Transformer-based models and Applications for Information Retrieval
Attention mechanism

✓ Attention mechanism

• Encodes long-distance dependency
• Captures contextual relationship
• Widely used in a variety of neural networks
✓ Attention mechanism

...is just a fancy word for **weighted average**

\[
\alpha_i \quad 0.04 \quad 0.14 \quad 0.06 \quad 0.73 \quad 0.01 \quad 0.02
\]

They eat two apples this morning  \hspace{2cm} <query> red fruit

\[
\omega_i \quad (300\text{-dim embedding for each word})
\]

• Conceptually, \hspace{1cm} Attention = \alpha_i \omega_i
**Scaled dot-product attention**

\[
\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V
\]

Feed query/document through individual FFNs to get \(Q, K, V\):

- \(W_{\text{query}}, W_{\text{doc}}\): a sequence of word embeddings of query/document
- \(d_k\): dimension of word embeddings
- \(Q = \text{FFN}_Q(W_{\text{query}})\); \(K = \text{FFN}_K(W_{\text{doc}})\); \(V = \text{FFN}_V(W_{\text{doc}})\)
✓ RNN with attention

- Usually take the last hidden output as query
- Performance degrades if the distance is very long
✓ **Bidirectional RNN with self-attention**

- Attention of **all possible pairs** of any hidden outputs
- Alleviate but still suffer for long-distance problem
Transformer

✓ Self-attention

\[
\begin{align*}
W_Q & \times \mathbf{d}_k = Q \\
W_K & \times \mathbf{d}_k = K \\
W_V & \times \mathbf{d}_k = V \\
\text{softmax}(\frac{Q K^\top}{\sqrt{d_k}}) & = \text{Attention}
\end{align*}
\]
✓ Multi-head attention

• Combine multiple attention layers in parallel
• Increase feature resolution
• Similar effect of ensemble
Transformer

✓ Transformer

• The first model entirely relies on self-attention
• No recurrent nor convolution operations

<table>
<thead>
<tr>
<th>Layer Type</th>
<th>Complexity per Layer</th>
<th>Sequential Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self-Attention</td>
<td>$O(\text{length}^2 \cdot \text{dim})$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>Recurrent</td>
<td>$O(\text{length} \cdot \text{dim}^2)$</td>
<td>$O(\text{length})$</td>
</tr>
<tr>
<td>Convolutional</td>
<td>$O(\text{kernel} \cdot \text{length} \cdot \text{dim}^2)$</td>
<td>$O(1)$</td>
</tr>
</tbody>
</table>

length: input length; dim: hidden size; kernel: kernel size
✓ **Transformer encoder**

- A multi-head self-attention layer
- Residual connections
- Layer normalization

\[
x = \text{Emb(inputs)} + \text{Emb(position)} \\
z = \text{LayerNorm}(x + \text{MHSA}(x)) \\
y = \text{LayerNorm}(z + \text{FFN}(z))
\]
✓ BERT -- Jacob Devlin et al., October 2018.

• Bidirectional Encoder Representations from Transformers
• Pretrain representations from unlabeled text
• Can be easily finetuned for a wide range of tasks
• Obtained new state-of-the-art results on 11 NLP tasks
✓ BERT  -- Jacob Devlin et al., October 2018.

- Consists of stacked transformer layers and 3 embedding layers
- Officially provide multiple size of pretrained BERT
  - BERT-base: 12 layers, 12 heads, 768-hidden (110M params)
  - BERT-large: 24 layers, 16 heads, 1024 hidden (340M params)
✓ BERT -- Jacob Devlin et al., October 2018.

