = \begin{array}{cc} & |V| \\ |V| & \begin{bmatrix} | & | & \\ u_1 & u_2 & \cdots \\ | & | & \end{bmatrix} \end{array} \begin{array}{cc} & |V| \\ |V| & \begin{bmatrix} \sigma_1 & 0 & \cdots \\ 0 & \sigma_2 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \end{array} \begin{array}{cc} & |V| \\ |V| & \begin{bmatrix} - & v_1 & - \\ - & v_2 & - \\ & \vdots & \end{bmatrix} \end{array}$$

Reducing dimensionality by selecting first k singular vectors:

$$\begin{array}{cc} & |V| \\ |V| & \begin{bmatrix} & & \\ & \hat{X} & \\ & & \end{bmatrix} \end{array} = \begin{array}{cc} & k \\ |V| & \begin{bmatrix} | & | & \\ u_1 & u_2 & \cdots \\ | & | & \end{bmatrix} \end{array} \begin{array}{cc} & k \\ k & \begin{bmatrix} \sigma_1 & 0 & \cdots \\ 0 & \sigma_2 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \end{array} \begin{array}{cc} & |V| \\ k & \begin{bmatrix} - & v_1 & - \\ - & v_2 & - \\ & \vdots & \end{bmatrix} \end{array}$$

Intro: Pre-Deep Learning NLP (No Word Embeddings)

SVD methods have many problems:

- Quadratic cost to train due to SVD cost
- Matrix that SVD is performed on is high-dimensional, typically $10^6 \times 10^6$
- Requires the incorporation of some hacks on X to account for the drastic imbalance in word frequency

Some tricks can help to improve SVD methods:

- Reduce vocabulary with *stemming* (do=doing=does) and *stop-words* (delete repetitive words)
- Apply a *ramp window* – i.e. weight the co-occurrence count based on distance between the words in the document
- Use *Pearson correlation* and set counts with negative correlation to 0 instead of using raw counts

<u>But</u>: remains problematic and computationally expensive → **This is Motivation for DL Embeddings!**

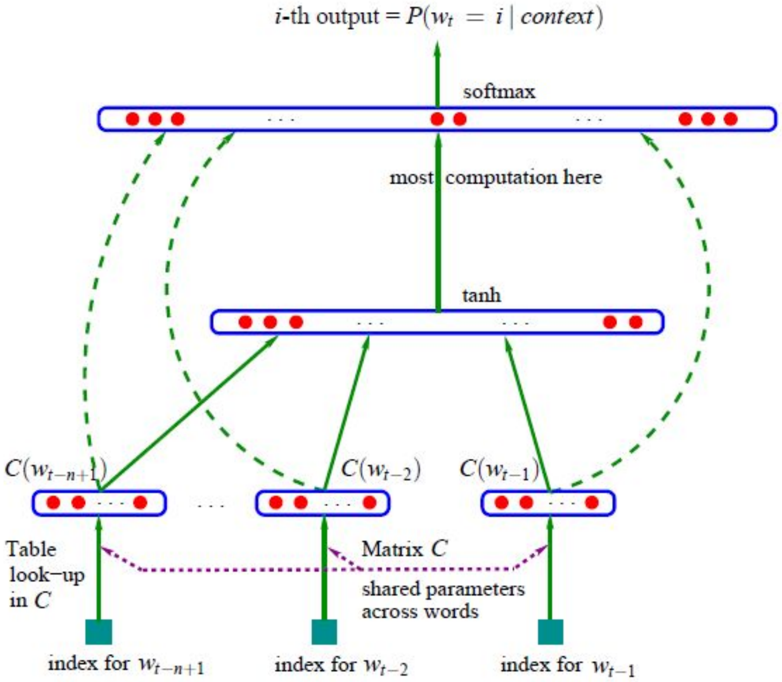
I. Static Embeddings (MLP)

