COMS 4774 - Final Project Report
Unsupervised Document Retrieval by Contrastive Learning

Ryan Anselm, Kuan-Yao Huang, Todd Morrill, Howard Yong

December 12, 2022

Introduction

Document retrieval is a canonical task in computer science where given a collection of documents, an input query is specified by the user and a ranked list of documents with the greatest relevance to the query is returned. The particular notion of 'relevance' used and the formats of queries and documents vary between different instances of the task. Recently, self-supervised learning has emerged as a prominent class of techniques that attempt to provide a supervisory signal to unlabeled data by leveraging structure in the data itself. Self-supervised learning excels for applications that rely on extremely large amounts of data to the point where it becomes infeasible for humans to label all the data manually, such as in natural language processing, where vast bodies of text on the internet lacking labels can be used in practice as the dataset in a self-supervised setting, but not in a supervised setting. Contrastive learning is a type of self-supervised learning that derives a supervisory signal by maximizing similarity between 'positive pairs' of training examples and minimizing similarity between 'negative pairs' of training examples based on a contrastive loss function. In the natural language processing context, a positive pair of training examples could be a string of text and the same string of text with a series of data augmentations applied to it, or alternatively a second string taken from the neighborhood of the first string, and negative pair would be anything that does not meet the criteria for being a positive pair.

We expand upon a previous approach to the document retrieval task based on contrastive learning that attempts to improve upon traditional approaches by going beyond the use of lexical similarity alone, instead using learned vectorial representations of query/document semantic meaning from a contrastive learning algorithm. We experiment with modifying this approach to include an improved quality of contrastive pairs given to the learning algorithm by eliminating cases of document overlap, and a different framework for contrastive learning than the one used in the previous approach.

Related Work

Term Frequency Inverse Document Frequency

Term Frequency Inverse Document Frequency (TF-IDF) (Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. Journal of documentation, 1972) is a classic unsupervised technique used for document retrieval. The TF-IDF algorithm creates a vector representation of queries and documents and computes a dot product or a cosine similarity score between the query and documents to rank results. TF-IDF has several benefits, namely that: 1) it is easy to implement, 2) it is unsupervised and does not require labeled data, and 3) tends to be very precise because it is entirely keyword based. However, TF-IDF does have some limitations, namely: 1) it ignores word order (i.e. bag of words) and 2) doesn't handle synonyms or abbreviations. For example, a search for "ML" may not retrieve documents related to "machine learning", while a neural embedding based is likely to capture the similarity between the two queries. BM25 (Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. Okapi at TREC-3. NIST Special Publication Sp, 1995.), an extension of TF-IDF and TF-IDF are still widely used in industry.

Deep Document Retrieval

Modern supervised document retrieval systems outperform traditional unsupervised methods. In particular, pre-trained transformer based language models (Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proc. NAACL, 2019. 3) have become the standard backbone for document retrieval systems. The modern neural approach to the document retrieval problem is a 2 stage system [Nils Reimers. Retrieve and Re-Rank — Sentence-Transformers documentation. 2022. Url:`https://www.sbert.net/examples/applications/retrieve_rerank/README.html`.]. Stage 1 is a retrieval stage, which retrieves 100s or 1,000s of potentially relevant documents given the query. A typical architecture is a bi-encoder, which can be thought of as a siamese network, where both encoders are pre-trained transformer models. The reason for a bi-encoder (as opposed to a cross-encoder, see below) is that documents can be encoded ahead of time and so when a new query comes in, the system

only needs to encode the query. An inexpensive similarity function such as the dot product or cosine similarity is computed between the encoded query and the encoded documents. This inexpensive similarity function is critical to scaling to databases of billions of documents. Stage 2 is a re-ranking phase, where a more expensive model is used to provide a final ranking of a small set of candidate documents (e.g. 1,000). It is common to use cross-encoder at this stage, which concatenates the query text together with the document text and allows the attention mechanism to assess the relevance of the document to the query with a high degree of precision. The trouble is that the cross-encoder requires more computation and cannot be pre-computed, hence the 2 stage system. The benefit of a supervised system as described above is that it is highly accurate and improves with more labeled data. The downside is that collecting this labeled data is expensive and often models must be adapted from domain-to-domain (e.g. healthcare vs. finance, etc.), which motivates the use of unsupervised methods to adapt to a new domain.