- Use **WordPiece** tokens (a subword tokenization method)
- Sum up token/segment/position embeddings as input
✓ Masked Language Model (MLM)

- A pretraining objective inspired by the Cloze task
✓ Masked Language Model (MLM)

- A pretraining objective inspired by the Cloze task
- Randomly mask 15% tokens in each sequence
  - 80% are replaced with [MASK] token
  - 10% are replaced with other random tokens
  - 10% are keep unchanged
  - Masking and replacement is performed once in the beginning
✓ Next Sentence Prediction (NSP)

- Pretraining for understanding of sentence-level relationship

$\text{label} = 1$ if Sentence B is the actual next sentence of Sentence A, else 0

Sentence A: [CLS] she is vegetarian [SEP]
Sentence B: she hates meat [SEP]
Next Sentence Prediction (NSP)

- Pretraining for understanding of sentence-level relationship
- Extract contiguous sequences from document-level corpus
  - 50% of Sentence B is the actual next sentence of sentence A
  - 50% of Sentence B is a random sentence from the corpus
✓ Pretraining of BERT

- Jointly pretrain MLM and NSP objectives
- Officially pretrain on BooksCorpus (800M words) & Wikipedia (2.5B words)
- Classifiers in MLM and NSP are only used for pretraining
  (i.e. you should stack a new classifier on BERT for downstream tasks)
✓ Finetune BERT on downstream tasks

• Highly compatible due to its architecture and pretraining

label = 1 if Document is relevant to Query, else 0

Binary Classifier for Document Retrieval

From NSP to IR :-)!

Query

Document
✓ Autoregressive (AR)

• Autoregressive language modeling: (unidirectional)
  Predict $x_3$ with \{ $x_1, x_2$ \}; predict $x_4$ with \{ $x_1, x_2, x_3$ \}

✓ Autoencoding (AE)

• Denoising autoencoding (DAE): data reconstruction (bidirectional)
  Predict $x_3$ with \{ $x_1, x_2, x_4$ \}; predict $x_4$ with \{ $x_1, x_2, x_3$ \}

Drink beer at bar

\[
\begin{align*}
  x_1 & \quad \text{Drink} \\
  x_2 & \quad \text{beer} \\
  x_3 & \quad \text{at} \\
  x_4 & \quad \text{bar}
\end{align*}
\]
✓ ELMo -- Matthew E. Peters et al., March 2018.

• AR language modeling with bidirectional LSTM
• Base layer only encodes unidirectional information

Backward LSTM

Forward LSTM

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>He</td>
<td>drinks</td>
<td>beer</td>
<td>at</td>
<td>bar</td>
<td></td>
</tr>
<tr>
<td>x_1</td>
<td>x_2</td>
<td>x_3</td>
<td>x_4</td>
<td>x_5</td>
<td></td>
</tr>
</tbody>
</table>

He drinks beer at bar
He drinks beer at bar
He drinks beer at bar
✓ MLM is a DAE-based pretraining

• Has capability of modeling bidirectional contexts
• BERT outperforms AR-based models like ELMo, GPT
• So... what’s wrong with MLM?
MLM is a DAE-based pretraining

- Predict \( \{x_3, x_5\} \) with \( \{x_1, x_2, x_4\} \):
  - unable to model the dependency between \( x_2 \) and \( x_4 \)

- [MASK] token is never used in downstream tasks → input noise
XLNet

✓ XLNet -- Zhilin Yang et al., June 2019.

• A novel AR pretraining method which learns bidirectional context
• Overcomes the limitations of MLM thanks to AR formulation
• Not using [MASK] token for pretraining
• Outperforms BERT on 20 NLP tasks

In brief, XLNet has both the advantages of AR and AE
✓ Permutation Language Modeling (PLM)

- AR pretraining with all possible permutations of a sequence

1. Permutate \( \{1, 2, 3, 4\} \) → predict \( x_3 \) with \( \{x_1, x_2\} \) (regular AR)
2. Permutate \( \{2, 4, 3, 1\} \) → predict \( x_3 \) with \( \{x_2, x_4\} \)
3. Permutate \( \{1, 4, 2, 3\} \) → predict \( x_3 \) with \( \{x_1, x_4, x_2\} \)
4. ...etc
XLNet

✓ Permutation Language Modeling (PLM)

