

Exa Interview: Improving embedding performance for complex search

Intro

Exa's goal is to harness the capabilities of LLM's and apply them to search. As you know, it's clearly infeasible (for now!) to have an LLM attend to the entire internet. Your solution is to perform filter results down with an efficient nearest-neighbor search over pre-computed embeddings, and then process and re-rank them. This works reasonably well, but is much less effective at capturing the subtleties of complex queries than an LLM would be. Consider this motivating example:



```
query = "youngest company based in the same region as Apple"

company_1 = "Founded in 2024 in Los Altos, Simile systems produces gpus for the blockchain"
company_2 = "In 2020, Synechdoche was founded by the youngest ex-Apple engineers, based out of the New York"
company_3 = "Analogy.ai was not founded in the same region as Apple and is not the youngest company"

q_embed = model.encode(query)

print(f"Company 1 query similarity: {q_embed @ model.encode(company_1).T}")
print(f"Company 2 query similarity: {q_embed @ model.encode(company_2).T}")
print(f"Company 3 query similarity: {q_embed @ model.encode(company_3).T}")
```

✓ 0.0s Python

```
Company 1 query similarity: [[0.5484736]]
Company 2 query similarity: [[0.73154294]]
Company 3 query similarity: [[0.7733072]]
```

If we're looking the youngest company based in the same region as Apple, the first company is clearly the one that best satisfies the criteria. It was founded the latest of the companies, and is based in the bay area. Yet it ranks last in query similarity, and the highest ranked company was explicitly said to *not* satisfy any of the query. I'm not aware of any name for this (common) problem in the literature, but I'll refer to it as **semantic similarity without relevance**.

This may be rather obvious, but is useful to establish – embeddings (at a high level) simply pool the similarity of individual words in the sentences. Modern transformer based embeddings can take more word context into account, but still currently fail at that task.

Of course, even GPT3.5 can solve this easily:



You

Which of these two companies most satisfies the query: youngest company based in the same region as Apple:

1. Founded in 2024 in Los Altos, Simile systems produces gpus for the blockchain
2. In 2020, Synechdoche was founded by the youngest ex-Apple engineers, based out of the Big Apple



ChatGPT

Based on the information provided, the youngest company based in the same region as Apple would be Simile Systems, founded in 2024 in Los Altos. While Synechdoche was founded by ex-Apple engineers, it was founded in 2020, making it older than Simile Systems. Therefore, Simile Systems satisfies the query as the youngest company.



Will and I discussed his thoughts on how Exa can solve this. There are many approaches, but the simplest one is simply to fine tune an embedding model

This Work: Synthetic Hard Query Mining

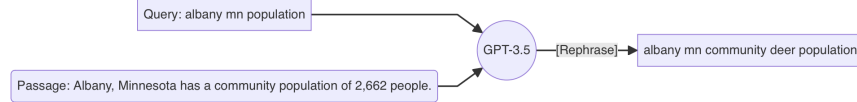
From a theoretical perspective, it's unlikely that a dense embedding will be able to perfectly learn the task of retrieving passages when they are semantically similar but not relevant to the query. Pooling sentence embeddings via [CLS] token rather than merely averaging the word embeddings give the model a little more flexibility, but even then its capacity is limited. However, it's possible that we can fine tune embeddings to approximately solve this use case. Regardless, an answer in the negative would be useful information. In particular, training embeddings with a contrastive loss function *requires* hard examples for good performance – and these make for very hard examples. This motivates my work in the past few days: **can we train embeddings to downrank semantically similar but irrelevant passages, and does this improve performance in general retrieval tasks?**

Approach To my knowledge, there is no natural dataset of similar-but-irrelevant retrieval passages. We could attempt to mine hard examples by taking a dataset of (query, related_passage) pairs and for each query, finding an unrelated passage that is maximally similar to the query. This is both over and under-inclusive – it is unlikely that there are many passages outside of the related_passage that are embedded close to a given query, and even if we do find one, it may in fact **be** a relevant answer to the query.

