UNIVERSITY
OF OSLO

Retrieval and assisting language model generations

Lucas Georges Gabriel Charpentier
Pacheco

31st March 2025



Contents

1. Introduction
   - Limitations of Language Models
   - Information retrieval
2. Sparse Retrieval
3. Dense Retrieval
4. Retrieval-augmented Language Models
5. RAG pipeline and applications
6. Problems with Retrieval
7. Long Context Language Models

Introduction

- PhD Candidate in the Language Technology Group at the UiO
- Work on de-anonymization of text documents as a method to ameliorate text anonymization

Knowledge Cut-off

- Models are pre-trained on a given dataset.
- Datasets are a snapshot of a given time period.
- Knowledge is limited to the domain and time period within the dataset.

Opaque

- When we get a response from a language model, we do not know where it came from
- Potentially hallucination?
- What if we want to understand more, read the source?

Rigid

- No simple or cheap way to internally change weights.
- Continual pre-training might lead to forgetting previous knowledge.
- Continual pre-training is expensive

What is Information Retrieval?

Problem A user/system want to get information relevant to a specific query.

Goal Create a system that can automatically search a database based on a query, and return the most relevant documents/images/tables/etc.

Search Engines

- One of the most commonly used example of *Information Retrieval* is *Search Engines*.
- Takes a query from a user and tries to return a list of the best possible answers/websites/etc.
- (Various optimizations for speed and efficiency)

Contents

1. Introduction
2. Sparse Retrieval
   - Introduction
   - BM25
   - Recent Expansions
3. Dense Retrieval
4. Retrieval-augmented Language Models
5. RAG pipeline and applications
6. Problems with Retrieval
7. Long Context Language Models

How can we do retrieval?

- We need a datastore containing the knowledge we want to retrieve from. (In this case, we consider only datastores containing exclusively free-text)
- We need a model able to retrieve the data based on a free-text query.
- We need a way to index the datastore to enable comparisons between the query and texts in the database.

Non-parametric solution

- In previous lectures and for Mandatory Assignment 1, you saw bag-of-words.
- We can also use this for retrieval.
- Slight variation: we will use a TF–IDF (term frequency–inverse document frequency) inverse dictionary.

$$\mathrm{tf}(t, d) = \frac{f_{d,t}}{\sum_{t' \in d} f_{t',d}}$$

$$\mathrm{idf}(t) = \ln\left(\frac{N}{n_t + 1}\right)$$

BM25

- The most famous sparse retrieval technique is called BM25 (Robertson et al., 1995).
- It was developed in the 70s/80s and first used in the 80s/90s under the name Okapi BM25.
- It is a variation of TF–IDF.
- It is still used today.

BM25 contd.

$$\text{BM}_{25}(Q, d) = \sum_{i=1}^{n} \text{idf}(t_i) \cdot \frac{f(t_i, d) \cdot (k_1 + 1)}{f(t_i, d) + k_1 \cdot \left(1 - b + b \cdot \frac{|d|}{\bar{D}_l}\right)}$$

$$\text{idf}(t) = \ln\left(\frac{N - n(t) + 0.5}{n(t) + 0.5} + 1\right)$$

- Q is a query containing terms $t_1, t_2, \ldots, t_n$.
- $f(t, d)$ is the frequency of term t in document d.
- N is the number of documents in the datastore.
- $n(t)$ is the number of documents containing the term t.
- $\bar{D}_l$ is the average document length.
- k_1, b are hyperparameters set beforehand.

Pitfalls of BM25

- Does not take into account semantics.
- For example:
  - Norway beat Brazil in the World Cup group stages *vs.* Brazil beat Norway in the World Cup group stages.
  - The man owned a dog *vs.* The man owned a puppy *vs.* The man owned a car
- Can only take one query at a time.

Extensions

- BM25F (Robertson et al., 2004) considers the document as composed of several fields, each with varying importance, term frequency, and length.
- BM25+ (Lv and Zhai, 2011) helps fix the short-long document problem by adding an additional free parameter
- BM$\mathcal{X}$ (Li et al., 2024a) adds entropy weighting to control for repetitive text as well as weighted query augmentation to incorporate semantics.

Contents

1 Introduction

2 Sparse Retrieval

3 Dense Retrieval
- Machine Reading Comprehension
- Dense Passage Retrieval
- ColBERT

4 Retrieval-augmented Language Models

5 RAG pipeline and applications

6 Problems with Retrieval

7 Long Context Language Models

What is a dense retriever?