• Autoregressive self-attention mask

\[
\begin{array}{cccc}
\text{drink} \quad & \text{bar} \quad & \text{beer} \quad & \text{at} \\
\text{x}_1 & \text{x}_4 & \text{x}_2 & \text{x}_3 \\
\text{x}_1 & \bullet & & \\
\text{x}_4 & \bullet & \bullet & \\
\text{x}_2 & \bullet & \bullet & \bullet \\
\text{x}_3 & \bullet & \bullet & \bullet & \bullet \\
\end{array}
\]

\[
\begin{array}{cccc}
\text{drink} \quad & \text{beer} \quad & \text{at} \quad & \text{bar} \\
\text{x}_1 & \text{x}_2 & \text{x}_3 & \text{x}_4 \\
\text{x}_1 & \bullet & & \\
\text{x}_2 & \bullet & \bullet & \bullet \\
\text{x}_3 & \bullet & \bullet & \bullet & \bullet \\
\text{x}_4 & \bullet & \bullet & \bullet & \bullet \\
\end{array}
\]
Permutation Language Modeling (PLM)

- AR now learns bidirectional contexts with input permutation
- The token classifier shall not see the target token!
- So... how to mask the target token without using [MASK] token?

(MLM)  

drink  beer  [MASK]  bar

(PLM)  

drink  beer  ???  bar

\[
\begin{align*}
x_1 & \quad \text{drink} \\
x_2 & \quad \text{beer} \\
x_3 & \quad [\text{MASK}] \\
x_4 & \quad \text{bar}
\end{align*}
\]
Two-Stream Self-Attention

Assume we have an AR self-attention mask, and we want to predict $x_1$ with $\{x_2, x_3, x_4\}$.
Two-Stream Self-Attention

Content Stream:
can see self

Query Stream:  (for PLM only)
cannot see self
$e(x_i) : \text{embedding of } x_i$

$w : \text{positional encoding}$
✓ Pretraining of XLNet

• Pretrain PLM objective only (NSP is found unhelpful for XLNet)

• XLNet uses SentencePiece tokenization

• Officially pretrain on BooksCorpus (800M words), Wikipedia (2.5B words), Giga5 (16GB text), ClueWeb 2012-B, and Common Crawl

• Query Stream is only used in PLM;
  Content Stream is used in PLM and downstream tasks

10x MORE DATA ;-)!
RoBERTa

✓ RoBERTa  -- Yinhan Liu et al., July 2019.

• Robustly optimized BERT approach
• Original BERT (by Google) is significantly undertrained
• Propose an improved recipe for training BERT models
• Match or outperform all post-BERT methods (e.g. XLNet)
<table>
<thead>
<tr>
<th></th>
<th>BERT</th>
<th>RoBERTa</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch size</td>
<td>256</td>
<td>8K</td>
</tr>
<tr>
<td>Training steps</td>
<td>1M</td>
<td>500K</td>
</tr>
<tr>
<td>Corpus size</td>
<td>13GB</td>
<td>160GB</td>
</tr>
<tr>
<td>MLM masking</td>
<td>Static (fixed once prepared)</td>
<td>Dynamic (generate on the fly)</td>
</tr>
<tr>
<td>Objective</td>
<td>MLM + <strong>NSP</strong></td>
<td>MLM</td>
</tr>
<tr>
<td>Tokenization</td>
<td>WordPiece (vocab. size = 30k)</td>
<td>Byte-level <strong>BPE</strong> (vocab. size = 50k)</td>
</tr>
</tbody>
</table>
RoBERTa

✓ RoBERTa

- Significantly outperforms BERT on GLUE benchmark
- Outperforms XLNet on every single task in GLUE

*General Language Understanding Evaluation (GLUE) benchmark*

<table>
<thead>
<tr>
<th></th>
<th>MNLI</th>
<th>QNLI</th>
<th>QQP</th>
<th>RTE</th>
<th>SST</th>
<th>MRPC</th>
<th>CoLA</th>
<th>STS</th>
<th>WNLI</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT_LARGE</td>
<td>86.6/-</td>
<td>92.3</td>
<td>91.3</td>
<td>70.4</td>
<td>93.2</td>
<td>88.0</td>
<td>60.6</td>
<td>90.0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>XLNet_LARGE</td>
<td>89.8/-</td>
<td>93.9</td>
<td>91.8</td>
<td>83.8</td>
<td>95.6</td>
<td>89.2</td>
<td>63.6</td>
<td>91.8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RoBERTa</td>
<td>90.2/90.2</td>
<td>94.7</td>
<td>92.2</td>
<td>86.6</td>
<td>96.4</td>
<td>90.9</td>
<td>68.0</td>
<td>92.4</td>
<td>91.3</td>
<td>-</td>
</tr>
</tbody>
</table>
✓ ALBERT -- Yinhan Liu et al., September 2019.