A. **Neural Probabilistic Language Model** *(Bengio et al. - 2003)*

- Model probability distribution of next word given a sequence of N words:

$$J_\theta = \frac{1}{T}\sum_{t=1}^{T} \log P(w_t|w_{t-1},\cdots,w_{t-n+1})$$

- Perform table lookup on the embedding matrix C, i.e. *one-hot encoding* **x** *embedding matrix* $=$ *embedding*
- Bengio et al. suggest to replace MLP by LSTM to leverage the sequential nature of inputs
- Cost of computing softmax is proportional to the vocabulary size V



I. Static Embeddings (MLP)

B. **Word2Vec** *(Mikolov et al. - 2013)*

- Keep NPLM, but with *Continuous Bag-of-words* or *Skip-gram* model
- In both cases, avoid softmax by *ranking* instead of *predicting*
- **Continuous Bag-of-words** (CBOW): from context predict a word

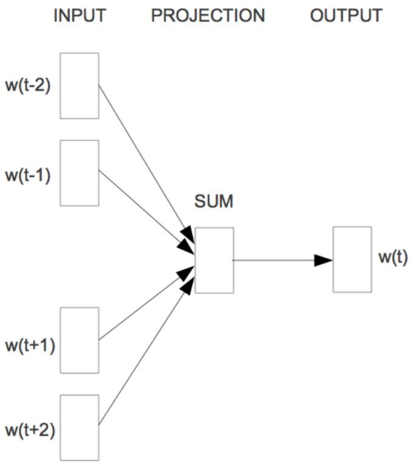
$$J_\theta = \frac{1}{T}\sum_{t=1}^{T} \log P(w_t \mid w_{t-n}, \cdots, w_{t-1}, w_{t+1}, \cdots, w_{t+n})$$

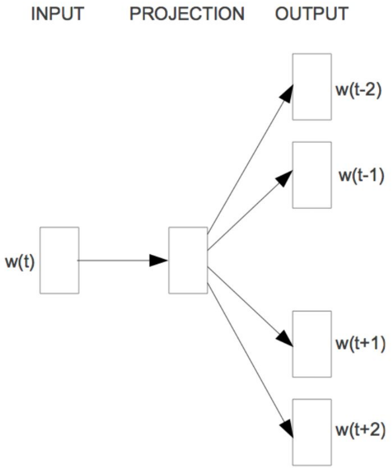
- **Skip-gram**: predict context from a word

$$J_\theta = \frac{1}{T}\sum_{t=1}^{T} \sum_{-n \leq j \leq n, \neq 0} \log P(w_{t+j} \mid w_t)$$

- The main advantages of CBOW and Skip-gram:

  i) no costly hidden layer (architecture)   ii) additional context (task)





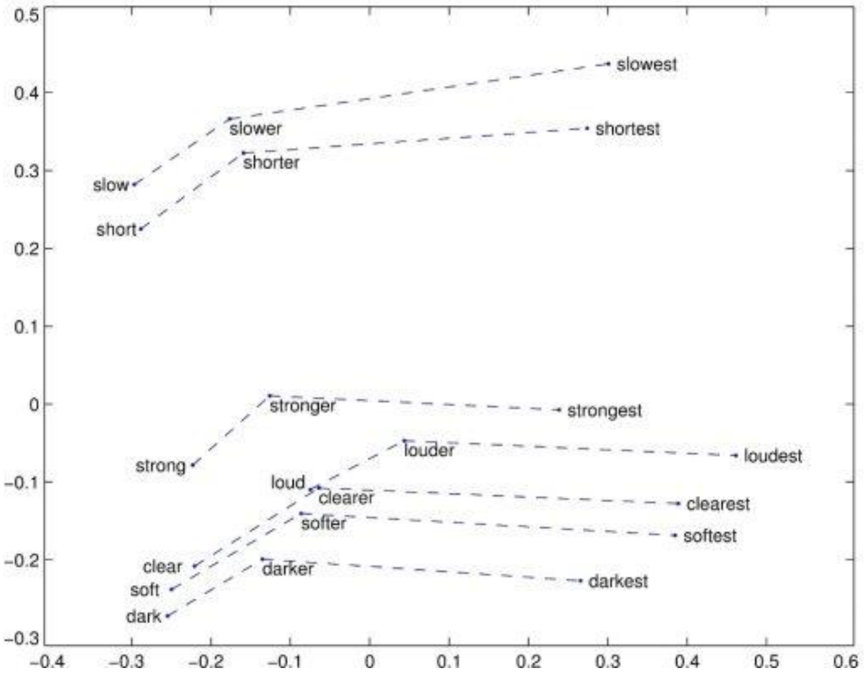
I. Static Embeddings (MLP)

