**The Untold Side of RAG: Addressing Its Challenges in Domain-Specific Searches**

https://towardsdatascience.com/the-untold-side-of-rag-addressing-its-challenges-in-domain-specific-searches-808956e3ecc8

Using hybrid search, hierarchical ranking and instructor embedding to address domain-specific documents that bear similarities for our RAG setup.

[[](https://agustinus-nalwan.medium.com/?source=post_page-----808956e3ecc8--------------------------------)](https://agustinus-nalwan.medium.com/?source=post_page-----808956e3ecc8--------------------------------)

[[Towards Data Science](https://towardsdatascience.com/?source=post_page-----808956e3ecc8--------------------------------)](https://towardsdatascience.com/?source=post_page-----808956e3ecc8--------------------------------)

[Agustinus Nalwan](https://agustinus-nalwan.medium.com/?source=post_page-----808956e3ecc8--------------------------------)

·

Follow

Published in

[Towards Data Science](https://towardsdatascience.com/?source=post_page-----808956e3ecc8--------------------------------)

·

29 min read

·

Oct 18, 2023

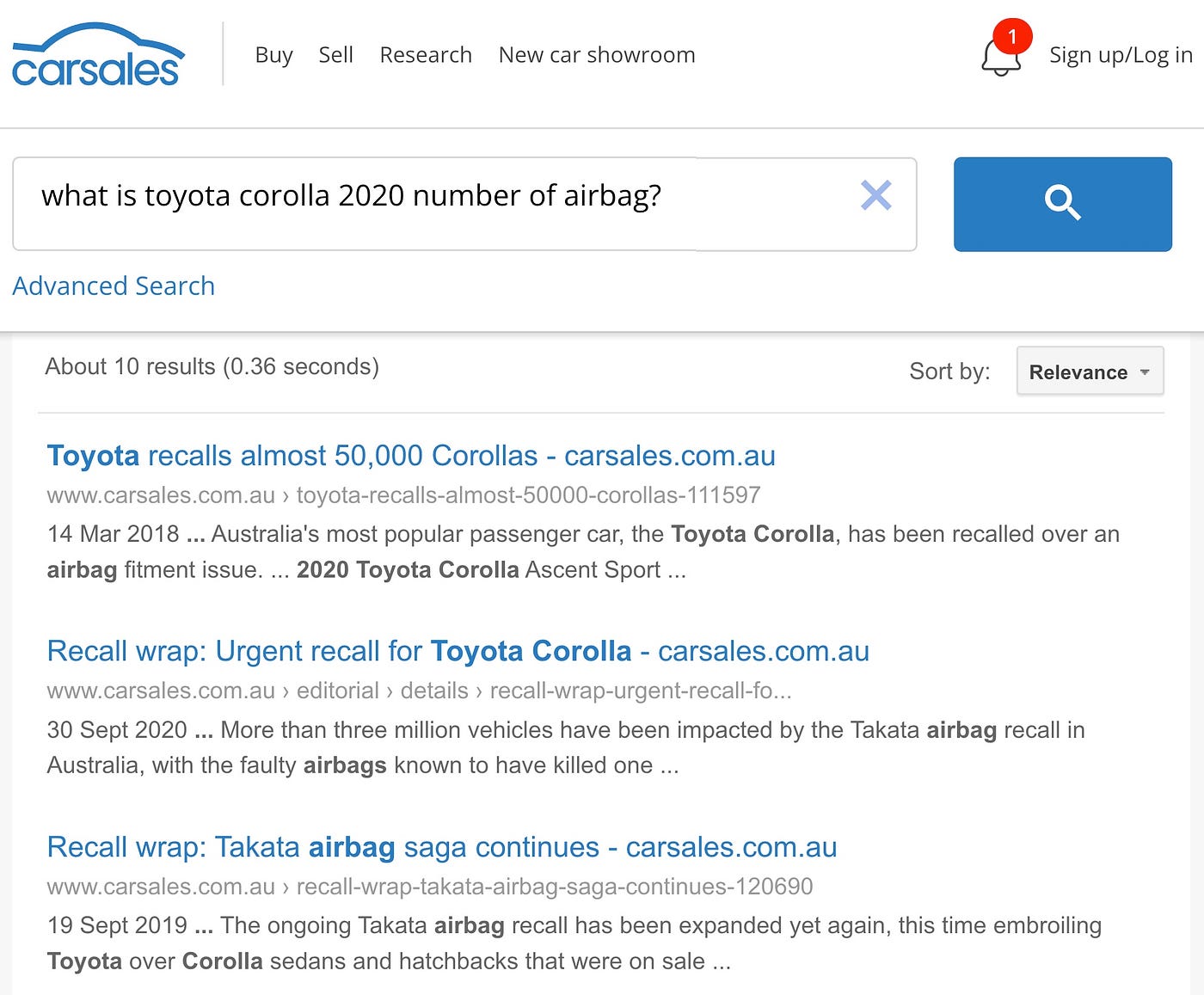
739

15



Generative AI augmented search technology (image generated with MidJourney)

[Carsales](https://www.carsales.com.au/) stands as a leading automotive platform, serving the car and lifestyle vehicle markets across Australia, Chile, South Korea, and the United States. Our ambition is to redefine the car buying and selling experience, setting unparalleled standards. To this end, one of our pivotal features is a comprehensive search tool that scans through tens of thousands of car-related editorial pieces. We currently integrated Google Search — tailored exclusively for our editorial content and presented through an iframe — the results, although decent, primarily relied on lexical (keyword) associations, sometimes missing the true essence or semantic/meaning behind a search query.



Search result using existing Google search

For instance, a search for “What is Toyota Corolla 2020 number of airbag?” would yield results for any article with words “Toyota Corolla and airbag”, etc. Yet, these articles mostly discuss airbag recalls rather than the actual number of airbags. There’s a pronounced demand from the business side to not only enhance this tool technologically but also redesign its interface to make it more seamlessly integrated into our website, moving beyond a mere Google search result in an iframe.

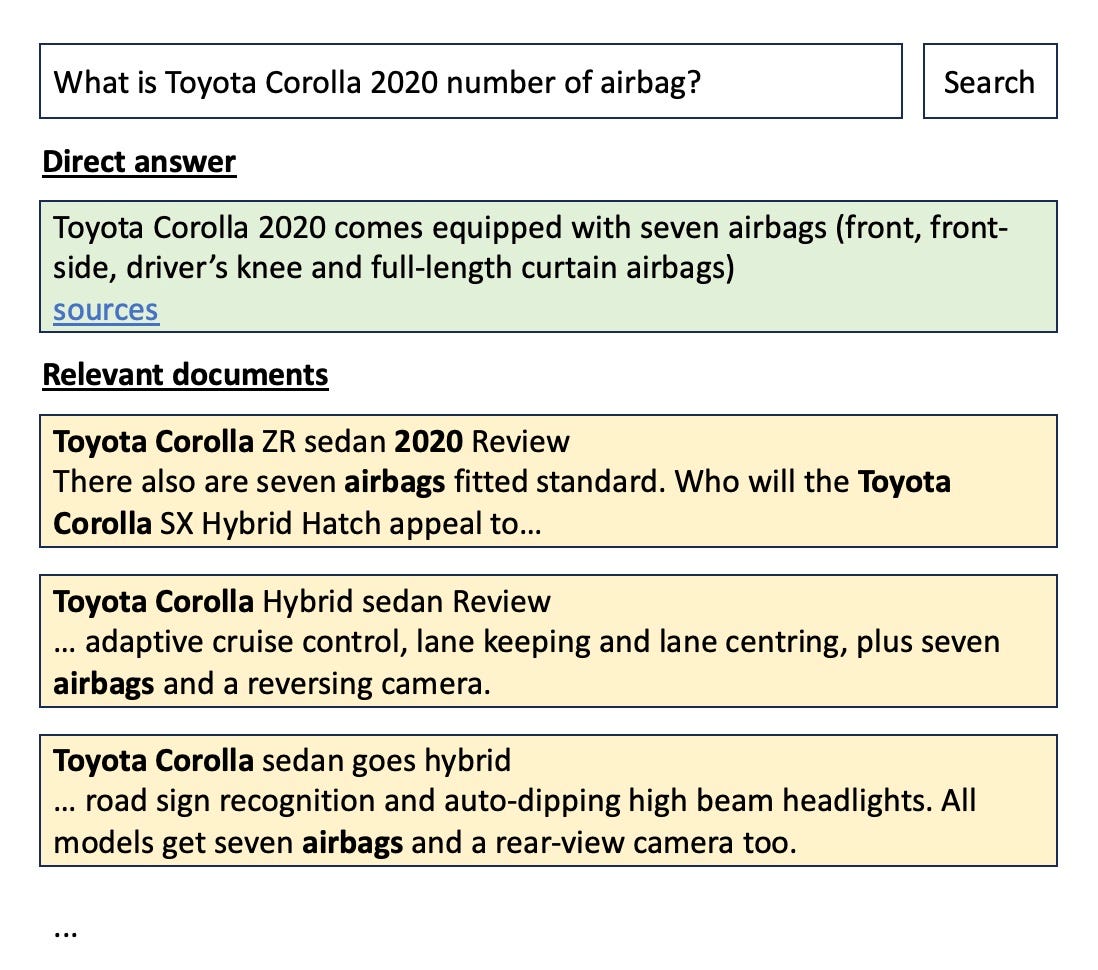
In July 2023, we held one of our bi-annual global hackathons. This event typically sees participation from various departments where teams are formed, ideas pitched, and within a tight 3-day window, they’re transformed into working prototypes. With the rising prominence of Large Language Models (LLMs), we identified an ideal project for this hackathon: revamping our search tool using a LLM. To top that, we won the hackathon! Credit to our awesome team who worked super hard on this!