To get a large enough dataset of guaranteed semantically similar but irrelevant (passage, query) pairs, we can use the fact that LLM's are much better at evaluating semantic similarity than embedding models. In essence, we can rephrase this challenge as a knowledge distillation one – how can we distill

GPT’s semantic similarity capabilities into a smaller, more efficient embedding model.

In particular, we sample (query, passage) pairs from the MS-Marco dataset, and ask GPT-3.5 to make a minimal change to the query such that it looks similar to the old query, but the passage no longer contains the answer to the query.



Generating valid (query, passage, similar_but_irrelevant_query) pairs this way was challenging. There is no automated way to verify the new query is in fact semantically irrelevant, beyond feeding it back into the LLM to check its work. There are many ways even GPT-4 failed to generated actually irrelevant queries – below is a taxonomy of failed examples

Query	Passage	Failed Rephrasing	Reason
albany mn population	Albany, Minnesota has a community population of 2,662 people	albany mn not population	Semantically invalid negation
how long to recover from wisdom teeth removal	According to the offices of practicing oral surgeon Dr. Joseph Arzadon of Arlington, Virginia, typical wisdom teeth recovery time is three to four days	How short is the recovery time from wisdom teeth removal	Opposite query is still answered by the passage
how many carbohydrates in asparagus	Asparagus is a low-calorie, low carbohydrate, and high fiber food choice. One-half cup contains only 20 calories and 3.7 grams carbohydrate.	How many calories are there in asparagus	Rephrasing chose a different question also answered by passage

Query	Passage	Failed Rephrasing	Reason
How many terms did William Bradford serve as governor of Plymouth Colony?	William Bradford (c.1590 – 1657) was an English Separatist leader in Leiden, Holland and in Plymouth Colony was a signatory to the Mayflower Compact. He served as Plymouth Colony Governor five times covering about thirty years between 1621 and 1657.	How many years did William Bradford serve as governor of Plymouth Colony excluding his first term	Rephrased query is less relevant to passage, but still substantially relevant

GPT-4 had fewer semantic errors, and generated slightly more plausible rephrasing, but surprisingly suffered from many of the same problems as GPT-3.5. Due to the $\sim 10\times$ greater cost/token of GPT-4, we used GPT-3.5.

We experimented with many different prompts, prompt formats, and examples to generate the highest quality rephrasing. Because of the lack of automated evals, we couldn’t use something like DSPy to optimize the prompt (or the model), so tuning was done by hand. The final prompt used was:

Here is a text query, and a passage that contains the answer to the query. Your goal is to produce 5 queries that *looks* similar to the original query, but evaluated in formal logic is asking a very different question – one that is not answered by the passage. Make it so the information needed to answer the new query is not present in the passage.

One approach is to add a modifier to the query, one that the passage does not answer. For example, if the original query is “What is the average temperature in the Sahara desert?”, you could add a negation to the query, like “What is the average temperature in the Sahara desert, when it is not summer?”.

Make sure that the new query sounds plausible, and asks a question that someone might actually ask.

Ensure that if you negate the query, the passage does not implicitly answer the negated query. For example, if the original query is

“what is a prime number”, and you negate it to “what is not a prime number”, the passage still implicitly answers the negated query.

Output a numbered of 10 queries, with no prefix. Do not say “the new query is” or anything like that, or include the original query or passage or reasoning in the response.

For example: 1. What is the average temperature in the Sahara desert, when it is not summer? 2. What is the average temperature in the Gobi desert 3. When is it hottest in the Sahara desert? 4. What is the average ground temperature in the Sahara desert? 5. What is the average body temperature in the Sahara desert?

