Article Presentation
IA368DD_2023S1: Deep Learning aplicado a Sistemas de Buscas
Student: Marcus Vinícius Borela de Castro

Pretrained Transformers for Text Ranking BERT and Beyond
Chapter 1

By: Jimmy Lin, Rodrigo Nogueira, and Andrew Yates
Link

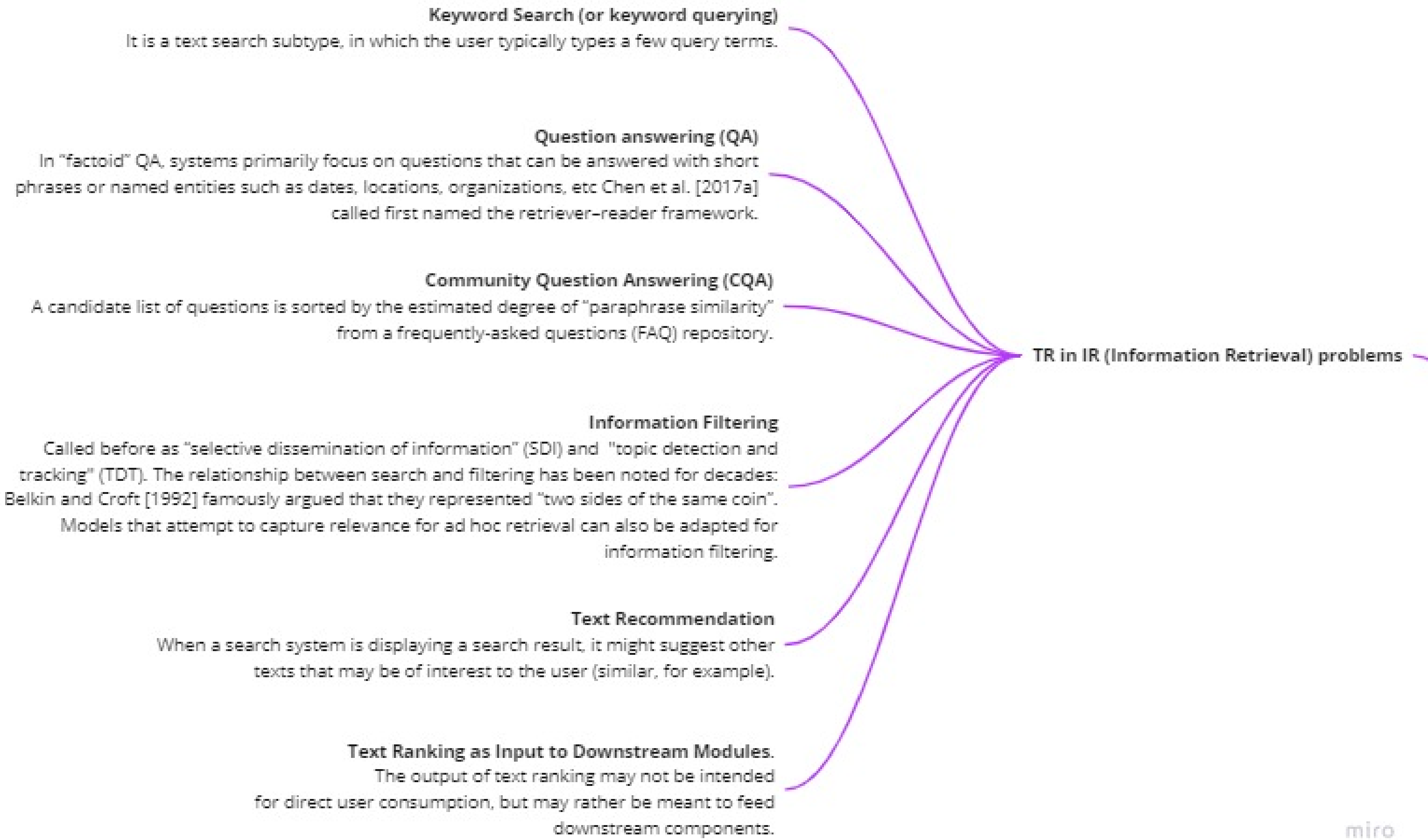
Main concepts

Text ranking (TR)