In this article, we’ll kick off by outlining the fundamental concept of our project. After that, we’ll touch upon our preliminary RAG approach. Subsequently, we’ll highlight the challenges faced in ensuring accurate and relevant document retrieval and generating direct answers. We’ll then discuss the solutions we implemented to tackle the document retrieval dilemma. However, we’ll save the direct answer generation discussion for a subsequent article to keep this one succinct.

**Concept**

Our envisioned model is straightforward. We aim for a two-fold output upon receiving a search query:

* Relevant Documents: While this mirrors our existing features, our objective is to enhance its efficacy by retrieving documents that resonate with the query’s intent (meaning-based matching) rather than merely on a lexical basis (direct word matching).
* Direct Answer: Whenever the query is posed as a question and our system can derive an answer from our editorial resources, we want to present that direct answer. This novel feature promises to significantly enrich user interaction.



Search result using our conceptualised RAG approach

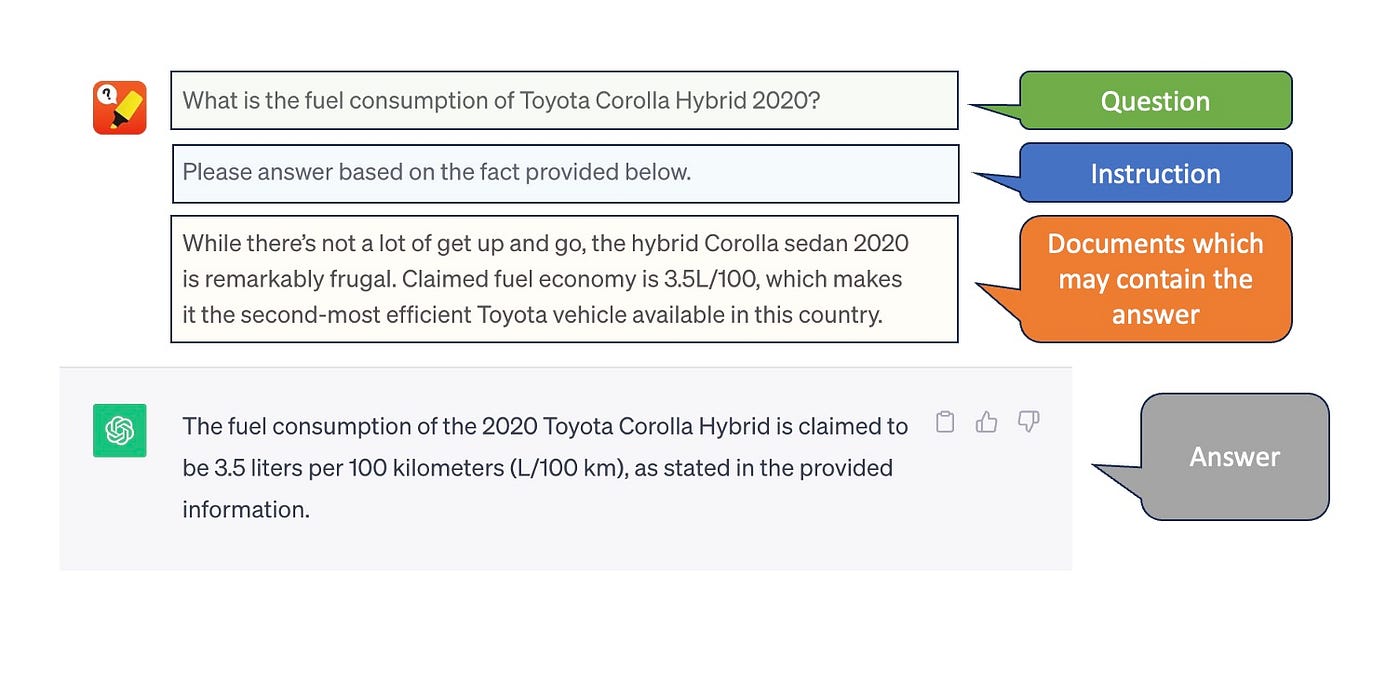
When leveraging LLMs for document retrieval plus providing direct answer using an organisation’s knowledge base, there are typically two methodologies:

* Fine-Tuning an LLM: This involves refining a foundational LLM using your organisation unique knowledge reservoir — in our scenario, our assortment of editorial articles. This ensures the LLM can aptly respond to queries related to these articles.
* Retrieval Augmented Generation (RAG): Here, the strategy is to provide the LLM with contextually relevant editorial articles alongside the posed question, enabling it to craft an accurate response.

Fine-tuning directly incorporates knowledge into the model, reducing reliance on extensive prompts with the LLM. However, this comes with challenges: the constant need to update the LLM with new or revised articles, and the significant increase in operational costs to run the fine-tuned model — potentially ten times more than a foundation LLM. Given these, most, including us, prefer the RAG approach.

**Retrieval Augmented Generation**

Retrieval Augmented Generation (RAG) might sound complex, but its principle is straightforward. Imagine breaking our query into three segments: Firstly, the question itself, like “What is the fuel consumption of Toyota Corolla Hybrid 2020?” Secondly, an instruction to the LLM to construct a response using the documents we supply next. Lastly, the document context where the answer might be found.



RAG in a nutshell

At its core, that’s RAG for you. Naturally, you might wonder how we sift through tens of thousands of articles in our repository to pinpoint the relevant ones as a document context to answer the posed query.

Why not include all our articles in the document context? Consider 45,000 articles, each with 10,000 words totalling 450 million words. The latest LLM, Claude 2, only supports a 100,000-token window. Even if future models expand capacity, the associated costs and processing times would be prohibitive. Our aim is to rank articles by their relevance to a query. The more similar an article’s content to the query, the more likely it contains the answer. So, we select the top X articles as our document context. This is why we need a vector database.

**Vector Database and Dense Embedding**

A vector database essentially stores vectors, which are arrays of floating-point numbers. But how does this address our challenge? Enter dense embedding, an NLP technique that transforms text (our document) into a vector form. This vector typically encapsulates the text’s semantic essence more effectively, allowing a more accurate comparison of the similarity between two pieces of text.

Before the advent of dense embedding, we gauged document similarity by tallying common words between them. Let’s illustrate this with an example. Given the question, “How many airbags does Toyota Corolla 2020 have?”, which of the following sentences aligns more closely with it?