Query: {}

Passage: {}

One worry with this approach is that GPT-3 will learn a small class of negations or modifications, such as prepending “not” before a word, and therefore the embedding model will easily identify modified queries. It may then learn to be less confident for queries that contain “not”, for example, without really learning how to separate hard queries with conditions. To avoid this, we experimented with in addition to generating negative queries, attempting to have GPT-3.5 produce a positive rephrased query, such that the new query contains many of the same negations and qualifiers that a negative query might, but importantly is still semantically answered by the passage – in effect, generating specific classes of failed rephrasings of negative queries. For example, “albany nm population” might be rephrased as “albany nm estimated population”.

Unfortunately, GPT-3.5 (and GPT-4), failed to consistently produce semantically equivalent rephrasings, and so we jettisoned this approach.

We also experimented with using Mistral-7b and Mixtral-8x7b due to its cheap inference costs, but surprisingly found the quality even worse than GPT-3.5. Perhaps this could be ameliorated with better prompts.

In the end, we generated ~**50,000** examples, containing 5 examples of semantically irrelevant rephrasings each. This cost approximately \$4.5 on the Azure endpoint – suggesting that given more time, it would be cost feasible to generate the data with GPT-4.

We also experimented with having GPT-3 generate rephrased *passages*, rather than queries, but found GPT-3 and 4 failed to accurately modify longer passages to be similar-but-not-relevant.

Data Quality I know of no good ground truth model for evaluating relevance (not similarity) of a query to a passage. In lieu of that, I manually evaluated a sample of 60 generated queries, and classified them as relevant (bad) or irrelevant (good). Note that this was not checking that they remained similar to the passage, just relevant. Of them, I estimated 38 were good (irrelevant), and

22 bad. For a population of 50,000, we can estimate the true fraction of good datapoints to be $0.63 \pm .104$.

While this is not of great quality, note that many of the bad (relevant) generated queries are likely still *less* relevant (and less similar) than the original queries, and thus the model will not learn erroneous similarity rankings. In fact the bad examples are likely *harder*, and thus better training data, even if they won't directly teach the model to distinguish similar-but-relevant passages.

Model

We used the Baidu AI's pretrained embedding model, BGE. It is the standard transformer based embedding model, where the hidden state of the "[CLS]" token is the embedding for a sentence. It has competitive performance on many reranking benchmarks, especially MTEB, and comes in multiple sizes for scaling studies.

Training

We trained the model with a variant of the standard TripletLoss. Typically, contrastive embedding is trained with a single query, and multiple possible documents, but due to our data generation we had multiple **queries**, and a single document. Because the loss is symmetric wrt to query and documents, this was an equivalent objective. For a sample i , let q_i^+ be the correct query, q_i^- the incorrect query, and p_i the passage. Let λ be the margin, and $d(x)$ denote the cosine distance in embedding space. Then the loss is

$$\ell_i = \max(d(q_i^+, p_i) - d(q_i^-, p_i) + \lambda, 0)$$

Because each sample actually contains multiple negative queries $Q_i^- = [q_{i1}^-, \dots, q_{ij}^-]$, typically we let take the min distance over all negative embeddings

$$d(p_i, Q_i^-) = \max_j d(p_i, q_{ij}^-)$$

However, I found taking the hard-max over the queries led to training difficulties, so to weaken the difficulty we take the mean distance over the top_k distances in the set.

There wasn't time for a proper hyperparameter search, but the hyperparameters that led to the best results were a lr=4e-5 with cosine decay, 10 training epochs, and a margin of 10. It was trained on a single A10G GPU for ~8 hours. The largest batch size we could do was 32, with fp16 mixed precision training.

Results

In the brief time, we tested a few metrics. To evaluate performance on the specific tasks of separating similar-but-irrelevant queries, we evaluated a few metrics over a held out set of size 2660:

- average similarity between passages and the positive query (AVG_SIM_POS) (higher is better),
- average similarity between passages and the synthetic negative queries (AVG_SIM_NEGATIVE) (lower is better),
- fraction of passages closer to negative queries than positive queries (FRAC_INCORRECT) (lower is better)
- fraction of passages closer to randomly drawn queries than positive queries (FRAC_INCORRECT_RANDOM) (lower is better)
- fraction of passages less than margin farther from positive queries vs negative queries (FRAC_MARGIN_ERRORS)