• A Lite BERT

• Difficult to experiment with large models due to memory constraints

• Propose several parameter-reduction techniques

• Propose a new pretraining method to replace NSP
✓ Factorized embedding parameterization

- Word embeddings are meant to be context-independent
- Hidden outputs are meant to be context-dependent
- It is efficient and reasonable to have a smaller word dimension

Both BERT and ALBERT have vocabulary size = 30,000:

- BERT = Embedding(30000, 768) (23M params)
- ALBERT = Embedding(30000, 128) $\rightarrow$ FFN(128, 768) (4M params)
✓ Cross-layer parameter sharing

**BERT**

- Transformer
- Transformer
- Transformer

**ALBERT**

- Transformer

=N layers

N times

* Share all parameters across layers *
Why is Next Sentence Prediction (NSP) unhelpful?

- NSP conflates topic prediction and coherence prediction
- It is much easier to learn topic prediction (high word overlapping)
- Models tend to learn the easier topic-prediction signal

Sentence A | Sentence B
---|---
(Positive) | I play with my dog. | My dog makes me exhausted.
(Negative) | I play with my dog. | She drank a glass of water.
✓ Sentence Order Prediction (SOP)

- Swap the order of positive samples from NSP as negative samples
- Force models to learn coherence prediction (topics are unchanged)

<table>
<thead>
<tr>
<th>Sentence A</th>
<th>Sentence B</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Positive) I play with my dog.</td>
<td>My dog makes me exhausted.</td>
</tr>
<tr>
<td>(Negative) My dog makes me exhausted.</td>
<td>I play with my dog.</td>
</tr>
</tbody>
</table>
✓ Pretraining of ALBERT

- Jointly pretrain MLM and SOP objectives
- ALBERT uses SentencePiece tokenization
- Officially pretrain on BooksCorpus (800M words) & Wikipedia (2.5B words)

* to fairly compare with BERT *
✓ Parameter-efficiency of ALBERT

- ALBERT can have significantly fewer parameters and faster speed without seriously hurting performance

<table>
<thead>
<tr>
<th>Model</th>
<th>Parameters</th>
<th>SQuAD1.1</th>
<th>SQuAD2.0</th>
<th>MNLI</th>
<th>SST-2</th>
<th>RACE</th>
<th>Avg</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT</td>
<td>base</td>
<td>108M</td>
<td>90.4/83.2</td>
<td>80.4/77.6</td>
<td>84.5</td>
<td>92.8</td>
<td>68.2</td>
<td>82.3</td>
</tr>
<tr>
<td></td>
<td>large</td>
<td>334M</td>
<td>92.2/85.5</td>
<td>85.0/82.2</td>
<td>86.6</td>
<td>93.0</td>
<td>73.9</td>
<td>85.2</td>
</tr>
<tr>
<td>ALBERT</td>
<td>base</td>
<td>12M</td>
<td>89.3/82.3</td>
<td>80.0/77.1</td>
<td>81.6</td>
<td>90.3</td>
<td>64.0</td>
<td>80.1</td>
</tr>
<tr>
<td></td>
<td>large</td>
<td>18M</td>
<td>90.6/83.9</td>
<td>82.3/79.4</td>
<td>83.5</td>
<td>91.7</td>
<td>68.5</td>
<td>82.4</td>
</tr>
<tr>
<td></td>
<td>xlarge</td>
<td>60M</td>
<td>92.5/86.1</td>
<td>86.1/83.1</td>
<td>86.4</td>
<td>92.4</td>
<td>74.8</td>
<td>85.5</td>
</tr>
<tr>
<td></td>
<td>xxlarge</td>
<td>235M</td>
<td>94.1/88.3</td>
<td>88.1/85.1</td>
<td>88.0</td>
<td>95.2</td>
<td>82.3</td>
<td>88.7</td>
</tr>
</tbody>
</table>
✓ Vocabulary issues

- Common vocab. size= 50k~200k for TFIDF/RNN-based models
- Millions of unique words in a big corpus like Wikipedia!
- Out-of-vocabulary (OOV) words **ALWAYS** exists! (e.g. internet slang)
- Subword: seek a trade-off between **semantics** and **memory**
✓ **Byte Pair Encoding**  -- Philip Gage, February 1994.