Goal: is to generate an ordered list of texts retrieved from a corpus in response to a query for a particular task

Text search (ad hoc retrieval): is the most commom problem. The search engine (retrieval system) produces a ranked list (hit lists, hits, “ten blue links”, or search engine results pages SERPs) of texts ordered by estimated relevance (are “about” the topic) with respect to the user’s query.

It is not “**document ranking**”. In many applications, the “atomic unit” of text to be ranked is not a document, but rather a sentence, a paragraph, or even a tweet


Keyword Search (or keyword querying)
It is a text search subtype, in which the user typically types a few query terms.
Question answering (QA)
In "factoid" QA, systems primarily focus on questions that can be answered with short phrases or named entities such as dates, locations, organizations, etc Chen et al. [2017a] called first named the retriever–reader framework.
Community Question Answering (CQA)
A candidate list of questions is sorted by the estimated degree of "paraphrase similarity" from a frequently-asked questions (FAQ) repository.
Information Filtering
Called before as "selective dissemination of information" (SDI) and "topic detection and tracking" (TDT). The relationship between search and filtering has been noted for decades: Belkin and Croft [1992] famously argued that they represented "two sides of the same coin". Models that attempt to capture relevance for ad hoc retrieval can also be adapted for information filtering.
Text Recommendation
When a search system is displaying a search result, it might suggest other texts that may be of interest to the user (similar, for example).
Text Ranking as Input to Downstream Modules.
The output of text ranking may not be intended for direct user consumption, but may rather be meant to feed downstream components.
TR in IR (Information Retrieval) problems
miro

Semantic matching refers to techniques and attempts to address a variety of linguistic phenomena, including synonymy, paraphrase, term variation, and different expressions of similar intents, specifically in the context of information access [Li and Xu, 2014]

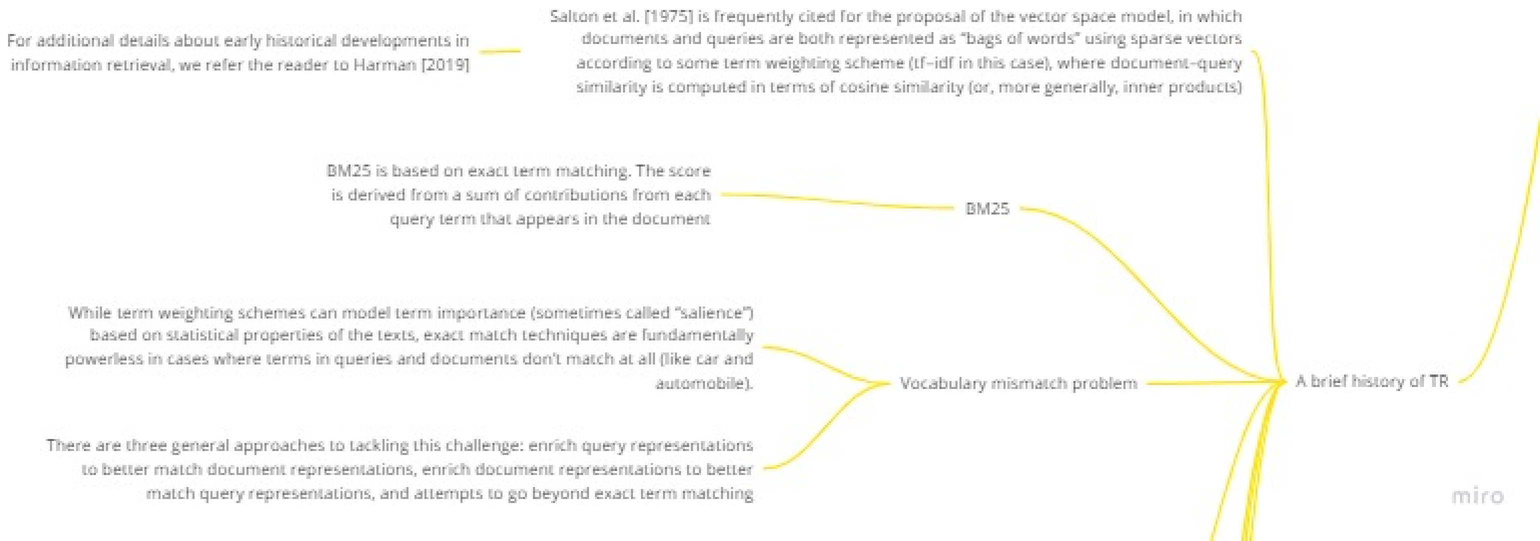
Relevance matching is generally understood to comprise both exact match and semantic match components