***Toyota Corolla 2020******have****been recalled due to****airbag****filament among****many****other issues.*

***Toyota Corolla 2020****features seven****airbags****and a backup camera.*

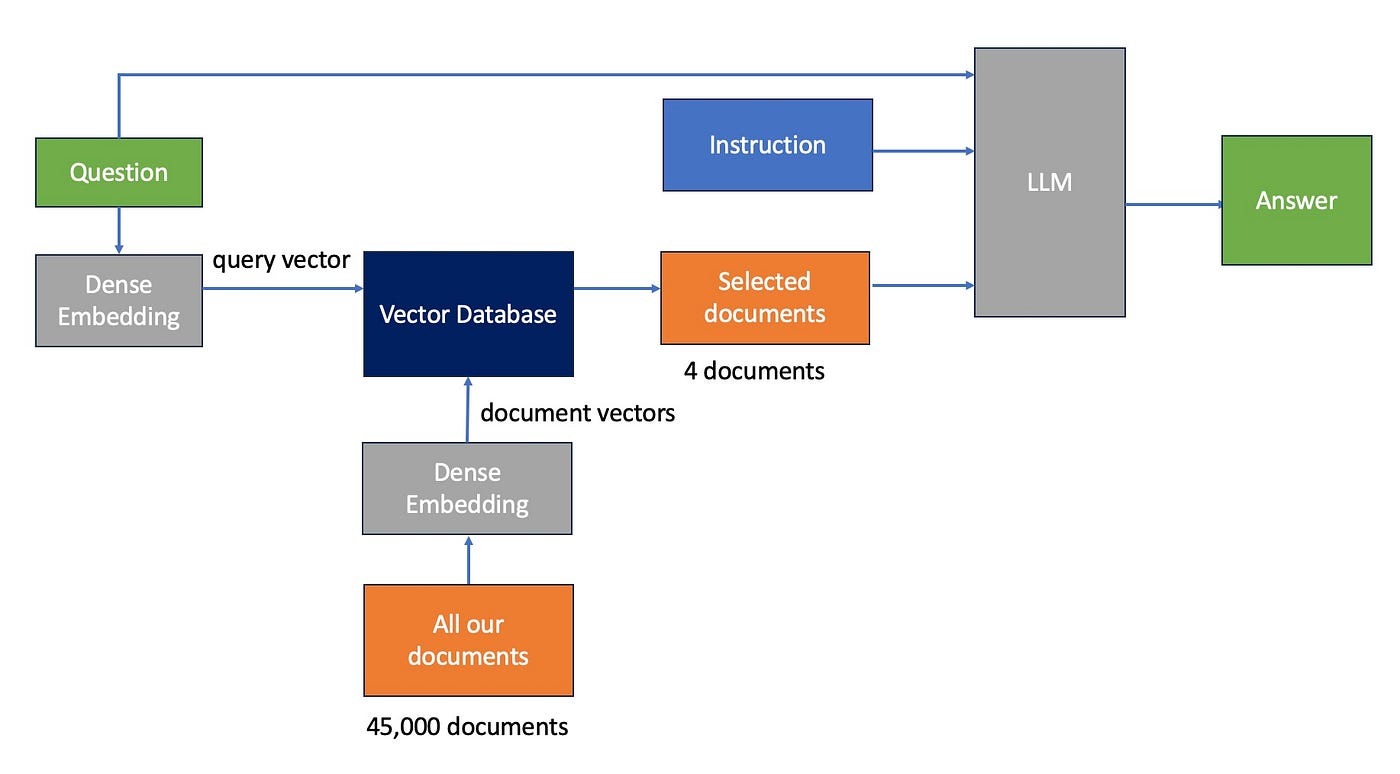
Using a simplistic word match, the first sentence, with six common terms, seems more akin to the question. But in essence, the second, despite having only four matched terms, is more pertinent since it addresses the query about the airbag count. Even by excluding the stop-words (have and many) they both have equally four matching terms.

When we convert the question and both sentences into vectors using the dense embedding technique and visualise them onto a two-dimensional plane, the proximity of the second sentence to the question becomes evident, especially when gauging similarity based on the cosine distance (angle between the vectors).

Leveraging this capability, we can rank all our documents in relation to a given question, choosing only the top four most similar articles to feed as context to our LLM, instead of sifting through the entirety of 45,000 documents. The vector size can vary based on the dense embedding method employed. Initially, we employed the text-embedding-ada-002 from OpenAI, which yielded a vector size of 1536.

Here’s a breakdown of the process:

Initially, during the indexing phase, we convert all our articles into vector representations using the dense embedding method and save these vectors in a vector database like Pinecone. When it’s time for inference, we similarly convert the posed question into a query vector with the same embedding technique. This query vector is then input into Pinecone, which retrieves the top four most relevant articles. Pinecone seamlessly computes the cosine distance between the query vector and all our stored document vectors to return the top 4 relevant documents. Following this, we introduce the content of these selected articles to the LLM as document context, along with the original question and an instruction to formulate an answer.



RAG architecture

Essentially, this captures the essence of Retrieval Augmented Generation. Most RAG implementations include certain refinements to enhance the relevance of the chosen documents, which subsequently influences the precision of the resulting answer which I will cover next.

**Chunking our documents**

Typical editorial articles might extend to up to 10,000 words. Sending the top four documents for context could mean processing up to 40,000 words, a potentially resource-intensive task, especially with frequent calls. For instance, an editorial review of the Toyota Corolla 2020 might encompass various aspects, from safety features and pricing to fuel efficiency, distributed across different paragraphs. More often than not, answers to specific queries can be pinpointed to just a couple of these paragraphs. This negates the necessity of transmitting entire articles for processing.

The practical implication? Instead of converting entire documents into vectors, it’s more efficient to segment them into smaller chunks (typically one or two paragraphs) and transform these into dense vectors — a process known as “document chunking.”

Interestingly, document chunking doesn’t just economise our processing; it also amplifies the accuracy of document retrieval. How, you ask?

Take, for example, a snippet from a Toyota Corolla Hybrid Sedan review:

*The new Toyota Corolla ticks a lot of boxes. For the record, the four door is 255mm longer overall than its five-door hatch sibling, measuring 4630mm long.*

*Inside it’s got loads of useful features, like Apple CarPlay and Android Auto across the range, displayed on a decent 8-inch touchscreen display. Adaptive radar cruise control is also standard on all models.*

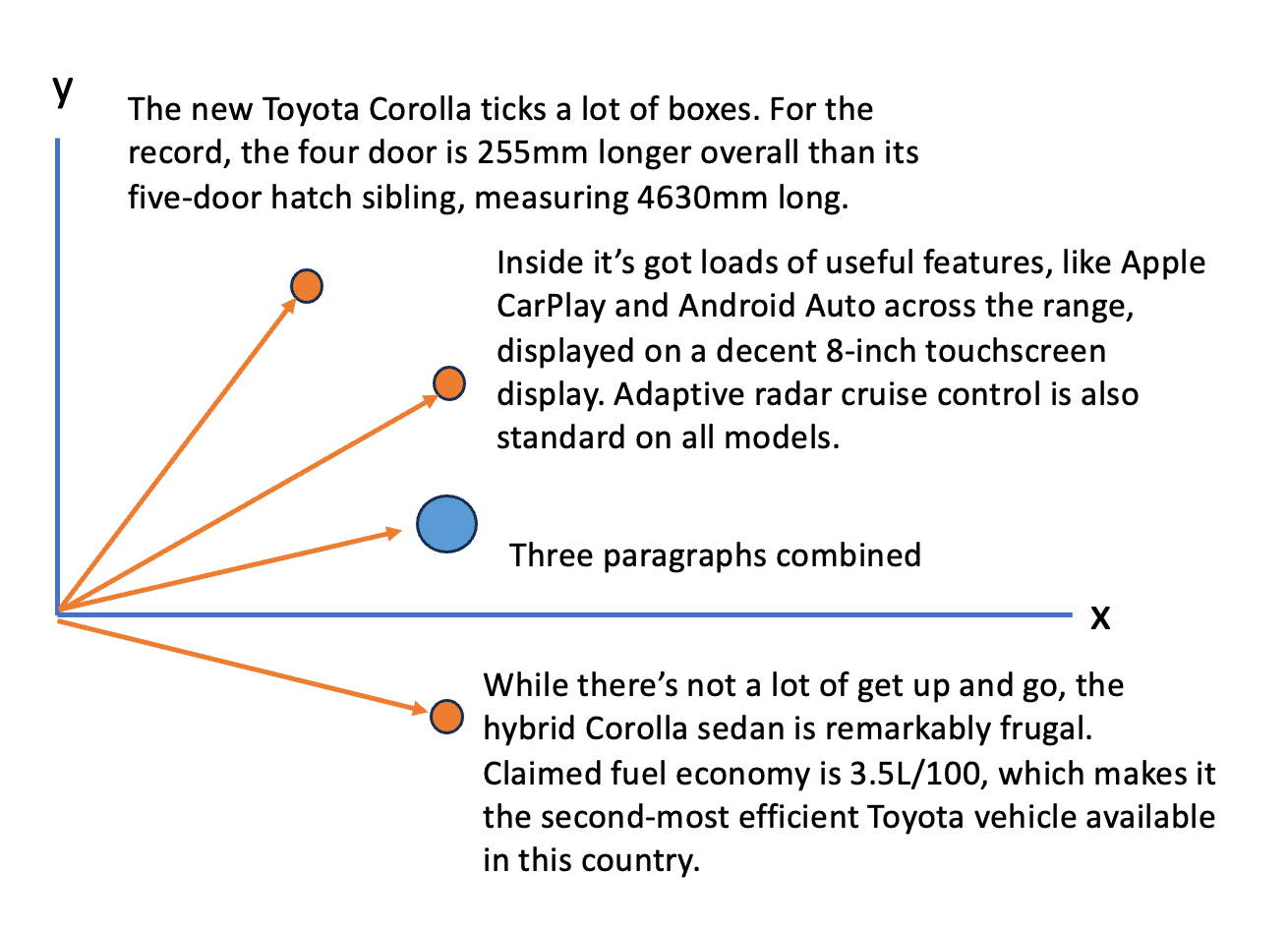
*While there’s not a lot of get up and go, the hybrid Corolla sedan is remarkably frugal. Claimed fuel economy is 3.5L/100, which makes it the second-most efficient Toyota vehicle available in this country.*