C. **GloVe** *(Pennington et al. - 2014)*

- *Assumption*: co-occurrence probabilities P_{ij} are the main driver of language information: $P_{ij} = \frac{X_{ij}}{\sum_{k=1}^{V} X_{ik}}$
- *Goal*: train embeddings by predicting P_{ij}

$$J = \sum_{i,j=1}^{V} f(X_{ij})\,(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log P_{ij})^2$$

- w_i represents the embedding of word i
- f is the weighting function that assigns relatively low weight to rare and frequent co-occurrences
- Dot product is unnormalized cosine similarity



- Learns representative “word algebra” like the famous: king – man + woman = queen

I. Static Embeddings (MLP)

D. **Character Embeddings** *(Kim et al. - 2015)*

- Embed characters instead of words
- Formed as **ngrams**, e.g. unigram, bigram, etc.
- 2 fundamental advantages:
  - Vocabulary size drastically reduced
  - Can deal with out-of-vocabulary words (OOV), while word embeddings cannot
- Can be concatenated with word embeddings
- Easy to train, thus in general task-specific

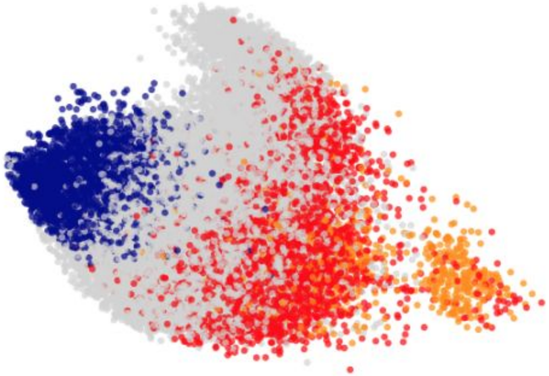


Figure 2: Plot of character n-gram representations via PCA for English. Colors correspond to: prefixes (red), suffixes (blue), hyphenated (orange), and all others (grey). Prefixes refer to character n-grams which start with the start-of-word character. Suffixes likewise refer to character n-grams which end with the end-of-word character.

II. Contextualized Embeddings - Attention *with* LSTM

A. **CoVe** *(McCann et al. - 2017)*

- *Until now*: static embeddings, i.e. the contextualized information has to be learnt by the network
- *Now*: **dynamic embeddings**, i.e. **encoder/decoder** architecture trained using bidirectional LSTM with attention, where encoder is used to encode the word vectors
- Embeddings trained on **machine translation task**; can leverage transfer learning
- Word Vector inputs are concatenation of GloVe and Character embeddings
- CoVe Embedding is the hidden-state of biLSTM's *top* layer