Tthere is one major difference: inputs to a model for computing semantic similarity are symmetric, i.e., Rel(s1,s2) = Rel(s2,s1), whereas queries and documents are obviously diferente and cannot be swapped as model inputs.

Semantic Similarity Comparisons

The question of whether two texts "mean the same thing" is a fundamental problem in NLP and closely related to the question of whether a text is relevant to a query. Researchers have explored similar approaches and have often even adopted the same models to tackle both problems.

A brief history of TR



BM25 formula

$$\text{BM25}(q,d) = \sum_{t \in q \cap d} \log \frac{N - \text{df}(t) + 0.5}{\text{df}(t) + 0.5} \cdot \frac{\text{tf}(t,d) \cdot (k_1 + 1)}{\text{tf}(t,d) + k_1 \cdot \left(1 - b + b \cdot \frac{l_d}{L}\right)} \qquad (2)$$

As BM25 is based on exact term matching, the score is derived from a sum of contributions from each query term that appears in the document. In more detail:

- The first component of the summation (the log term) is the idf (inverse document frequency) component: N is the total number of documents in the corpus, and $\text{df}(t)$ is the number of documents that contain term t (i.e., its document frequency).
- In the second component of the summation, $\text{tf}(t,d)$ represents the number of times term t appears in document d (i.e., its term frequency). The expression in the denominator involving b is responsible for performing length normalization, since collections usually have documents that differ in length: l_d is the length of document d while L is the average document length across all documents in the collection.

A provocative and historical question

As approaches based on deep learning (before BERT) required large amounts of training data, Lin [2018] posed the provocative question, asking if neural ranking models were actually better than "traditional" keyword-matching techniques in the absence of vast quantities of training data?

Under this limited data condition, studies showed thet most of the neural ranking methods were unable to beat the keyword search baseline

With BERT, though, everything changed, nearly overnight, as many researchers quickly demonstrated that with pretrained transformer models, large amounts of relevance judgments were not necessary to build effective models for text ranking.