Metric	Old model (bge-large-v3)	Fine-tuned model
AVG_SIM_POS	0.8306	0.8780
AVG_SIM_NEGATIVE (lower is better)	0.7815	0.3364
FRAC_INCORRECT	0.2493	0.0004
FRAC_INCORRECT_RANDOM	0.2150	0.0019
FRAC_MARGIN_ERRORS (lower is better)	0.7540	0.0026

In addition, we tested the standard MRR@10 on the ms-marco dataset

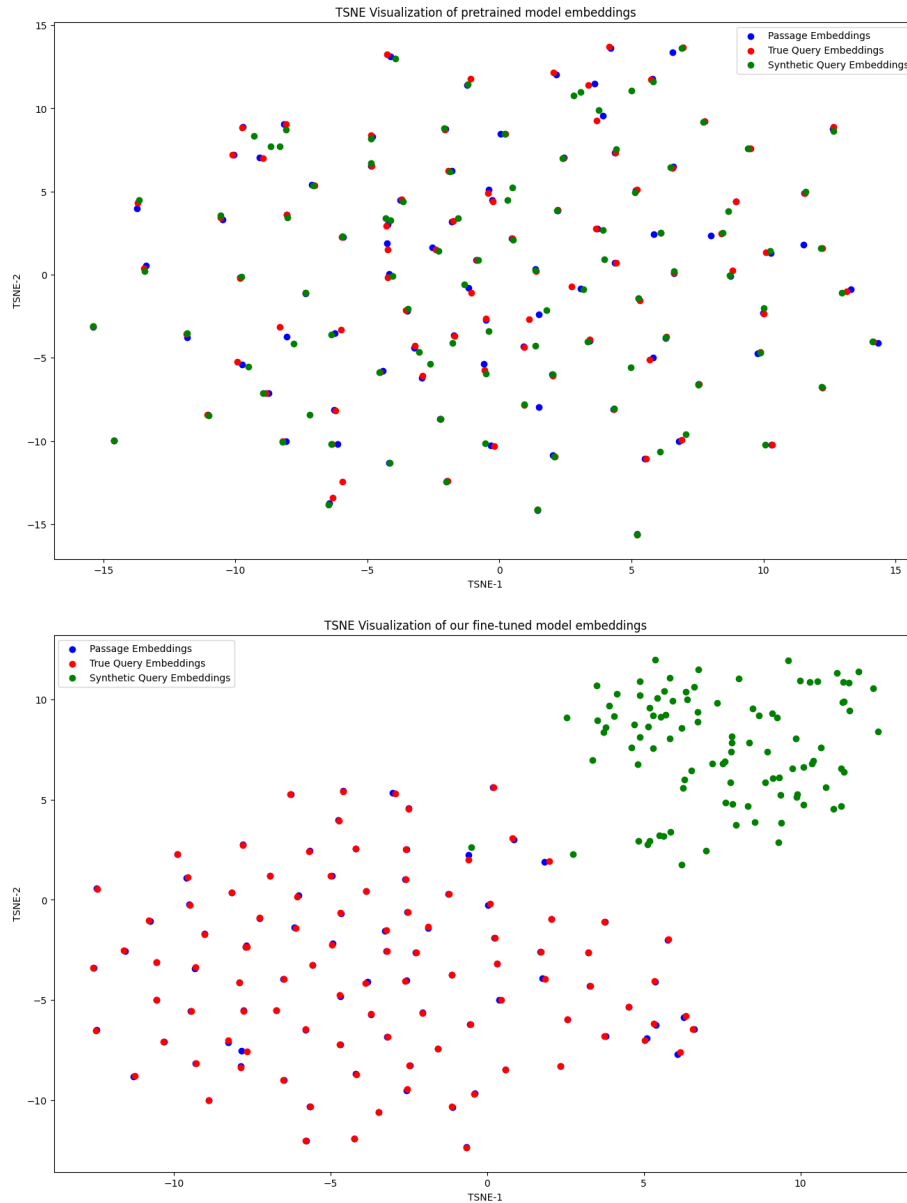
Metric (higher is better)	Old model (bge-large-v3)	Fine-tuned model
MRR@100 (Dev)	0.442	0.436
MRR@100 (Eval)	0.393	0.385

In conclusion, we were able to get dramatically better results at the specific task of distinguishing similar-but-irrelevant from relevant queries. We strictly improved on every metric over the original model, on entirely held out data.