- Instead of sparse retrieval models, we can use language models to perform retrieval.
- This is called dense retrieval.
- Due to the encoded vectors being dense.
- Usually, we have a document encoder and a query encoder.

How do we train a dense retriever?

Need A pair of pre-trained language models (could be the same), a labeled query-document dataset, a datastore

Goal Given a triple (q, d^+, d^-), where q is a query, d^+ is a relevant document, and d^- is an irrelevant document, we want to train two encoders, one for the query (Enc_q) and one for the documents (Enc_d), such that $\text{sim}(\text{Enc}_q(q), \text{Enc}_d(d^+)) > \text{sim}(\text{Enc}_q(q), \text{Enc}_d(d^-))$, where sim is a similarity measure such as cosine similarity.

Caveat Labeled data is not always available; we could use weak supervision to simulate it.

MS MARCO

- A common dataset to use to train retrievers is the **M**icro**S**oft **MA**chine **R**eading **CO**mprehension dataset (Nguyen et al., 2016).
- It is human-generated.
- Contains various labeled datasets for tasks such as question-answering, passage retrieval, keyphrase extraction, etc.

Dense Passage Retrieval (DPR)

Goal Score the similarity between a query and a document.

Needs Two encoder models, one for the queries and one for the documents, and a datastore.

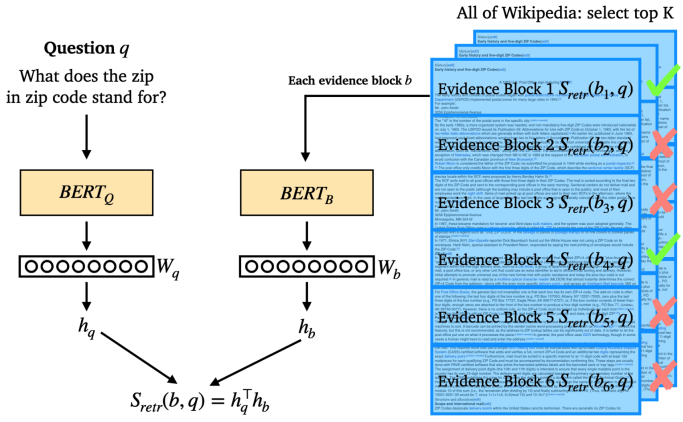
Method Encode each the query and document into a single *n*-dimensional vector to run a similarity measure on.

Proposed by Karpukhin et al. (2020).

DPR

Retriever score: $S_{retr}(b, q)$

$$h_q = \mathbf{W_q}\mathbf{BERT}_Q(q)[\text{CLS}]$$
$$h_b = \mathbf{W_b}\mathbf{BERT}_B(b)[\text{CLS}]$$
$$S_{retr}(b, q) = h_q^\top h_b$$



From the CS224n stanford lectures: https://web.stanford.edu/class/cs224n/ (Lecture 13)

ColBERT

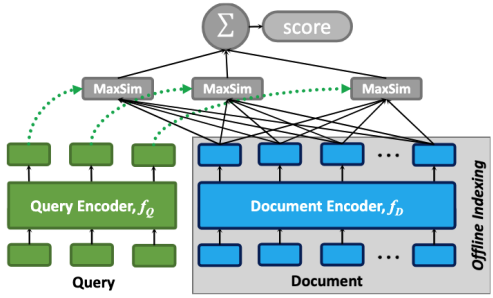
Goal Fine-grained similarity scoring between a query and a document.

Needs Two encoder models, one for the queries and one for the documents, and a datastore.

Method Encode each the query and document into multiple *n*-dimensional vectors (one per token), calculate the similarity score between a token of the query and every token of the document, retrieve the maximum, and sum over all tokens of the query.

Proposed by Khattab and Zaharia (2020).

ColBERT contd.



ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT (Khattab and Zaharia, 2020)

Contents

1. Introduction
2. Sparse Retrieval
3. Dense Retrieval
4. Retrieval-augmented Language Models
   - Introduction & Goals
   - REALM
   - RAG
   - RETRO/ATLAS
5. RAG pipeline and applications
6. Problems with Retrieval
7. Long Context Language Models

Bridging Language Models and Retrieval

- We now have a retriever model.
- How can we use it with a language model?
- Instead of only relying on the internal knowledge of the model, we can use a datastore and retriever model to enhance the prediction of the model
- Multiple ways to integrate this knowledge into the language model.