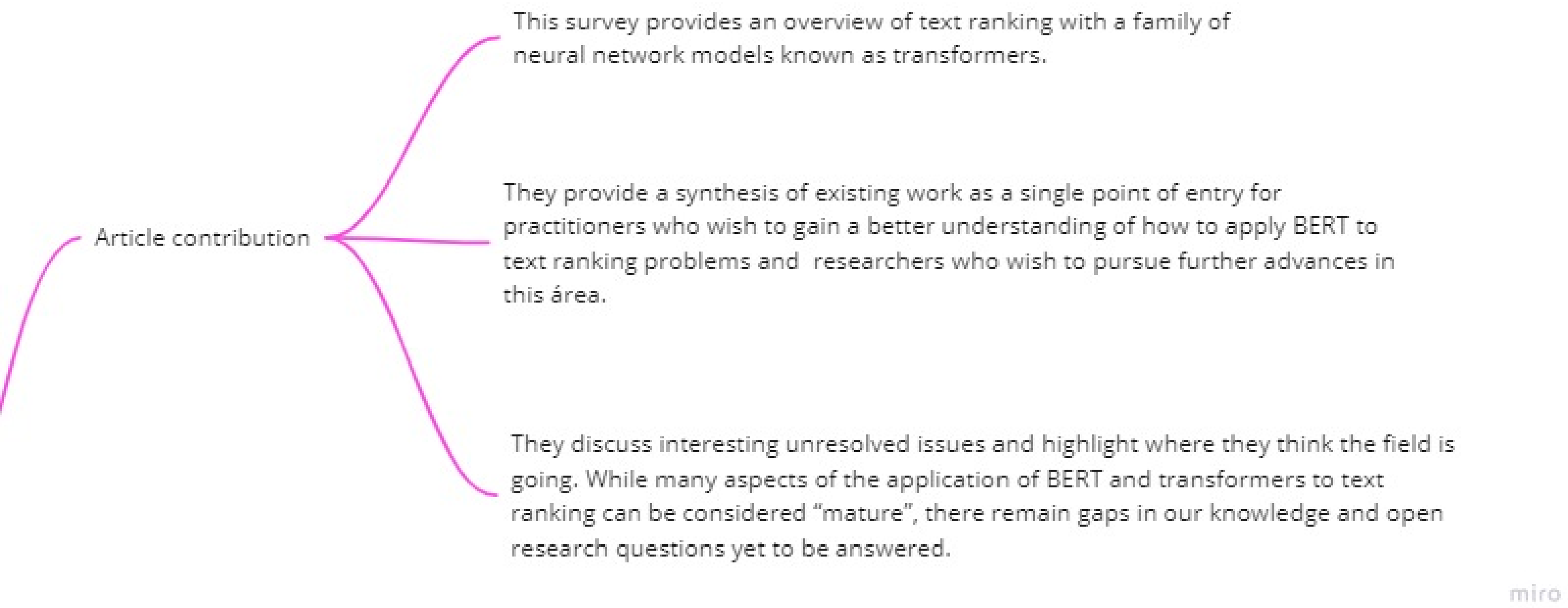
Bert era - the correct answer

The first application of BERT to text ranking was reported by Nogueira and Cho [2019] in January 2019 on the MS MARCO passage ranking test collection [Bajaj et al., 2018]

Within less than a week, effectiveness shot up by around eight points absolute, which corresponds to a ~30% relative gain

In the Deep Learning Track at TREC 2019, the organizers of the evaluation recognized BERT as a meaningful distinction that separated two different "eras" in the development of deep neural approaches to text ranking. BERT-based models achieved substantially higher effectiveness than pre-BERT models, across implementations by different teams.

BERT [Devlin et al., 2019] arrived on the scene in October 2018.


Article contribution
This survey provides an overview of text ranking with a family of neural network models known as transformers.
They provide a synthesis of existing work as a single point of entry for practitioners who wish to gain a better understanding of how to apply BERT to text ranking problems and researchers who wish to pursue further advances in this área.
They discuss interesting unresolved issues and highlight where they think the field is going. While many aspects of the application of BERT and transformers to text ranking can be considered “mature”, there remain gaps in our knowledge and open research questions yet to be answered.

Interesting/unexpected results

Considering the speed of searches and the overwhelming volume of discoveries, they elaborated an important research involving Multi-Stage Architectures for Reranking, Refining Query and Document Representations and Learned Dense Representations for Ranking.

- Basic doubts that may arise
  - Is there a boundary between the NLP and IR communities?
    - Researchers have explored similar approaches and have often even adopted the same models to tackle both problems.
    - In the context of learned dense representations for ranking, the connections between these two problems have become even more intertwined, bringing the NLP and IR communities closer and further erasing the boundaries between text ranking, question answering, paraphrase detection, and many related problems
  - How to train a list approach to loss function in learning methods for classification?