- A data compression algorithm

- Algorithm flow:
  1. Vocabulary initialized with all unique characters
  2. Greedily merge bigram with highest frequency
  3. Stop merging if conditions are met (e.g. vocab. size = 30k)
True Byte Pair Encoding -- Philip Gage, February 1994.

1. \{ "low": 5, "lower": 2, "newest": 6, "widest": 3 \} → Bigram freq. = \{ es: 9, st: 9, we: 8, lo: 7, ow: 7, ne: 6, ... \}

2. \{ "low": 5, "lower": 2, "new est": 6, "wid est": 3 \} → Bigram freq. = \{ est: 9, lo: 7, ow: 7, ne: 6, ..., we: 2, ... \}

3. \{ "low": 5, "lower": 2, "new est": 6, "wid est": 3 \} → Bigram freq. = \{ lo: 7, ow: 7, ne: 6, ..., we: 2, ... \}

......
✓ **WordPiece**  -- Mike Schuster, March 2012.

- A variation of BPE with adaptation to language modeling
- Merge bigram which increases likelihood the most on the corpus

1. \{ "low": 5, "lower": 2, "newest": 6, "widest": 3 \}

→ \( \Delta \text{Likelihood} = \{ \text{es: 0.14, st: 0.14, we: 0.12, lo: 0.11} \ldots \} \)

...
Unigram Language Modeling (Uni. LM) -- Taku Kudo, April 2018.

Algorithm flow:

1. Vocabulary initialized with all possible substrings
2. Prune 20% tokens which decrease likelihood the most
3. Stop pruning if conditions are met

1. lower, ... → \{l, o, w, e, r, lo, ow, we, er, ..., low, owe, ..., lowe, ower, lower...\}
   → Δ Likelihood = \{ ower: $-2.7$, lo: $-1.3$, ..., we: 0.12, low: 0.11 ... \}

...
✓ BPE  with vocab. = \{\text{dis}, \text{car}, \text{de}, \text{ed}, \text{d}\}

\[
\text{discarded} \rightarrow \text{dis} \mid \text{car} \mid \text{de} \mid \text{d}
\]

* Greedy substitution *

✓ Uni. LM

\[
\begin{array}{llll}
\text{discarded} & \rightarrow & \text{discarded} & p = 0.0005 \\
\text{discarded} & \rightarrow & \text{discard} \mid \text{ed} & p = 0.0018 \\
\text{discarded} & \rightarrow & \text{dis} \mid \text{car} \mid \text{d} \mid \text{ed} & p = 0.0012 \\
\text{discarded} & \rightarrow & \text{dis} \mid \text{car} \mid \text{de} \mid \text{d} & p = 0.0008 \\
\text{discarded} & \rightarrow & \text{d} \mid \text{i} \mid \text{s} \mid \text{c} \mid \text{a} \mid \text{r} \mid \text{d} \mid \text{e} \mid \text{d} & p \approx 0.00026
\end{array}
\]

* Viterbi decoding for maximum likelihood *
✓ **SentencePiece**  -- Taku Kudo et al., August 2018.