Advantages

Knowledge By simply changing the datastore, we can edit the knowledge of the language model. This allows us to update the model by proxy without re-training.

Transparency Since the retriever gives us a document, we can see where the language model takes its information from.

Flexibility As mentioned before, we can easily change the domain or time period the model focuses on by changing our datastore.

Initial Applications

- While the adoption of such models is widespread.
- It was first used for open-book question-answering.

Question-Answering

What is it It is exactly what it sounds like, the model gets a question, and its goal is to answer.

How does it work Traditionally, it was called Machine Comprehension since the model would get both a question and a passage, and the goal was to extract the answer from the passage.

How does retrieval come into play In the case of open-book question answering, we have a collection of texts from which a retriever model needs to find one relevant for the model to answer the question.

REALM

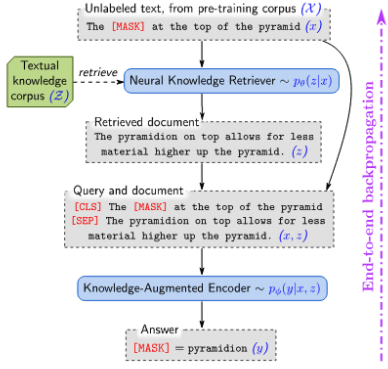
Goal Train end-to-end a masked language model and retriever.

Needs Three encoder models, two for the retriever and one for the language model, and a datastore.

Method For a given masked text, a retriever gets the *k* most relevant documents (chunks) from a datastore. These are then appended to the masked text to create *k* examples to give to the language model. The outputs of these *k* examples are then marginalized over the score of the retrieval to give the final prediction.

Proposed by Guu et al. (2020).

REALM contd.



REALM: Retrieval-Augmented Language Model Pre-Training (Guu et al., 2020)

RAG

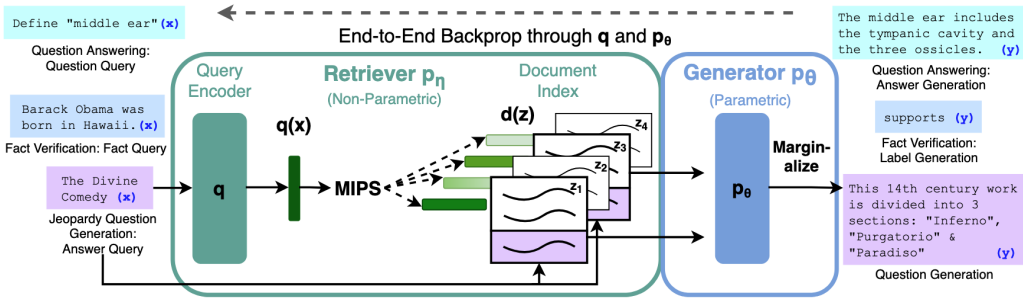
Goal Train end-to-end a generative language model and retriever.

Needs Two encoder models for the retriever and one causal/generative model for the language model, and a datastore.

Method For a given text, extract the *k* most relevant documents for either the entire text or per-token. These are prepended to the text to create *k* examples to give to the language model. The outputs of these *k* examples are then marginalized over the score of the retrieval to give the final prediction.

Proposed by Lewis et al. (2020).

RAG contd.



Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks (Lewis et al., 2020)

What about training both together from scratch?

- Previous methods used pre-trained models as a starting point.
- What if we train from scratch with retrieval?
- Could it possibly change the performance of the underlying model? (Samuel et al., 2024)

RETRO

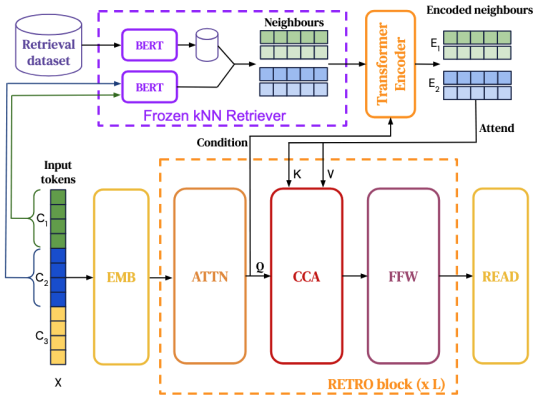
Goal Train from scratch a causal language model with retrieval embedded