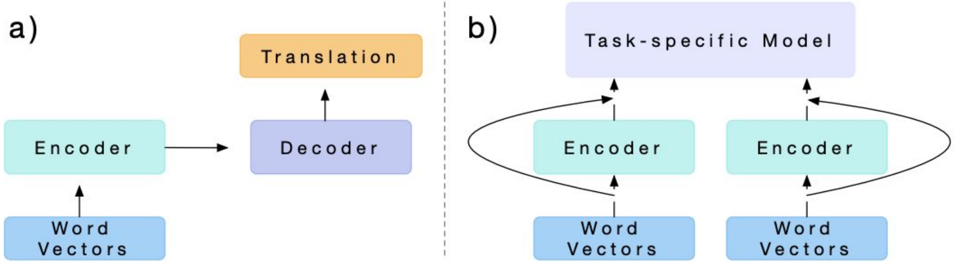


Figure 1: We a) train a two-layer, bidirectional LSTM as the encoder of an attentional sequence-to-sequence model for machine translation and b) use it to provide context for other NLP models.

- CoVe outperforms GloVe on downstream NLP tasks, *e.g.* sentiment analysis, name entity recognition

II. Contextualized Embeddings - Attention *with* LSTM

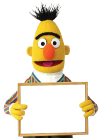
B. **ELMo** *(Peters et al. - 2018)*

- Train **deep embeddings** by training a biLSTM on a **language model task**, using *all* LSTM encoder layers
- *Advantage* over CoVe (supervised): ELMo (unsupervised) is trained on large amount of data
- **Character CNN** used to build initial word representations instead of mix of pretrained embeddings
- Obtains state-of-the-art (SOTA) results on 6 tasks, where performance gain comes only from ELMo:
  - Question Answering
  - Textual entailment
  - Semantic role labeling
  - Coreference resolution
  - Named entity extraction
  - Sentiment analysis

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%

II. Contextualized Embeddings - Attention *with* Transformers

C. **BERT** *(Devlin et al. - 2018)*



- Replaces seq2seq LSTM attention model by a **Transformer** (Attention is all you need, *Vaswani et al. - 2017*)
- Bidirectionality works differently: no concatenation of *independent* unidirectional representations as in ELMo
- **Masked language model task** and advanced location embeddings (used by transformers)

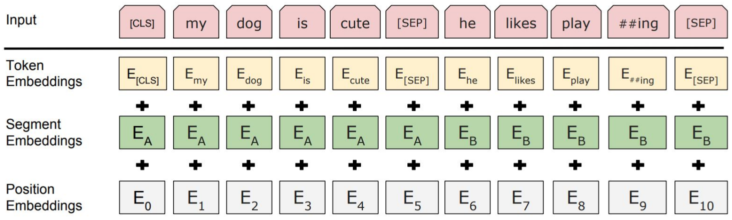
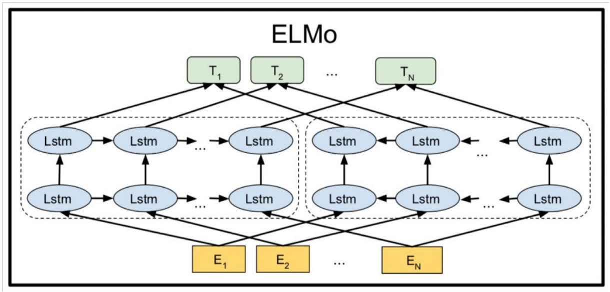
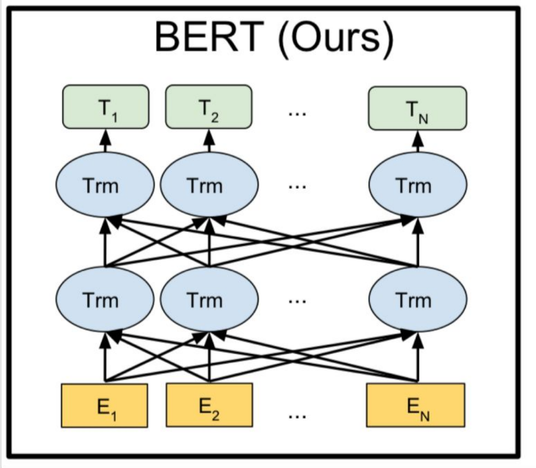