As observed, the first paragraph discusses the car’s dimensions, the second elaborates on digital device features, and the third zeroes in on fuel efficiency.

If we separate each of these paragraphs into distinct dense vectors, we generate three unique document vectors situated in different embedding spaces, each symbolising varied topics.

Suppose a query is “What are the dimensions of the Toyota Corolla?”. The first paragraph becomes the most relevant document. For a question like “Is CarPlay available?”, the second paragraph takes precedence.

Yet, combining all three paragraphs into a singular dense vector results in a more generalised vector, which may not pinpoint a particular topic as effectively. The larger the embedded document, the more diluted the vector becomes, eventually failing to accurately capture specific subjects. Over time, as dense vectors incorporate numerous overlapping topics, like safety or interior features, their distinctiveness diminishes, reducing the precision of document retrieval.



Dense embedding lost its specificity the larger the document chunk gets

Given these factors, we partition our documents into chunks, with each chunk comprising 300 tokens. Additionally, we employ the RecursiveCharacterTextSplitter technique from the Langchain framework.

This approach categorises texts hierarchically: prioritising paragraphs first, followed by sentences, and then individual words. Our objective is to preserve entire paragraphs within the confines of the 300-token boundary, if feasible. If not, we then break them down to sentences and, subsequently, words. Separating a paragraph into separate chunks might lead to missing crucial context related to the topic at hand.

To bolster the continuity of context within each segment, we ensure that subsequent segments overlap with their predecessors by 100 tokens. This strategy ensures that when a lengthy paragraph is divided into multiple segments, the shared words between these segments help reduce any potential loss of context.

As an example, consider the paragraph below.

*Elsewhere, the long-awaited Model S refresh borrows interior elements from the smaller Tesla Model 3 and forthcoming Tesla Model Y, including an updated 17.0-inch centre touch-screen display bearing satellite navigation, internet functionality and no shortage of games, video streaming and Tesla’s much-vaunted ‘Easter eggs’.*

When the paragraph is divided into two separate sections without any overlapping content, the subsequent segment loses reference to the specific car make or model being discussed.

*Chunk 1:*

*Elsewhere, the long-awaited Model S refresh borrows interior elements from the smaller Tesla Model 3 and forthcoming Tesla Model Y, including an updated 17.0-inch*

Chunk 2:

*centre touch-screen display bearing satellite navigation, internet functionality and no shortage of games, video streaming.*

With overlapping content (highlighted in bold), the context is now better preserved.

*Chunk 1:*

*Elsewhere, the long-awaited Model S refresh borrows interior elements from the smaller Tesla Model 3 and forthcoming Tesla Model Y, including an updated 17.0-inch*

*Chunk 2:*

***Tesla Model 3 and forthcoming Tesla Model Y, including an updated 17.0-inch****centre touch-screen display bearing satellite navigation, internet functionality and no shortage of games, video streaming.*

**Injecting high level context**

It’s common for paragraphs in our editorial pieces to omit explicit mention of the specific car model being discussed. This is usually because it’s implied from the article’s title or from earlier paragraphs. Let’s take the following second paragraph as an example. While the reader knows from prior context that it’s about the Toyota Corolla 2020 Hybrid sedan, this specific model isn’t explicitly mentioned in the paragraph. Imagine having paragraphs from 20 different articles, each about a distinct car but all mentioning common features like Apple CarPlay and Android Auto. When each paragraph are turned into a separate document chunk, their dense vector embeddings could end up looking quite alike. So, if a user asks, “Does the Toyota Corolla 2020 hybrid sedan feature Apple CarPlay and Android Auto?”, the query vector could be similar to any of those 20 document chunks, creating a retrieval challenge.

*The new Toyota Corolla ticks a lot of boxes. For the record, the four door is 255mm longer overall than its five-door hatch sibling, measuring 4630mm long.*

*Inside it’s got loads of useful features, like Apple CarPlay and Android Auto across the range, displayed on a decent 8-inch touchscreen display. Adaptive radar cruise control is also standard on all models.*

*While there’s not a lot of get up and go, the hybrid Corolla sedan is remarkably frugal. Claimed fuel economy is 3.5L/100, which makes it the second-most efficient Toyota vehicle available in this country.*

To address this challenge, we incorporate the article’s title, which usually states the car’s brand, model, and release year, at the start of each document chunk. This integration ensures that the dense vector embedding reflects this crucial information.

*Title: Toyota Corolla 2020 Hybrid sedan Review*

*The new Toyota Corolla ticks a lot of boxes. For the record, the four door is 255mm longer overall than its five-door hatch sibling, measuring 4630mm long.*

*Title: Toyota Corolla 2020 Hybrid sedan Review*

*Inside it’s got loads of useful features, like Apple CarPlay and Android Auto across the range, displayed on a decent 8-inch touchscreen display. Adaptive radar cruise control is also standard on all models.*

*Title: Toyota Corolla 2020 Hybrid sedan Review*

*While there’s not a lot of get up and go, the hybrid Corolla sedan is remarkably frugal. Claimed fuel economy is 3.5L/100, which makes it the second-most efficient Toyota vehicle available in this country.*

**Embracing one-shot learning**

LLMs excel at tasks they haven’t been directly trained for. Pairing instructions with a sample often yields better results. Performing a task without examples is called zero-shot learning, while with an example is termed one-shot learning. Following common RAG methodologies, we use one-shot learning, providing a sample example.

**Introducing source article citation**

Our next goal is to display the source from which an answer originates. Citing sources enhances user experience by offering a traceable reference, which promotes deeper exploration and verification. Such transparency boosts user trust, allowing them to validate answers, increasing confidence in our platform. Moreover, it curtails “hallucination” issues common in large language models like LLMs, which sometimes provide incorrect answers. By making LLMs reference the origin article, we deter inaccurate responses since they can’t cite a non-existent source. This mirrors human behaviour: people hesitate to answer without a verifiable source. Below, our user prompt, based on the [Langchain](https://www.langchain.com/) template, instructs the LLM to include the source in the SOURCES field.at you don't know. Don't try to make up an always return a "SOURCES" part in your answer.

We injected the actual question in the {question} placeholder and our top 4 document chunks with its source in the {document-context} placeholder.