However, the improvement in this task did not translate to improvements on the MS MARCO task, and even lead to small but statistically significant performance decreases. We hypothesize that this is because there simply aren’t many natural similar-but-irrelevant queries in the MS-Marco dataset, and thus the improvement on this task does not provide any benefit, and the decrease may indicate overfitting on the particular fine-tuning task.

In particular, the decrease in the number of passages closer to randomly drawn queries than positive queries (FRAC_INCORRECT_RANDOM) in the first table indicates some overfitting. The majority of randomly drawn queries from the dataset will be synthetic ones, and the fact that that improved while standard MS-Marco metrics did not indicates that the model may be learning to push “synthetic-sounding” queries.

Embedding Visualization We also can visualize the high-dimensional embedding space in 2d using TSNE. We take each model embeddings of the true queries, the synthetic negative queries, and the passages, cluster them with TNSE, and plot them.



In this, we see results indicating learning – before training, all three embeddings are roughly clustered together. In the second plot after training, the negative

synthetic queries are pushed much farther away from the still clustered passage and true queries – exactly the result we’d expect!

Conclusion and future work

In the past three days, I generated a dataset of hard similarity-without-relevance example data. I wrote the code to train and evaluate embedding models via contrastive learning. After experimenting with model size and quality, I managed to train a model that surprisingly successfully learned to distinguish relevant from similar-but-not-relevant queries, and visualized the results to try and evaluate the takeaways. Unfortunately, this training run in particular did not improve performance on MS-Marco dataset, for a couple of possible reasons

- There aren’t many similar-but-not-relevant queries in the dataset, and this sort of similarity matching isn’t how Bing users or evaluators in the dataset think about the problem. This would make the two tasks somewhat orthogonal
- It may be that the model is simply undertrained – it successfully learned to separate the held out dataset, but 10,000 examples isn’t particularly large, and especially a larger dataset of harder examples would more improve performance
- The model may have overfit on the particular phrasing of the GPT-3 query rephrasing, without improving the general capability.

I suspect to some extent the last one is true – it seems unlikely that a nearest neighbor classifier could approximate the highly non-convex shape of boolean classifiers. Still, there remains many possible areas of future exploration, both for this approach and for the problem more broadly.

- Use GPT-3 as a dataset curator – either by filtering synthetic queries, or by reranking queries in order of hardness on the similar-but-not-relevant task
- Perform more scaling experiments – see how these capabilities change with the size of the embedding model, or the size of the embedding themselves.
- Obviously, more ablation studies and hyperparameter tuning and more metrics – all things I would do with more time/GPUs. In particular, MS Marco has been noted as having numerous deficiencies as a dataset, so exploring other ground truth ones.
- Mix in MS Marco and synthetic data together in training to try to maintain general performance while improving task specific performance.
- Alternatively, experiment with model merging in order to add new skills to the base model without forgetting
- Generate more logically rigorous and grounded data – start with an database with many boolean or integer or categorical columns. Then randomly sample a possible column assignment/query, and have GPT-3 generate a text description of each database row. This allows us to exactly evaluate relevant rows – simply which ones satisfy the boolean query – at

the cost of less natural queries and passages.

- Use a GAN set up to mine hard queries

However, I remain convinced that nearest-neighbor embeddings alone are not the way forward here. There are numerous possible areas of exploration, from the tested (Colbert-style late interaction) to the currently ludicrous – end-to-end neural retrieval models. All are ones I’m excited to explore!