Self-Supervised Learning

Self-supervised learning is emerging as a very powerful approach to learning high quality representations of data (e.g. audio, video, text, etc.) using deep learning models without the need for any manually curated labeled data. One of the first neural methods to utilized self-supervised contrastive learning was word2vec (Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems, pp. 3111–3119, 2013.). Since then, transformer based language models have adopted a self-supervised training objective. Noise contrastive estimation is a learning technique whereby a system must assign more probability to a true distribution versus a noise distribution (`https://proceedings.mlr.press/v9/gutmann10a/gutmann10a.pdf`). The system learns to push positive examples (e.g. a query and a relevant document) closer together in euclidean space and push apart negative examples. Several implementation level details have been explored to increase the challenge of discriminating between the true distribution and the noise distribution. In particular, MoCo (`https://arxiv.org/abs/1911.05722`) keeps a queue of many thousands of negative examples that increase the challenge of detecting the positive example. Similarly, SimCLR (`https://arxiv.org/abs/2002.05709`) makes use of large in-batch negative examples to increase the size of the noise distribution. With respect to document retrieval, the most similar work to this report is Contriever (`https://arxiv.org/abs/2112.09118`), which our work builds upon, and more recent work using in-batch loss techniques (`https://arxiv.org/abs/2201.10005`). These ideas have also had a huge impact on computer vision (Learning Transferable Visual Models From Natural Language Supervision, Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, Ilya Sutskever) and speech recognition (wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations, Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, Michael Auli).

Methods

Datasets

We used Wikipedia to train our model without supervision and use BEIR benchmark to evaluate the effectiveness of our methods. The Wikipedia dataset contains 6,583,470 articles, from which we can generate positive and negative training examples. The citations, references, and other irrelevant information in the articles are removed. There are no test dataset nor label in this dataset.

The BEIR benchmark dataset is a collection of document retrieval datasets used to evaluate the performance of document retrieval systems. It includes MSMARCO, the Quora question-answer pairs dataset, and many more. Some of the datasets are much harder than others. For example, the MSMARCO dataset contains 3.2 million documents and all of them can be the candidate to be retrieved.

Architectures

Our architectures are works from two separate papers:

- Momentum Contrastive Learning(MoCo)[`https://arxiv.org/abs/1911.05722`], and
- A Simple Framework for Contrastive Learning of Visual Representations(SimCLR)[`https://arxiv.org/abs/2002.05709`].

Contrastive learning is a popular field of representation learning. It aims at attracting the representation of similar pairs together and repelling the dissimilar pairs away. A common choice of loss function is the information noise contrastive estimation loss:

$$\mathcal{L} = \frac{\exp\left(q \cdot k_i/\tau\right)}{\sum_{j=1}^{N} \exp\left(q \cdot k_j/\tau\right)}$$

MoCo

Providing high quality and diverse examples by augmentation is at the heart of contrastive learning method. The positive pair x_q(query) and x_k(positive key) are sentences from the similar document. We use two different augmentations selected from random text deletion, masking, and back-translation to preprocess the data.

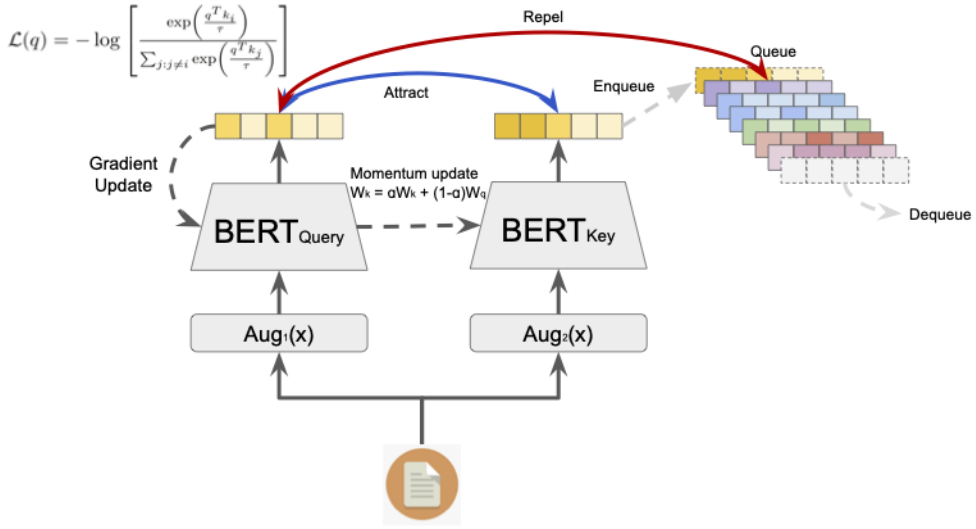

Figure 1: Architecture of MoCo

The contrastive learning usually requires more negative examples than positive ones. The MoCo model uses a queue to store 65,000 negative representations from previous iterations and use the key/query encoders to obtain the positive example pair at each step. Then the model tried to learn how to encode the representation such that the positive example has greater cosine similarity than that between query and negative examples. One way to interpret the loss function is that we are performing a 65,001-way classification using the cosine similarity from query to 1 positive and 65,000 negative examples.