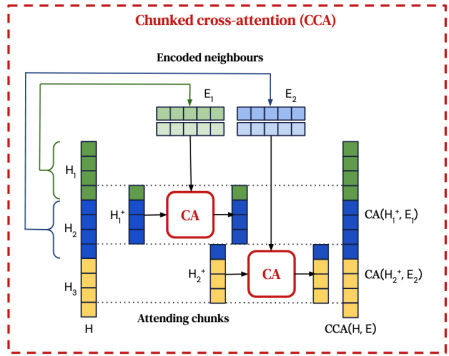
Needs One frozen encoder model as both query and document encoder, a blank causal language model, and a datastore of vectors and their continuations.

Method Split the input tokens into *n* chunks. For each chunk except the last one, retrieve the *k* most relevant documents and their continuation. Then, for each layer of the causal language model, do chunked cross-attention on each chunk.

Proposed by Borgeaud et al. (2022).

RETRO contd.





Improving language models by retrieving from trillions of tokens (Borgeaud et al., 2022)

ATLAS

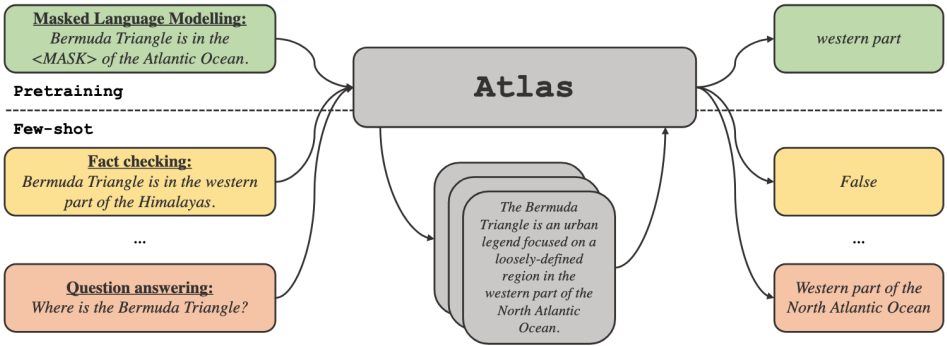
Goal Train end-to-end a sequence-to-sequence model from scratch and a retriever model.

Needs Two encoder models for the retriever, a blank encoder-decoder model, and a datastore.

Method Given a masked text (for inference, can be any query) retrieve the *n* most relevant documents. Create *n* examples to feed to the encoder by concatenating the masked text and retrieved documents. Concatenate the *n* outputs of the encoder to feed with cross-attention to the decoder.

Proposed by Izacard et al. (2023).

ATLAS contd.



Atlas: Few-shot learning with retrieval augmented language models (Izacard et al., 2023)

Contents

1. Introduction
2. Sparse Retrieval
3. Dense Retrieval
4. Retrieval-augmented Language Models
5. RAG pipeline and applications
   - RAG pipeline
6. Problems with Retrieval
7. Long Context Language Models

What to use to combine?

- As RAG has become popular, many libraries have appeared to simplify its application.
- They usually involve using a trained retriever, a trained language model, and a datastore.

Implementations

LangChain[1] Can be used to easily create any language model application. Has various API functionalities to chunk text, call language models, set up the retrieval pipeline, etc.

HayStack[2] More focused on RAG, but can also be used for other applications (such as chatbots). Contains various trained retriever models that can be loaded directly into a pipeline.

[1] https://python.langchain.com/docs/introduction/

[2] https://haystack.deepset.ai/

Contents

1. Introduction
2. Sparse Retrieval
3. Dense Retrieval
4. Retrieval-augmented Language Models
5. RAG pipeline and applications
6. Problems with Retrieval
   - Size of Datastore
   - Efficiency of Search time
   - Requires a second model
7. Long Context Language Models

What is your Database?

- Retrieval is great, but it does have some downsides.
- The datastore can be quite big and needs a lot of memory to store.
- This is especially true if you want to store encoded vectors rather than free-text.

Space Requirements

- Each character is usually stored in one byte (before compression)
- A single encoded vector is usually 768 float values, which equates to 3 072 bytes (and is usually harder to compress due to it being dense).
- In the case of ColBERT, this would mean that the resulting datastore would be at least two orders of magnitude bigger than its free-text counterpart.