Given the following extracted parts of a long document (with its source) and a question, create a final answer with references ("SOURCES").   
If you don't know the answer, just say that you don't know. Don't try to make up an answer.  
ALWAYS return a "SOURCES" part in your answer.QUESTION: Which state/country's law governs the interpretation of the contract?=========  
Content: This Agreement is governed by English law and the parties submit to the exclusive jurisdiction of the English courts in relation to any dispute (contractual or non-contractual)...  
Source: 28-plContent: No Waiver. Failure or delay in exercising any right or remedy under this Agreement shall not constitute a waiver of such (or any other) right or remedy.\n\n11.7 Severability.  
Source: 30-plContent: (b) if Google believes, in good faith, that the Distributor has violated or caused Google to violate any Anti-Bribery Laws (as defined in Clause 8.5) or that such a violation is reasonably likely to occur,  
Source: 4-pl=========  
FINAL ANSWER: This Agreement is governed by English law.  
SOURCES: 28-plQUESTION: What did the president say about Michael Jackson?  
=========  
Content: Madam Speaker, Madam Vice President, our First Lady and Second Gentleman. Members of Congress and the Cabinet. Justices of the Supreme Court. My fellow Americans.  
Source: 0-plContent: We have lost so much to COVID-19. Time with one another. And worst of all, so much loss of life.  
Let's use this moment to reset. Let's stop looking at COVID-19 as a partisan dividing line and see it for what it is: A God-awful disease.  
Source: 24-plContent: And a proud Ukrainian people, who have known 30 years of independence, have repeatedly shown that they will not tolerate anyone who tries to take their country backwards.  
To all Americans, I will be honest with you, as I've always promised. A Russian dictator, invading a foreign country, has costs around the world. And I'm taking robust action to make sure the pain of our sanctions is targeted at Russia's economy.  
Source: 5-plContent: More support for patients and families. To get there, I call on Congress to fund ARPA-H, the Advanced Research Projects Agency for Health. It's based on DARPA-the Defense Department project that led to the Internet, GPS, and so much more.  
Source: 34-pl=========  
FINAL ANSWER: The president did not mention Michael Jackson.SOURCES:  
QUESTION: {question}=========  
{document-context}  
=========FINAL ANSWER:

Upon implementing our system, We eagerly put our RAG to the test, anticipating the first interaction with the LLM. Yet, once the initial enthusiasm faded, we discovered certain issues spanning two main aspects: the quality of document retrieval and the accuracy of generated direct answer. This happened to nearly half of our test cases.

To be candid, half of the problems related to the direct answers were rooted in the inadequate quality of the selected documents as context. Nonetheless, this article will center exclusively on the document retrieval challenges and our strategies to address them. Stay tuned for an upcoming article that delves deeper into the complexities of answer generation.

**Problems**

**Article of car with irrelevant year ranked higher**

When we searched the Pinecone database with the query “Mazda CX-9 2018 review”, the system returned three document snippets. Surprisingly, the top two results were from Mazda CX-9 2019 and 2017 Review articles, with the 2018 Review (our actual target) only ranking third.

*(1) Title: 2019 Mazda CX-9: Video review  
The Mazda CX-9 is already a favourite in the Carsales office. It took out our 2016 Car of the Year title and still manages to finish at the top of the pile in most multi-car comparisons. For 2019, the CX-9 has been updated and Mazda says it has done away with the very few niggles bugging the …*

*(2) Title: Mazda CX-9 2017 Review  
CX-9 model, behind only the Azami, the GT is quite a lavish affair inside and out . , Twenty-inch wheels, chrome accenting, keyless entry and LED driving lights imbue our ‘Soul Red’ example with a classy outside appearance …*

*(3) Title: Mazda CX-9 2018 Review — Australia  
There’s a new model within the Mazda CX-9 ranks. The range-topping CX-9 Azami LE arrives in Australia this month bearing more equipment, technology and, importantly, added luxe factor. It headlines a Mazda CX-9 range that has received an ever-so-slight nip and tuck for the 2019 model year …*

For the query “What is Mazda CX-9 2019 fuel efficiency?”, the system failed to provide results specifically about the Mazda CX-9 model year 2019 fuel efficiency. All documents returned are of different year. The first one is for 2010 model. The document that did discuss the Mazda CX-9 2019 fuel efficiency was titled “Mazda CX-8 v Mazda CX-9 2019 Comparison”. This document was surprisingly ranked sixth, meaning it wasn’t provided to the LLM as a document context for generating a direct answer.

*(1) Title: Mazda CX-9 now leaner, lighter, brighter  
Mazda is establishing a tradition of knocking at least 800cc of the combined-cycle fuel consumption for its CX-9 model with each passing year. On its release three years ago, the CX-9 was rated at 13.0L/100km. An upgrade last year reduced that figure to 12.2L/100km and another upgrade announced yesterday has officially lowered the consumption again — to 11.3L/100km …*

*(2) Title: Mazda CX-9 2016 Review  
170kW, 2.5-litre four-cylinder turbocharged-petrol engine paired with a six-speed automatic transmission. There’s no manual option or diesel powertrain available, and you can thank the heavy focus on developing this vehicle for the US market for this lack of fuel choice. That said, the petrol engine is still capable of achieving impressive fuel economy figures for a vehicle of its size, promising a combined cycle fuel consumption figure of 8.8L/100km …*

*(3) Title: Mazda CX-9 2018 Review  
Fuel use is a constant consideration, too, we’ve piled on thousands of kilometres, mostly with two people on board, and the reading still hovers around 10.0L/100km. Note: this latest consumption figure is marginally up on the two-wheel drive version of the CX-9, which clocked an average of 9.2L/100km according to both the trip computer and our own independent testing. Both are well up on their claims …*

*…*

*(6) Title: Mazda CX-8 v Mazda CX-9 2019 Comparison  
The Mazda CX-9 is a very highly rated vehicle, having won carsales.com.au car of the year and several other awards as well. But it only comes with a 170kW/420Nm 2.5-litre four-cylinder turbo-petrol engine mated with a six-speed automatic transmission. Add in all-wheel drive — as we naturally did for this test — and you’re looking at a hefty 1924kg kerb weight. But Mazda still claims a reasonable 8.8L/100km fuel consumption figure. We’ll put that to the test over highways, byways and dirt tracks …*

**Document retrieval did not return article that prioritise article recency**

Further examination revealed that our document retrieval system overlooks the recency of articles, often neglecting to prioritise more recent content.  
For instance, a search for “Toyota Corolla” produces results with articles from 2010 and 2013 as the top hits. Surprisingly, a more relevant article about the Toyota Corolla 2020, which should be of higher interest due to its recency, is positioned at a distant 15th place.

*(1) Title: Toyota unveils next Corolla sedan  
extra wheel-house noise deadening. Standard US-market features will include eight airbags, Bluetooth, air-conditioning, an in-glass radio aerial, a 60/40-split rear seatback and LED low-beam lights and daytime running lamps — which Toyota says is a first in the small-car segment …*

*(2) Title: Toyota Corolla: Small car mega-test 2013  
Toyota Corolla Ascent Sport (Hatch) What we liked: Improved materials quality Sharp performance Impressive value Not so much: Flat dashboard design Fiddly touchscreen No rear seat vents ON THE ROAD When Akio Toyoda took control of the world’s largest car company back in 2009, he set as a goal the target of injecting …*

*…*

*(15) Title: Toyota Corolla sedan 2020 Review  
The Toyota Corolla is one of the world’s best-selling cars — but most people think of (and buy) the hatchback. The Corolla sedan has always been a bit, well, forgettable. But there’s change in the wind. A fresh new design, higher equipment levels, a more engaging drive experience …*

**Article of specific car make model is ranked higher than generic article**

Upon further analysis, it’s evident that queries which aren’t focused on specific car models, like “What is a hybrid car?”, yield mixed results. The first document returned is generic, the second is somewhat in between, while the third and fourth specifically discuss Honda hybrids. For such inquiries, most individuals are inclined towards content that broadly discusses the concept of hybrids, their advantages, and disadvantages, rather than specifics of a certain car brand or model. Clearly, our system doesn’t fully meet these expectations.