When optimizing the objective function, only the query encoder is updated by gradient descent and the key encoder is updated by the moving average of the query and key encoder.

$$W_k(t+1) \leftarrow \alpha W_k(t) + (1-\alpha) W_q(t)$$

Usually an α value of 0.95-0.99 is desired. That is the "momentum" part of the model.

After each iteration, we enqueue the key representation into the queue of the negative example so it will become the negative example in the next iteration. In addition, we will dequeue the oldest representation from the queue for each iteration to keep the size of the queue fixed. At the beginning, the queue is filled up with random Gaussian vector normalized to 1. It can be shown that the expected value of cosine similarity between a random unit vector and a random Gaussian vector is 0 and the variance is close to zero.

The benefit of MoCo is it can create as much negative examples as we wish. In addition, the momentum update of the key encoder guarantees a stable change of the encoding of negative examples. As shown in the original paper of MoCo, the standard deviation of all the negative examples are shown to be lower for momentum update compared to update the key encoder by gradient descent.

The drawback of MoCo also lies in the design of queue. Since the negative representations are collected from previous iterations, the closer the data to the tail of queue, the representation is more inaccurate due to staleness. It makes the noisy training process of MoCo.

SimCLR

The Siamese-like network such as MoCo was popular in the field of contrastive learning. However, models with only one encoder that creating positive and negative examples within a batch are now proven to be stabler at training phase and naturally does not have the staleness problem as we describe in MoCo.

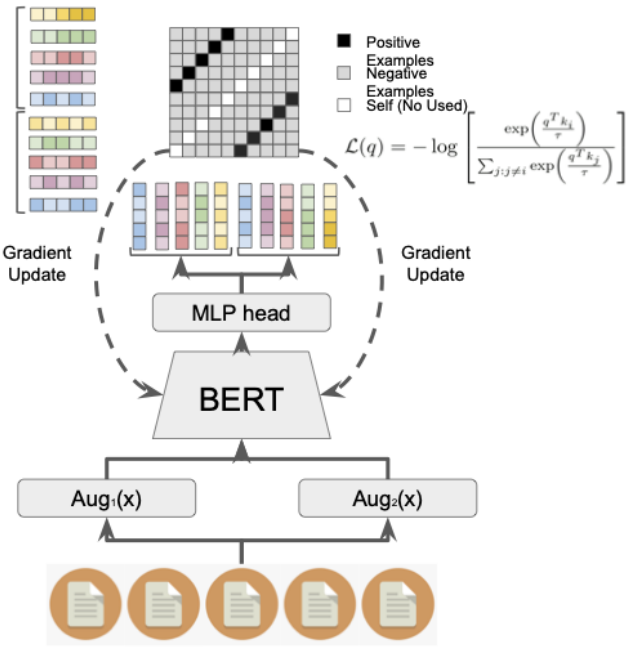

Figure 2: Architecture of SimCLR

It is natural to collect the positive and negative examples in a batch of data when doing the batch-training. We can perform two different augmentations on a batch of data and concatenate their representations, and then doing the tensor product of the representation. The similarity table shown in the figure indicates that the representations from similar documents are positive examples and different documents for negative examples. When ignoring the similarity of identity, there is one positive example for each data and $B-2$ negative examples, where B is the batch size. Then we can compute the contrastive loss similar to MoCo.

The benefit of SimCLR is that it requires only one encoder, saving the memory to store and update the bulky transformer. In addition, the representation of positive and negative examples are always up-to-date, unlike in MoCo.

However, to provide the model with more negative examples, a larger batch size is required. In our experimental settings, we can only achieve a batch size of 100 across 3 GPU. On the other hand, in the experiment of MoCo, we are able to collect 65,000 negative example with one positive example.

Joint Training with MoCo and SimCLR

Knowing the pros and cons of the two architecture, we are curious whether jointly training with the two techniques improves the performance of the model? Can MoCo provide a coarse direction for gradient descent using low quality but abundant negative examples, and SimCLR provide an accurate direction of gradient updates using high quality but scarce negative examples at the same time? Our curiosity lead to the experiment on jointly training the model with both MoCo and SimCLR. We share the query encoder of MoCo to SimCLR. In this case, we are able to provide a large number of slightly outdated negative examples and a batch of high quality representations. The training objective can be written as a linear combination of two losses:

$$\mathcal{L} = \mathcal{L}_{\text{MoCo}} + \lambda \cdot \mathcal{L}_{\text{SimCLR}}$$

The hyperparameter λ stands for different importance for the two objectives.