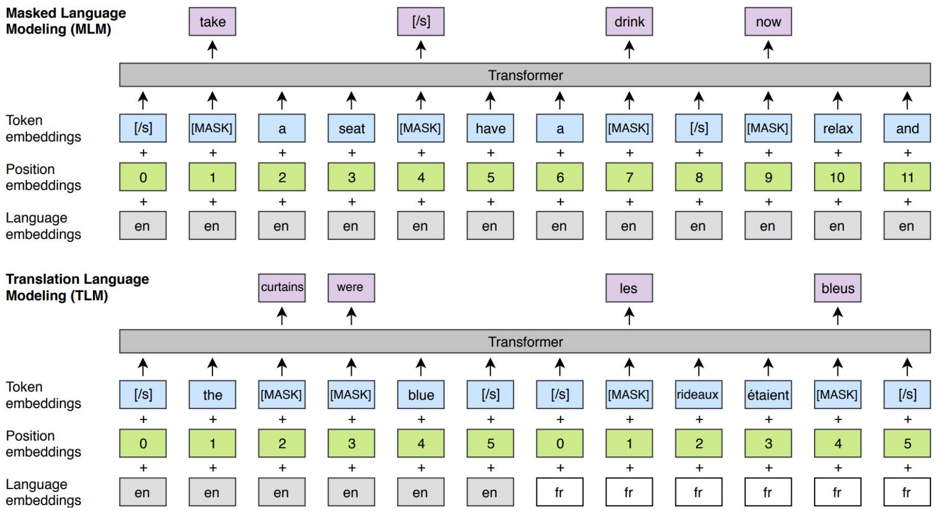
Figure 2: BERT input representation. The input embeddings is the sum of the token embeddings, the segmentation embeddings and the position embeddings.





II. Contextualized Embeddings - Attention *with* Transformers

D. **XLM** *(Lample et al. - 2019)*

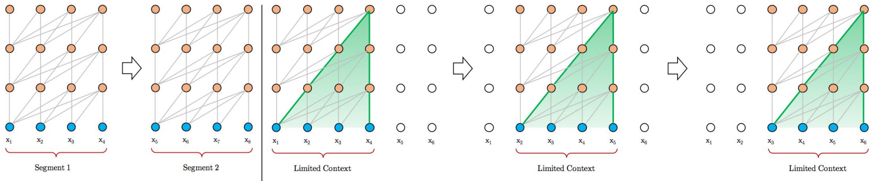


- Cross-lingual pre-training on an *unsupervised* **masked language model task** and a *supervised* **translation language model task**
- Processes *all* languages with the same shared vocabulary (Byte pair encoding) using arbitrary number of sentences (while BERT uses pairs)
- Improved SOTA on:
  - XLNI by 4.9% measured in terms of accuracy
  - unsupervised WMT'16 German-English by 9 BLEU
  - supervised WMT'16 Romanian-English by more than 4 BLEU

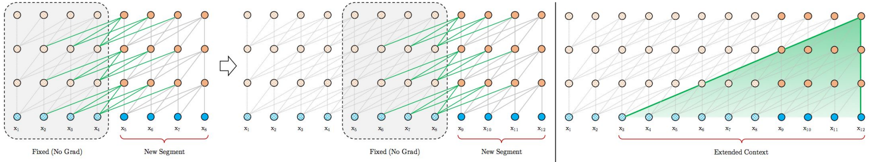
II. Contextualized Embeddings - Attention *with* Transformers

E. **XLNet** *(Yang et al. - 2019)*

Transformers train/eval for long sequences (Vanilla, XL)



(a) Train phase. (b) Evaluation phase.



(a) Training phase. (b) Evaluation phase.