*(1) Title: What is a hybrid vehicle?  
In the animal world, a hybrid is a cross between two different species or varieties. In the motoring world, the term has evolved to define an electric car that is also powered by an internal combustion engine. The electric motor does much of the work, but in the original concept of hybrid car …*

*(2) Title: Hybrids set to bloom  
it’s a hybrid. It’s no ordinary hybrid in the same mould as the Prius or the Civic, however. This is a car with a petrol V6 and traditional rear-wheel drive, up to date looks and high-tech, ergonomic design. Doug Soden, Manager, Product Development and Engineering at Toyota, anticipates that the uptake in hybrid sales in Australia may just …*

*(3) Title: Honda hybrid’s a good sport  
February 23, 2007. Hybrid technology meets the performance enthusiast in a sports car prototype (pictured) being shown by Honda at the coming Geneva Motor Show. The Small Hybrid Sports Concept uses advanced hybrid technology …*

*(4) Title: Honda Civic (2001-)  
next 50–100 years. Fuel cells are the current flavour for the future but affordable commercialisation is still at least 20 years away so in the meantime, a number of carmakers are heading down the hybrid route. Hybrids are essentially a traditional car powered by a small petrol engine with an additional electric motor run off rechargeable batteries …*

**Article about the wrong model is being returned**

When searching for “Mazda CX-9 2023 review”, the system predominantly retrieved articles about the Mazda CX-8, an incorrect model, despite the articles being from 2023. An article on the Mazda CX-9 from 2021 is tucked away further in the results, ranking fifth. Generally, users expect to see articles pertaining to the specific make and model they searched for, even if from a previous year, rather than content about a completely different model from the queried year.

*(1) Title: Mazda CX-8 2023 Review  
Since its 2018 launch, the Mazda CX-8 three-row family SUV has been part of the support act for more popular members of the CX line-up. But with the departure of the CX-9 by year’s end and a family of all-new, more expensive SUVs soon to debut, it’s time for the CX-8 to step into the limelight as clearly Mazda’s most affordable family wagon …*

*(2) Title: Mazda CX-8 2023 Video Review  
The Mazda CX-8 has been the tail-ender when it comes to sales results for the comprehensive line-up of CX-badged Mazda SUV models. But at a time when the top-end of the line-up is being overhauled with an all-new generation of models and the award-winning CX-9 is being dropped, the CX-8 is now clearly the brand’s three-row seven-seat price-leader …*

*…*

*(5) Title: Mazda CX-9 2021 Review  
There’s plenty of compliance in the suspension tune to ensure passengers aren’t bounced around too much (good when there’s sleeping kids). Tested in isolation, the CX-9 is hard to fault in terms of ride comfort, however as our recent comparison with the Kia Sorento proved , the Mazda is no longer class-leading in this department …*

To summarise the issues with our document retrieval system:

* When query does not specify model year, more recent articles aren’t prioritised.
* The system often ranks articles based on the match in year mentioned in the query, even if it sacrifices match in make-model.
* For broad queries, the system sometimes prioritises specific make-model articles over general ones.
* The system fails to rank articles about the exact car model year inquired over different model years.

Recognising these issues is pivotal, signalling that we’re far from the finish line.

After an intensive period of analysis, which spanned weekends and late nights, we have zeroed in on the primary issues.

Our repository is abundant with editorial pieces that delve into the same vehicle brand and model but from various years, such as the Mazda CX-9 reviews from 2018, 2019, and 2021. The structural layout of these articles bears striking similarities. Certain sections discuss fuel efficiency, while others focus on safety or the roominess of the interiors. When presented with the query, “What is Mazda CX-9 2019 fuel efficiency?”, several document chunks appear, each sourced from Mazda CX-9 articles spanning various years and addressing fuel efficiency. It’s worth noting that every document chunk denotes the title and year of its originating article. Thus, one could expect the dense vector of “Mazda CX8 v Mazda CX-9 2019 Comparison” to be more similar to the query’s dense vector compared to the dense vector of “Mazda CX-9 2016 Review”.

However, this isn’t the reality. The year, being just a singular word within the roughly 300 tokens of the content, doesn’t carry the weight it should. Evaluating the broader content, the first article seems to lean heavily on fuel efficiency than the second. The dense embedding process fails to recognise the pivotal role the year plays in the automotive sector. For instance, different years correspond to distinct model years of vehicles, hence the year should carry more weight during dense embedding vector creation. This predicament is labeled as the domain out-of-context challenge. Utilising a foundational embedding model, designed on a general dataset, means it misses the specific nuances of the automotive domain.

*(1) Title: Mazda CX-9 2016 Review  
170kW, 2.5-litre four-cylinder turbocharged-petrol engine paired with a six-speed automatic transmission. There’s no manual option or diesel powertrain available, and you can thank the heavy focus on developing this vehicle for the US market for this lack of fuel choice. That said, the petrol engine is still capable of achieving impressive fuel economy figures for a vehicle of its size, promising a combined cycle fuel consumption figure of 8.8L/100km …*

*(2) Title: Mazda CX-8 v Mazda CX-9 2019 Comparison  
The Mazda CX-9 is a very highly rated vehicle, having won carsales.com.au car of the year and several other awards as well. But it only comes with a 170kW/420Nm 2.5-litre four-cylinder turbo-petrol engine mated with a six-speed automatic transmission. Add in all-wheel drive — as we naturally did for this test — and you’re looking at a hefty 1924kg kerb weight. But Mazda still claims a reasonable 8.8L/100km fuel consumption figure. We’ll put that to the test over highways, byways and dirt tracks …*

This very issue is also the reason why “Mazda CX-8 2023 Review” receives a higher ranking than “Mazda CX-9 2021 Review” for the query “Mazda CX-9 2023 Review”. Our standard model fails to prioritise model matches over year matches, leading to such discrepancies.

A simpler analogy would be pondering whether “Fried chicken” is more similar to “Chicken soup” or “Fried rice”. The answer varies based on context. If ingredients are the focus, “Fried chicken” aligns with “Chicken soup”. But from a preparation perspective, it’s closer to “Fried rice”. Such interpretations are domain-centric.

Many RAG setups might sidestep the challenges we confront, given that their indexed articles don’t bear resemblances as ours do. For instance, for customer support use case most articles have distinct topics like “how to reset my password” or “how to change my email address”.

Addressing this could involve customising our embedding model with automotive-focused datasets. However, fine-tuning presents challenges. We’d need datasets of both positive (two sentences that are similar) and negative pairs (two sentences that are different), which is a substantial undertaking. Considering we leverage the OpenAI text-embedding-ada-002 model, fine-tuning isn’t even an option as they do not yet offer this service.

**Solution**

Our exploration led us to a more promising solution, merging four strategies:

* Hybrid Search (Combining Dense and Sparse Embedding)
* Hierarchical Document Ranking
* Instructor Dense Embedding
* Year Boosting.

**Hybrid Search (Combining Dense and Sparse Embedding)**

The key to our solution lies in the fusion of Dense and Sparse embeddings, aka “Hybrid Search.” Sparse embedding, in essence, similar to the conventional keyword or lexical matching. Surprisingly, this is the very traditional approach we initially aimed to supersede with dense embedding.

Let’s discuss more about this sparse embedding.

In the realm of dense embedding, every document chunk is encapsulated within a 1536 floating point vector. However, sparse embedding portrays these document chunks within an extended dimensional space, resulting in a considerably expansive vector. At its core, a basic sparse embedding is a vector wherein each slot represents a distinct word in your vocabulary. If a word is found in the document chunk, its corresponding slot in the vector is marked as 1; if absent, it remains 0. To optimise this, stopwords such as “is”, “and”, and “what” are typically excluded to streamline the vocabulary size. In our application, we’re dealing with a vocabulary comprising roughly 50,000 words, translating to a vector of equivalent size.

Furthermore, their primarily zero-filled nature optimises storage and computation, since a majority of the entries (being zeros) can be efficiently compressed or bypassed during calculations. The method for calculating the similarity score between two sparse embedding vectors mirrors that of two dense embedding vectors, utilising the cosine distance.

Moving beyond this elementary method, we encounter sophisticated techniques like the [BM25 ranking](https://medium.com/@evertongomede/understanding-the-bm25-ranking-algorithm-19f6d45c6ce) algorithm. While the foundational idea of BM25 builds upon the widely-used TF-IDF (Term Frequency-Inverse Document Frequency) approach — which emphasizes the importance of terms based on their frequency in a document relative to their frequency across all documents — BM25 refines it. Rather than simply allocating binary values or raw term frequencies, BM25 determines a weight for each term in the document chunk. This weight, influenced by both term frequency and the inverse document frequency, quantifies the term’s relevance, making BM25 more nuanced in capturing the significance of terms within context.