Online vs. Offline

- What if we create the vectors on-the-fly?
- This would require a large amount of compute and time for any non-trivial datastore.

Offline optimizations

- We can optimize the offline datastore.
  - **Size** We can quantize the vectors into smaller sizes by either shrinking the number of dimensions or reducing the precision.
  - **Speed** We can optimize the search time by various means; batching, approximate search, etc.

FAISS

- Faiss (Douze et al., 2024) is a library of **A**pproximate **N**earest **N**eighbours **S**earch algorithms.
- It lets you create an efficient vector datastore, both in terms of speed and memory.
- Allows usage on the GPU and CPU.
- It allows for various index solutions such as exact search, quantized search, **H**ierarchical **N**avigable **S**mall **W**orld graph exploration, etc.
- It allows for various comparison metrics, such as cosine similarity, L2, or inner product.

Problems of using a retriever model

- We need to train a secondary model to retrieve information.
- Can it adapt to different domains without re-training?
- Can we do without a retriever model?

Contents

1. Introduction
2. Sparse Retrieval
3. Dense Retrieval
4. Retrieval-augmented Language Models
5. RAG pipeline and applications
6. Problems with Retrieval
7. Long Context Language Models
   - Skip the second model
   - Problems
   - Potential Solutions

Longer Context Window

- With the introduction of various optimizations to language models, we are able to have longer context windows.
- Before, we were limited to a few thousand tokens; now we can have up to a million.
- This is equivalent to 1 500 pages of text.

Using an LLM for both retrieval and generation at the same time

- Instead of using a retriever, we can put our datastore as context for the language model.
- We let the model decide which parts of the text are important.
- This can help with multi-hop reasoning (Li et al., 2024b).
- This is called **L**ong **C**ontext **L**anguage **M**odeling.

Long Context Window, yes, but is it enough?

- Even with one million tokens, this is still too small for most datastores.
- Most datastores contain tens of thousands of documents.
- In addition, using such a big context is slow and requires much memory.

Does not necessarily work out of the box

- We train a retriever to find the most relevant text.
- Will an out-of-the-box language model be able to do the same?
- Will we still have the same transparency?

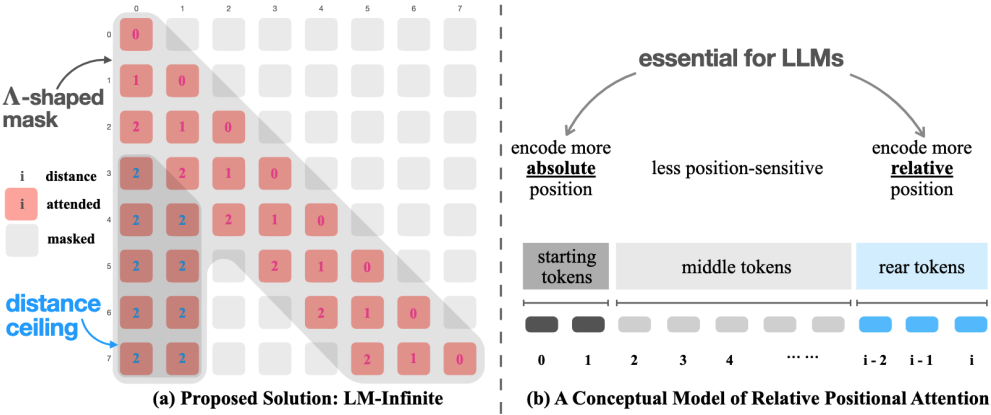
Use a sparse retriever for initial filtering

- Rather than passing the entire datastore to the language model, we filter it using a sparse retriever.
- Ensures that the documents are somewhat relevant.
- Reduces the memory and compute consumption of the language model.

Extended Context Window without re-training

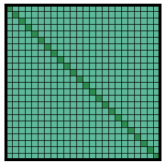
- We could extend the context window.
- There exist techniques that can do this without re-training.
- It could also be part of the training of the model.