- *BERT* relies on corrupting the input with masks, neglects dependency between the masked positions and suffers from a *pretrain-finetune discrepancy*
- XLNet is a **generalized autoregressive pretraining method** with objective $\max_{\theta} \quad \log p_{\theta}(\mathbf{x}) = \sum_{t=1}^{T} \log p_{\theta}(x_t \mid \mathbf{x}_{<t})$ instead of $\max_{\theta} \quad \log p_{\theta}(\bar{\mathbf{x}} \mid \hat{\mathbf{x}}) \approx \sum_{t=1}^{T} m_t \log p_{\theta}(x_t \mid \hat{\mathbf{x}})$ in BERT
- Uses **Transformer-XL architecture** (*Dai et al. 2010*) with an autoregressive structure $H_t = f(X_t, H_{t-1})$
- Achieves new SOTA on 7 out of 9 tasks benchmarked by GLUE, becoming a leader on *June 2019*
- Empirically, XLNet outperforms BERT on 20 tasks and achieves SOTA results on 18 tasks

II. Contextualized Embeddings - Attention *with* Transformers

F. **RoBERTa** *(Liu et al. - 2019)*

- Studies the impact of key hyperparameters and of training data size in BERT's training
- Modifications: (*1*) longer model training with bigger batches and more data
  (*2*) removing the next sentence prediction objective (NSP loss);
  (*3*) dynamic changing of the masking pattern applied to the training data
  (*4*) training on longer sequences
- GLUE (*July 2019*), RACE and SQuAD leader
- Results illustrate the importance of previously overlooked design decisions and suggest that **BERT pre-training remains competitive** with recently proposed alternatives

Results reported on GLUE

	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
Single-task single models on dev										
$\text{BERT}_{\text{LARGE}}$	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
$\text{XLNet}_{\text{LARGE}}$	89.8/-	93.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa	**90.2/90.2**	**94.7**	**92.2**	**86.6**	**96.4**	**90.9**	**68.0**	**92.4**	**91.3**	-
Ensembles on test (from leaderboard as of July 25, 2019)										
ALICE	88.2/87.9	95.7	**90.7**	83.5	95.2	92.6	**68.6**	91.1	80.8	86.3
MT-DNN	87.9/87.4	96.0	89.9	86.3	96.5	92.7	68.4	91.1	89.0	87.6
XLNet	90.2/89.8	98.6	90.3	86.3	**96.8**	**93.0**	67.8	91.6	**90.4**	88.4
RoBERTa	**90.8/90.2**	**98.9**	90.2	**88.2**	96.7	92.3	67.8	**92.2**	89.0	**88.5**

References

Bengio, Y., Ducharme, R. & Vincent, P. *A neural probabilistic language model*. In Proc. Advances in Neural Information Processing Systems 13 932–938, 2001.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. & Dean, J. *Distributed representations of words and phrases and their compositionality*. In Proc. Advances in Neural Information Processing Systems 26 3111–3119, 2013.

R. Jeffrey Pennington and C. Manning. *Glove: Global vectors for word representation*. 2014.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. *Character-aware neural language models*, 2015.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. *Learned in translation: Contextualized word vectors*, 2017.

ME. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, L. Zettlemoyer. *Deep contextualized word representations, 2018*.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin. *Attention is all you need*, 2017.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. *Bert: Pre-training of deep bidirectional transformers for language understanding*. arXiv preprint arXiv:1810.04805, 2018.

J. Devlin, M.-W. Chang, K. Lee, K. Toutanova. *Cross-lingual Language Model Pretraining*, 2019.

Zhilin Yang , Zihang Dai, Yiming Yang , Jaime Carbonell , Ruslan Salakhutdinov , Quoc V. Le. *XLNet: Generalized Autoregressive Pre-training for Language Understanding*, 2019.

Zhilin Yang , Zihang Dai, Yiming Yang , Jaime Carbonell , Ruslan Salakhutdinov , Quoc V. Le. *Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context*, 2019.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, Veselin Stoyanov. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*, 2019.