- A open-sourced library made by Google
- Take *whole sentences* to train BPE/Uni. LM

**e.g. SentencePiece with BPE**

1. \{“this_is_lower”: 3, “what_is_the_newest”: 2\}  
   → **Bigram freq.** = \{ is: 8, th: 5, we: 5, ... \}

...
<table>
<thead>
<tr>
<th>Tokenization</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT</td>
<td>WordPiece</td>
</tr>
<tr>
<td>RoBERTa, GPT-2</td>
<td>BPE over raw bytes</td>
</tr>
<tr>
<td>T5</td>
<td>BPE over Unicode characters</td>
</tr>
<tr>
<td>XLNet, ALBERT</td>
<td>SentencePiece</td>
</tr>
</tbody>
</table>

- Both papers don’t clarify whether they use BPE or Uni. LM
Homework 6

✓ Homework 6

• Goal: Rescore BM25 retrieval with Transformer-based models

• Performance are measured with MAP@1000

• It is NOT allowed to use any approach that doesn’t involve a Transformer-based model!

  e.g. X Simply finetune a strong BM25 system

  e.g. O Strong BM25 + weak BERT (allowed but not encouraged)

  e.g. Δ Strong BM25 = 45.08 → + BERT = 43.78 (harmful BERT rescoring)
Homework 6

Given data:

• 100,000 documents
• 120 training queries & 80 testing queries (50% for private)
• Positive (i.e. relevant) document IDs for each training query
• Top-1000 BM25 document IDs and scores for all queries
This file contains all the documents for information retrieval.

<table>
<thead>
<tr>
<th>doc_id</th>
<th>doc_text</th>
</tr>
</thead>
<tbody>
<tr>
<td>FBIS3-100</td>
<td>English &lt;/F&gt; Article Type:BNF [Text] Nairobi, 28 Feb (KNA) -- The chai...</td>
</tr>
<tr>
<td>FBIS3-10095</td>
<td>Spanish &lt;/F&gt; Article Type:BNF [Text] Havana, 25 Feb (DPA) -- Today, Cu...</td>
</tr>
<tr>
<td>FBIS3-10007</td>
<td>Spanish &lt;/F&gt; Article Type:BNF «F P=106» [From the 'Evening Information R...</td>
</tr>
<tr>
<td>FBIS3-10009</td>
<td>Spanish &lt;/F&gt; Article Type:BNF «F P=106» &quot;Stenographic version&quot; of state...</td>
</tr>
</tbody>
</table>
### Data Explorer

- **336.33 MB**
- **documents.csv**
- **sample_submission.csv**
- **test_queries.csv**
- **train_queries.csv**

---

**train_queries.csv** (2.93 MB)

#### About this file

This file contains queries, positive (relevant) and BM25 top-1000 document IDs for training.

<table>
<thead>
<tr>
<th>query_id</th>
<th>query_text</th>
<th>pos_doc_ids</th>
<th>bm25_top1000</th>
<th>bm25_top1000_s...</th>
</tr>
</thead>
<tbody>
<tr>
<td>302</td>
<td>Poliomyelitis and Post-Polio</td>
<td>FBIS3-20548 FBIS3-222599 FBIS3-22568 FBIS3-22590 FBIS3-26593 FBIS3-4174 FBIS3-60483 FBIS...</td>
<td>32.84784386 31.61291452 23.57999093 22.78184683 19.16013566 18.75911496 18.37569719 18.03671536 18.6...</td>
<td></td>
</tr>
</tbody>
</table>
Document IDs/scores are saved as space-delimited strings:

- `query_id` = 302
- `pos_doc_ids` = “FBIS3-20548 FBIS3-22539 FBIS3-22560 FBIS3-22589 FBIS3-26593 FBIS3-41672 FBIS3-41724 FBIS3-60403 ...
- `bm25_top1000` = “FBIS4-67701 LA043090-0036 LA031489-0032 FBIS4-30637 FR940126-2-00106 FBIS3-60405 ...
- `bm25_top1000_scores` = “32.84784386 31.61291462 23.97999093 22.78184683 19.16614366 18.75011496 ...