- Advanced topic to discuss
  - How to take advantage of the knowledge of synonyms between terms of a controlled vocabulary to enrich representations of queries or documents?
  - Is it possible Topic Detection with text retrieval?
    - Models that attempt to capture relevance for ad hoc retrieval can also be adapted for information filtering.Belkin and Croft [1992] famously argued that they represented "two sides of the same coin".

Advanced topic to discuss 3/4

Back translation is a good option for data augmentation?

- Given a corpus of English sentences, we could translate them automatically using a machine translation (MT) system, say, into French, and then translate those sentences back into English (this is called back-translation). With a good MT system, the resulting sentences are likely paraphrases of the original sentence, and using this technique we can automatically increase the quantity and diversity of the training examples that a model is exposed to.
- An apocryphal story from the 1960s goes that with an early English–Russian MT system, the phrase "The spirit is willing, but the flesh is weak" translated into Russian and back into English again became "The whisky is strong, but the meat is rotten" [Hutchins, 1995]
- Which translator to use?

Advanced topic to discuss 4/4

Is automatic indexing of descriptor terms extracted from controlled vocabularies still necessary?

Indexing
is the process of assigning to texts descriptors (also known as "index terms") normally extracted from thesauri or "controlled vocabularies". In the beginning, it was carried out by human specialists in the subject (or at least trained indexers).

Throughout the 1960s and 1970s, researchers and practitioners debated the merits of "automatic content analysis" (see, for example, Salton [1968]) vs. "traditional" human-based Indexing.

Harman [2019] goes as far as to call these "indexing wars": the battle between human-derived and automatically-generated index terms. This is somewhat reminiscent of the rule-based vs. statistical NLP "wars" that raged beginning in the late 1980s and into the 1990s, And goes to show how foundational shifts in thinking are often initially met with resistance.

Article Presentation
IA368DD_2023S1: Deep Learning aplicado a Sistemas de Buscas
Student: Marcus Vinícius Borela de Castro

Pretrained Transformers for Text Ranking BERT and Beyond - Chapter 3 (partial)

By: Jimmy Lin, Rodrigo Nogueira, and Andrew Yates
Link

miro

Main concepts - 1

Relevance classification

Convert the task into a text classification problem: to estimate the probability that each text belongs to the "relevant" class, and then at ranking (i.e., inference) time sort the texts by those estimates

Probability Ranking Principle

States that documents should be ranked in decreasing order of the estimated probability of relevance with respect to the information need

BERT (Bidirectional Encoder Representations from Transformers) [Devlin et al., 2019]

Is a neural network model for generating contextual embeddings for input sequences (which provide context-dependent representations of the input) in English, with a multilingual variant ("mBERT") that can process input in over 100 different languages. Here we focus only on the monolingual English model.

A language model in NLP provides a probability distribution over arbitrary sequences of text tokens. BERT and GPT are often grouped together and referred to collectively as pretrained language models. . In truth, coaxing such probabilities out of BERT require a bit of effort, and transformers in general can do much more than "traditional" language models

Main concepts - 2

The original paper presented only the BERTBase and BERTLarge configurations, with 12 and 24 transformer encoder layers, respectively. Turc et al. [2019] pretrained a greater variety of model sizes with the help of knowledge distillation;

In general, size correlates with effectiveness in downstream tasks, and thus these configurations are useful for exploring effectiveness/efficiency tradeoffs.

The number of layers, the hidden dimension size, and the number of attention heads are hyperparameters in the model architecture.

Ultimately, this led to an explosion of innovation in nearly all aspect of natural language processing, including applications to text ranking.

Its popularity (and rapid reproduction and replication of the impressive results) is largely due to the authors' wise decisions (and Google's approval) not only to open source the model implementation, but also to publicly release pre-trained models (which are computationally expensive to pre-train from scratch) by Hugging Face.

The final input representation to BERT for each token comprises the element-wise summation of its token embedding, segment embedding, and position embedding.