Experiments

This investigation explored 2 key experiments on top of the baseline. The baseline model used to evaluate results against was a model released by Facebook Research, called Contriever. The first key experiment we launched was directed at data set representation during pre-training. The original paper proposed a training regime that invalidated document boundaries. This was confirmed by reviewing their data pipeline. The original authors of Contriever prepared each data set for training by concatenating tokenized text from all documents into a single, contiguous array and proceeded to sample key-query pairs from this new sequence. The tokenizer employed in all of our experiments was the pre-trained tokenizer released by Contriever. The issue with this method of data representation is that it invalidates a key premise of the document retrieval problem. When building positive samples of matching key-query pairs, the objective during training is to learn a transformation such that the 2 samples of text are pushed closer together in vector space. However, if text from other documents leak into a given

positive sample, the words may distort the model's understanding of the underlying "topic". Additionally, it may cause the model to learn word associations that are not really pertinent to a given positive sample.

Resolving this issue is our first contribution. We updated the representation of the training data set to a dictionary structure. Each document had a *docId* as its key. The value was another embedded dictionary that contained *tokens* and other meta information. We concatenated the tokens from each document into a contiguous array as well but with a special token following each document. In the training pipeline, random crops are selected to produce key-query pairs. We first search for the presence of the special token(s) in this crop. If we identify the special token(s), then we proceed with the maximum length sub-array for constructing key-query pairs.

The second key experiment in this paper is training with a custom loss function combining both the objectives from MoCo and SimCLR. The joint training was discussed in the previous section. This is our second critical contribution. The results of our experiments are presented in the following tables.

nDCG@10			
	Baseline	Respecting Doc. Boundaries	+SimCLR Loss
Quora	2.315	3.35	4.2
ArguAna	7.732	3.136	35.1
SCIDOCS	0.238	0.393	0.4
Trec-COVID	0.317	0	0.3
FiQA-2018	0.143	0.925	0.4
NFCorpus	4.241	5.613	10
SciFact	9.797	10.933	35.4
Recall@100			
	Baseline	Respecting Doc. Boundaries	+SimCLR Loss
Quora	7.922	14.321	19.9
ArguAna	56.33	29.374	92.8
SCIDOCS	2.258	7.583	5.9
Trec-COVID	0.056	0.119	0.1
FiQA-2018	0.949	5.844	8.4
NFCorpus	9.207	8.72	14.2
SciFact	51.589	45.144	77.3

Table 1. Summarized results for normalized discounted cumulative gain for top 10 ranked positions and recall scores for top 100 positions.

The metrics we chose to evaluate our model on are normalized discounted cumulative gain and recall. Normalized discounted cumulative gain effective finds the percentage of maximally correct rankings of items in the top k positions. Ranking scores are found by computing the discounted cumulative gain, which in short is the sum relevancy of documents such that relevancy for the top positions diminishes exponentially as we continue to consider lower ranked positions. We calculate this for the top 10 positions. The second metric presented is recall for the top 100 positions. Recall gives the proportion of correctly identified items. Our contributions are that we augmented performance by more correctly addressing the document retrieval problem. This is substantiated by the preceding data, which demonstrates that the updating data representation allowed for improvements on the baseline for 70% and 57% of evaluated test data sets for nDCG@10 and Recall@100, respectively. Additionally, Our second contribution is the updated architecture and training objective. This demonstrated direct improvements against the baseline on all of the benchmark data sets provided in Table 1 for both metrics. This is a result of the amplified gradient signal that is associated with the SimCLR objective. By considering the SimCLR objective in our loss function, the model is allowed to update its weights with respect to loss identified in both keys and queries. However, the MoCo objective, which is the sole loss function considered by the original authors of Contriever, only updates weights based on queries.

Conclusion

We found that replacing Momentum Contrast (MoCo) with SimCLR noticeably boosted the quality of document retrieval on almost all of the datasets tested on for both the nDCG@10 and Recall@100 evaluation metrics. This aligned with our expectations based on the better performance of SimCLR compared to MoCo on other contrastive learning tasks [ref SimCLR paper]. Surprisingly, we found that respecting document boundaries modestly boosted the quality of document retrieval on some of the datasets, but lowered the quality of document retrieval on others for both the nDCG@10 and Recall@100 evaluation metrics. A possible explanation for this result is that our boundary respecting approach truncated our examples

when it sampled across a boundary to enforce against document overlap in training samples. The inconsistent sizes of data samples may have resulted in a worse performance.