Naïve BERT method

All documents

Positive Documents

Negative Documents

Query

Binary classification task

Positive Documents

label = 1

Random sampling

Negative Documents

label = 0
✓ Naïve BERT method
Problems of naïve BERT method

• Need to score all documents for a single query

  Testing set: 80 queries * 100,000 docs = 8,000,000 computations

• Lack of difficult samples

  e.g. Positive/negative documents should look similar

• Positive/negative documents are independently trained

  BERT cannot learn by comparing between pos./neg. documents
Baseline approach

- Positive Documents
- BM25 top-1000
- More difficult samples
- Positive Documents (label = 1)
- Negative Documents (label = 0)

Query

random sampling
Baseline approach

At inference stage

Predict relevance scores of BM25 top-1000 documents only

100x faster than naïve method :-)}
## Homework 6 - Baseline

### Baseline approach

<table>
<thead>
<tr>
<th>Query</th>
<th>Document 1</th>
<th>Document 2</th>
<th>Document 3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Query</strong></td>
<td><strong>Negative</strong></td>
<td><strong>Positive</strong></td>
<td><strong>Negative</strong></td>
</tr>
<tr>
<td><strong>Query</strong></td>
<td><strong>Document 1</strong></td>
<td><strong>Document</strong></td>
<td><strong>Document 2</strong></td>
</tr>
<tr>
<td><strong>Query</strong></td>
<td><strong>Negative</strong></td>
<td><strong>Document 2</strong></td>
<td><strong>Document 3</strong></td>
</tr>
<tr>
<td><strong>Query</strong></td>
<td><strong>Negative</strong></td>
<td><strong>Document 3</strong></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BERT</th>
<th>FFN</th>
</tr>
</thead>
<tbody>
<tr>
<td>(scores)</td>
<td>(labels)</td>
</tr>
<tr>
<td>0.89</td>
<td>0</td>
</tr>
<tr>
<td>2.87</td>
<td>1</td>
</tr>
<tr>
<td>0.32</td>
<td>0</td>
</tr>
<tr>
<td>1.52</td>
<td>0</td>
</tr>
</tbody>
</table>

* Scores are independently predicted *

**Multiple-choice classification task**

Cross-entropy
✓ Baseline approach

• For each query,

  rescore BM25 top-1000 documents with:

\[
\text{score}_{\text{new}} = \text{score}_{\text{BM25}} + \alpha \cdot \text{score}_{\text{BERT}}
\]
Baseline settings

It takes 1~1.5 hrs to run everything on a free Kaggle kernel :-(

Hyperparameters for BERT:

- **Pretrained parameters**: “bert-base-uncased”
- **Optimizer**: AdamW w/ learning rate = 3e-5
- **Num. epochs** = 1
- **Num. of negative documents** = 3
- **Batch size** = 2
- **Split 20% of training queries to grid search optimal α for BERT**
## Baseline performance

<table>
<thead>
<tr>
<th>Method</th>
<th>MAP@1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>BM25 + $\alpha \cdot$ BERT</td>
<td>45.084</td>
</tr>
<tr>
<td>BM25</td>
<td>39.136</td>
</tr>
<tr>
<td>Rerank BM25 w/ BERT only</td>
<td>30.248</td>
</tr>
<tr>
<td>Random Documents</td>
<td>0.007</td>
</tr>
</tbody>
</table>
Free Kaggle GPU & TPU v3-8

- Kaggle provides free GPU (16GB VRAM) quota of 30+ hrs/week
- Requires phone verification
Free Kaggle GPU & TPU v3-8

- Kaggle provides free GPU (16GB VRAM) quota of 30+ hrs/week
- Requires phone verification
✓ Preferred tools & tutorials

• PyTorch:
  Official tutorial (~ 60 minutes)
  https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html

• Huggingface’s Transformers
  Quick tour on Github & usage examples in documentation
  https://github.com/huggingface/transformers
  https://huggingface.co/transformers/
 Submission

• Kaggle URL: https://www.kaggle.com/t/4a26f9f4ba1b4feb952d5aafd98eee94
• You can submit 5 times per day.
• You can select 2 submissions to be used for your final score.
• Deadline: 2021/1/4 23:59 (Monday)
Grading 15 points in total

1. Outperform baseline  get 5 points
2. Experiment report  2 point at max
3. Peer competition  8 point at max

Peer score = 8 \times \frac{\text{your MAP} - \text{baseline MAP}}{\text{1st MAP} - \text{baseline MAP}}

* Based on MAP of private leaderboard *
Thank you for your attention!

Any questions or comments?

(Chia-Chih Kuo)

Natural Language Processing Laboratory
National Taiwan University of Science and Technology