LM-Infinite

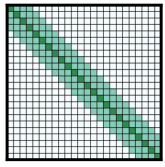


LM-Infinite: Zero-Shot Extreme Length Generalization for Large Language Models (Han et al., 2024)

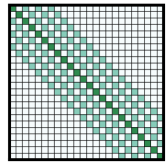
Sliding Window



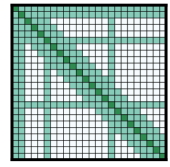
(a) Full n^2 attention



(b) Sliding window attention



(c) Dilated sliding window



(d) Global+sliding window

Longformer: The Long-Document Transformer (Beltagy et al., 2020)

Compression

- Very active area of research for solutions to LCLM.
- Instead of using the entire document, you train/few-shot a language model to create a summary of a document (compression).
- Then you add all the “compressed” documents to the context and do your retrieval-augmented generation.
- Various different implementation exist such as LLMLingua (Jiang et al., 2023), RECOMP (Xu et al., 2024), CompAct (Yoon et al., 2024), or CoLoR (Seo et al., 2024).

References I

Beltagy, I., Peters, M. E., and Cohan, A. (2020). Longformer: The long-document transformer.

Borgeaud, S., Mensch, A., Hoffmann, J., Cai, T., Rutherford, E., Millican, K., Van Den Driessche, G. B., Lespiau, J.-B., Damoc, B., Clark, A., et al. (2022). Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR.

Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvasy, G., Mazaré, P.-E., Lomeli, M., Hosseini, L., and Jégou, H. (2024). The faiss library.

Guu, K., Lee, K., Tung, Z., Pasupat, P., and Chang, M.-W. (2020). Realm: retrieval-augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org.

References II

Han, C., Wang, Q., Peng, H., Xiong, W., Chen, Y., Ji, H., and Wang, S. (2024). LM-infinite: Zero-shot extreme length generalization for large language models. In Duh, K., Gomez, H., and Bethard, S., editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3991–4008, Mexico City, Mexico. Association for Computational Linguistics.

Izacard, G., Lewis, P., Lomeli, M., Hosseini, L., Petroni, F., Schick, T., Dwivedi-Yu, J., Joulin, A., Riedel, S., and Grave, E. (2023). Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 24(251):1–43.

References III

Jiang, H., Wu, Q., Lin, C.-Y., Yang, Y., and Qiu, L. (2023). LLMLingua: Compressing prompts for accelerated inference of large language models. In Bouamor, H., Pino, J., and Bali, K., editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13358–13376, Singapore. Association for Computational Linguistics.

Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. (2020). Dense passage retrieval for open-domain question answering. In Webber, B., Cohn, T., He, Y., and Liu, Y., editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

References IV

Khattab, O. and Zaharia, M. (2020). Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., and Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

References V

Li, X., Lipp, J., Shakir, A., Huang, R., and Li, J. (2024a). Bmx: Entropy-weighted similarity and semantic-enhanced lexical search. *arXiv preprint arXiv:2408.06643*.

Li, Z., Li, C., Zhang, M., Mei, Q., and Bendersky, M. (2024b). Retrieval augmented generation or long-context LLMs? a comprehensive study and hybrid approach. In Dernoncourt, F., Preoţiuc-Pietro, D., and Shimorina, A., editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 881–893, Miami, Florida, US. Association for Computational Linguistics.

Lv, Y. and Zhai, C. (2011). Lower-bounding term frequency normalization. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 7–16.

References VI

Nguyen, T., Rosenberg, M., Song, X., Gao, J., Tiwary, S., Majumder, R., and Deng, L. (2016). MS MARCO: A human generated machine reading comprehension dataset. *CoRR*, abs/1611.09268.

Robertson, S., Zaragoza, H., and Taylor, M. (2004). Simple bm25 extension to multiple weighted fields. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 42–49.

Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M. M., Gatford, M., et al. (1995). Okapi at trec-3. *Nist Special Publication Sp*, 109:109.

References VII

Samuel, D., Charpentier, L., and Wold, S. (2024). More room for language: Investigating the effect of retrieval on language models. In Duh, K., Gomez, H., and Bethard, S., editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 282–305, Mexico City, Mexico. Association for Computational Linguistics.

Seo, M., Baek, J., Lee, S., and Hwang, S. J. (2024). Efficient long context language model retrieval with compression. *arXiv preprint arXiv:2412.18232*.

Xu, F., Shi, W., and Choi, E. (2024). Recomp: Improving retrieval-augmented lms with context compression and selective augmentation. In *The Twelfth International Conference on Learning Representations*.

References VIII

Yoon, C., Lee, T., Hwang, H., Jeong, M., and Kang, J. (2024). Compact: Compressing retrieved documents actively for question answering. *arXiv preprint arXiv:2407.09014*.

Lucas Georges Gabriel Charpentier Pacheco

Retrieval and assisting language model generations