The special tokens [CLS] and [SEP] that need to be positioned at specific locations

Input template
The format how queries and candidate texts are fed to BERT
[CLS] q [SEP] d [SEP]

Basic doubts that may arise

What is pseudo-relevance feedback and RM3? (see page 57)

What is intraquery parallelism (page 58)?

"latency increases linearly with the number of candidates processed (in the absence of intra-query parallelism)"

How many points of difference indicate a noticeable difference in results?

7 points: This design is inspired by "text only" (as in T5) we observe a drop in MRR@10. This suggests that [SEP] indeed does have a special status in BERT, likely due to its extensive use in pretraining

6 points: we see that removing the segment embeddings has little impact, with only a small loss in MRR@10. This shows that monoBERT can distinguish query and document tokens using only the separator tokens.

- Advanced topic to discuss
  - What would be the size limit for a corpora to apply inference to all documents per query?
    - Applying inference to every text in a corpus for every user query is (obviously) impractical from the computational perspective (costly neural network inference and the linear growth of query latency with respect to corpus size).
    - A brute-force approach can be viable for small corpora,
  - Can changing the order from (q, d) to (d, q) in the input model lead to better results? Could this be related to these two aspects: can the CLS token be affected more by closer tokens and can candidate texts have a much greater variance in terms of length than queries?
    - (4) swapping query and document [CLS] d [SEP] q [SEP] led to a better MRR@10 0.366 (X 0.365)

UNICAMP

Article Presentation
IA368DD_2023S1: Deep Learning aplicado a Sistemas de Buscas
Student: Marcus Vinícius Borela de Castro

Pretrained Transformers for Text Ranking BERT and Beyond - Cap 3

By: Jimmy Lin, Rodrigo Nogueira, and Andrew Yates
[Link]

Article contribution

Interesting/unexpected results

Basic doubts that may arise

Advanced topic to discuss

Main concepts - 1

Main concepts

Código para Aula 3

IA368DD_2023S1: Deep Learning aplicado a Sistemas de Buscas

Marcus Vinícius Borela de Castro

Classificação de Texto e Reranqueadores

Criei rastro no neptune.ai dos treinamentos.

Resultados interessantes

Base de validação fixa, leiaute qp, para comparação e mais próxima do teste no reranking)
As acurácias foram respectivamente: 93,9%, 93,3% e 94,2%

Parece interessante dobrar os dados de treinamento (pendente de ser confirmado no reranking)

Experimentei 3 input_layout: /pq/qppq
qp: query+passage;
pq: passage+query
qppq: qp union pq

Obs.: qppq com o dobro de registros no treinamento


Usei asserts no meio do código
Validei os códigos das funções com o chatgpt
Criei dados fictícios para testar a classe do dataset
Técnicas usadas para buscar a correção
miro

Dúvidas

No dataset: Precisamos gerar ou o modelo gera para nós quando não passamos?

Dúvida: Não salvei o tokenizador pois ele não foi alterado. Precisa?

Observações

- Criei a minha classe de Dataset. Nela, gerei os token_type_ids (0 e 1 para as 2 sequências)
- Ajustado erro no apóstrofe no texto de entrada (como em don't)
- Usei chat_gpt em todas as fases. Inclusive se mostrou muito útil pra documentar funções. (Veja no início: inicializa_seed, config_debug e config_display)
- Retomei (e aproveitei) conteúdo de cadernos de cursos anteriores na Unicamp

Ressalvas

Na fase do reranking, o ndcg_cut_10 deu baixo 0.0441.
Suspeita de causa: não ter usado torch.save(model) ao invés de model.save_pretrained().
Segundo ChatGPT, há uma diferença: model.save_pretrained() salva os pesos do modelo, bem como todos os outros artefatos necessários para carregar e inicializar o modelo em uma instância específica de um modelo do Transformers, enquanto torch.save(model) salva apenas os pesos do modelo.

Obs.: Fiz o ajuste no caderno e deixei treinando para ver a diferença.