The appeal of sparse embeddings is rooted in their accuracy; they shine in ranking documents based on precise word-to-word matches between the document and the query. For instance, when using sparse embedding, the query “What is **Mazda CX-9 2019** fuel efficiency?” ranks higher for “Mazda CX8 v **Mazda CX-9 2019** Comparison” compared to “**Mazda CX-9** 2016 Review”. This is because the former has three exact unique word matches, while the latter has only two.

However, we can’t completely swap out our dense embedding in favour of sparse embedding. Doing so would circle us back to the initial challenges we faced with our search system, mainly overlooking semantic meaning. While dense embedding excels in evaluating document chunks based on their semantic meaning, it falls short in ranking them by precise word match. Conversely, sparse embedding prioritises exact word matches but may overlook broader semantic context. Therefore, to harness the strengths of both approaches, it’s essential to combine them. This fusion ensures a more comprehensive and effective search system.

We determine the rank of our documents based on the weighted average of two distinct scores. The initial score employs cosine distance against dense embedding, an approach we’ve previously adopted. Meanwhile, the secondary score employs the cosine distance against BM25 sparse embedding.

We derive the final score, intended for document ranking, using the following equation:

hybrid\_score = 0.75 \* dense\_score + 0.25 \* sparse\_score

The sparse\_score favours documents with a greater occurrence of words that match the query. Luckily, Pinecone natively supports this Hybrid Search mechanism. Our responsibility is streamlined to inputting both the dense and sparse vectors. By assigning weights, we emphasise the dense/semantic matches over the sparse/keyword ones, as indicated by the 0.75 to 0.25 ratio, allowing us to retrieve the top x relevant document chunks.

**Hierarchical Document Ranking**

Our secondary methodology involves a two-tiered ranking system. Initially, scores are calculated based on the content within document chunks using our hybrid search formula. Subsequently, these scores receive a boost, determined from the document’s title employing the same formula.

What’s the rationale? Titles, though succinct, frequently encapsulate pivotal information, such as a car’s make, model, and production year, essentially serving as a condensed summary of the content. As a result, ranking documents based solely on their titles proves more effective roughly half the time compared to ranking based on content, especially with sparse embedding. Consider the query, “What is **Mazda CX-9 2019** fuel efficiency?” An article titled “**Mazda CX-9 2019** review” will outpace “**Mazda CX-9** 2018 review” because the former has more matching terms.

To incorporate this, our formula undergoes a modification:

final\_score = 0.5 \* title\_hybrid\_score + content\_hybrid\_score

Thanks to these enhancements, we’ve already observed a noticeable uptick in the relevance of our document retrieval results. For instance, the query “What is Mazda CX-9 2019 fuel efficiency?” now finally yields the correct document chunk from the article “Mazda CX-8 v Mazda CX-9 2019 Comparison”, a feat attributable to the sparse embedding.

*(1) Title: Mazda CX-8 v Mazda CX-9 2019 Comparison  
The Mazda CX-9 is a very highly rated vehicle, having won carsales.com.au car of the year and several other awards as well. But it only comes with a 170kW/420Nm 2.5-litre four-cylinder turbo-petrol engine mated with a six-speed automatic transmission. Add in all-wheel drive — as we naturally did for this test — and you’re looking at a hefty 1924kg kerb weight. But Mazda still claims a reasonable 8.8L/100km fuel consumption figure. We’ll put that to the test over highways, byways and dirt tracks …*

However, one challenge remains: our document retrieval system doesn’t prioritise matches based on car make/model over matches on model years.

**Instructor Large Dense Embedding**

The issue stems from our embedding model lacking domain-specific knowledge in the automotive sector, which is essential to understand that matches based on make/model carry more weight than matches on model years.

Our solution? Transitioning to the [Instructor Large Embedding](https://instructor-embedding.github.io/) model. This state-of-the-art model offers a unique advantage: it can be effortlessly customised to specific domains or tasks using straightforward text prompts. Beyond its customisation capabilities, the model itself is inherently deliver a better quality embedding.

By merely supplying the prompt, “Represent a car review article question for retrieving relevant documents,” to the Instructor Large model, it adeptly refines our embedding for the car review niche. This process mirrors the fine-tuning of an embedding model for a specific domain but sidesteps the associated costs. Consequently, our embedding now adeptly prioritises attributes such as car make and model over the production year.

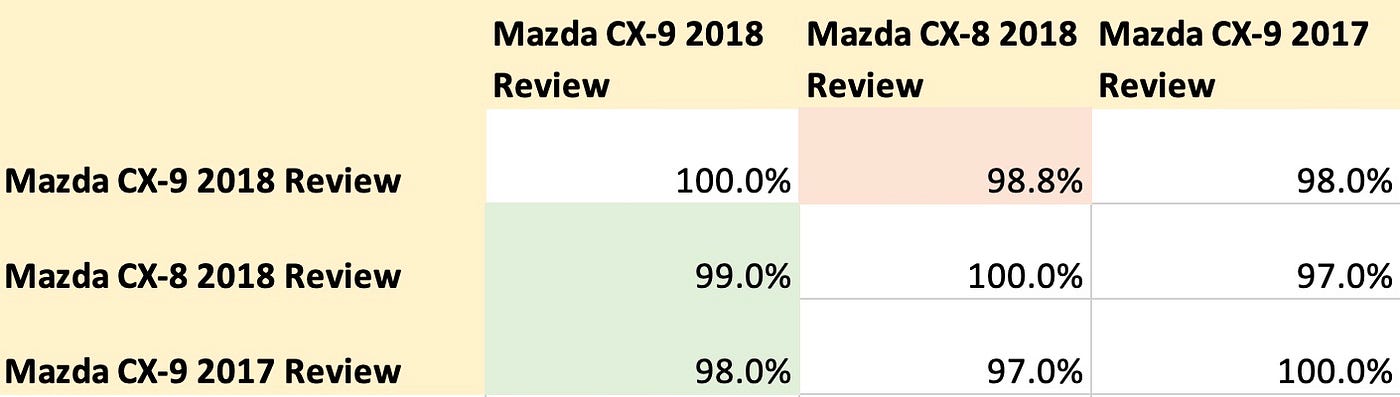
The following code showcases the superiority of the Instructor Large embedding for our specific needs when compared to OpenAI’s text-embedding-ada-002.

from embedding.azure\_embedding import AzureEmbedding  
import numpy as npsentences = ["Mazda CX-9 2018 Review", "Mazda CX-8 2018 Review", "Mazda CX-9 2017 Review"]print("Azure Embedding")  
emb = AzureEmbedding()  
emb\_1 = emb.embedd(sentences)  
emb\_2 = emb.embedd(sentences)  
emb\_1a = np.array(emb\_1)  
emb\_2a = np.array(emb\_2)  
print(emb\_1a @ emb\_2a.T)  
print("\n")print("Instructor-large")  
from InstructorEmbedding import INSTRUCTOR  
model = INSTRUCTOR('hkunlp/instructor-large')  
instruction = "Represent a car review article:"  
doc = [[instruction, sentence] for sentence in sentences]embeddings\_1 = model.encode(doc)  
embeddings\_2 = model.encode(doc)  
similarity = embeddings\_1 @ embeddings\_2.T  
print(similarity)  
print("\n")

We presented two distinct embeddings with the task of generating dense representations for these three sentences:

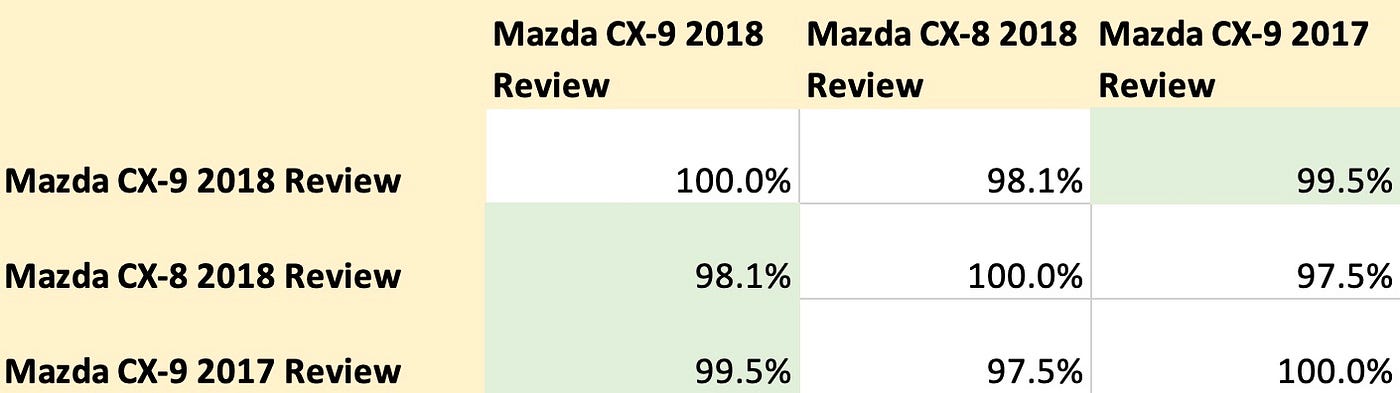
* Mazda CX-9 2018 Review
* Mazda CX-8 2018 Review
* Mazda CX-9 2017 Review

Next, we crafted a similarity matrix (based on cosine distance score) between these sentences. The results from text-embedding-ada-002 showed that “Mazda CX-9 2018 Review” had closer similarity to “Mazda CX-8 2018 Review” than it did to “Mazda CX-9 2017 Review,” which is not the expected outcome.



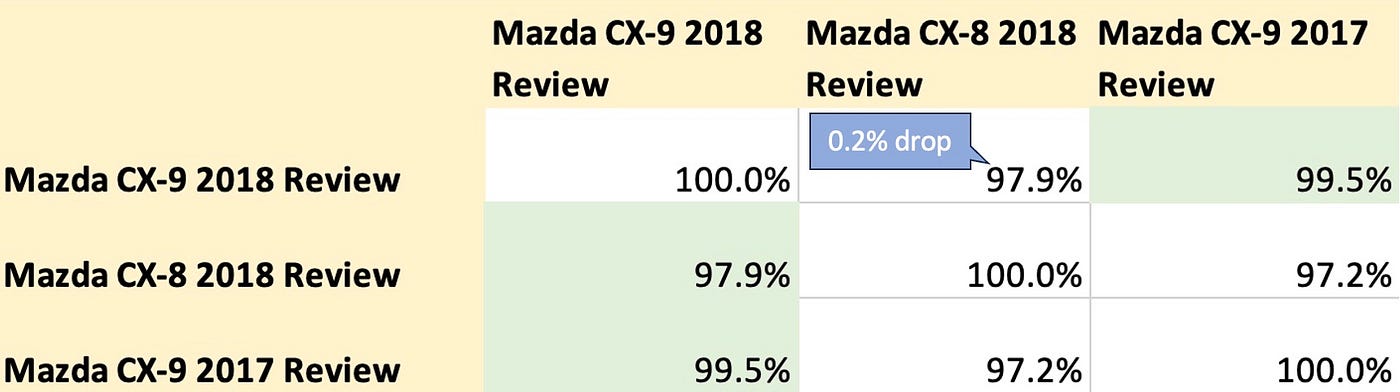
Similarity matrix for text-embedding-ada-002 (green is good, red is bad)

Conversely, using the Instructor Large model without a custom prompt, the dense embedding for “Mazda CX-9 2018 Review” showed greater similarity to “Mazda CX-9 2017 Review” compared to “Mazda CX-8 2018 Review” — which is the anticipated result.



Similarity matrix for Instructor Large without a custom prompt (green is good, red is bad)

When we incorporated the custom prompt “Represent a car review article question for retrieving relevant documents”, the dense vector’s similarity matrix displayed improved results. Did you notice the 0.2% decrease in similarity score between “Mazda CX-9 2018 Review” and “Mazda CX-8 2018 Review”, while the score between “Mazda CX-9 2018 Review” and “Mazda CX-9 2017 Review” stayed the same?



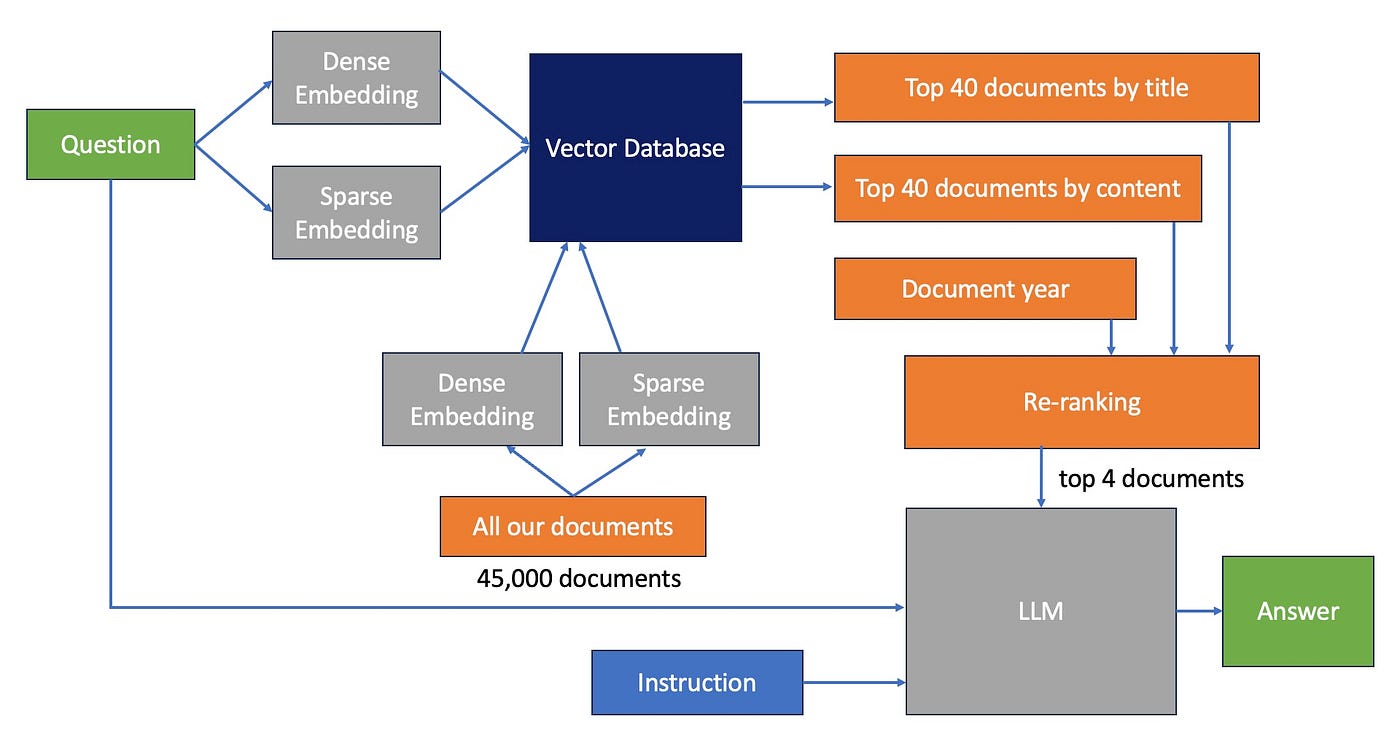
Similarity matrix for Instructor Large with a custom prompt (green is good, red is bad)

Additionally, while text-embedding-ada-002 requires 1 second for completion due to its external API call to Azure, Instructor Large operates at a brisk pace, finishing in under 150ms when hosted on a decently-sized EC2 instance. This significantly enhances our overall search response time.

**Year boosting**

One last challenge remains: our document retrieval system did not prioritise recent articles when the query didn’t specify a particular car model year. The fix for this was straightforward. We decided to assign additional scores to more recent documents. Through careful crafting, we formulated a quadratic function that adds a slight bonus score to recent articles, with this score diminishing as the article’s age increases. It’s important to note that we meticulously designed this function so that the boost from recency would never surpass the relevance of a year-match between the document and the query. This way, we avoid consistently favouring more recent articles even when a specific year is mentioned in the query.

With these adjustments, we’ve successfully addressed big majority our document retrieval concerns! Below, you’ll find the finalised architecture diagram.



Our final improved RAG architecture

**Conclusion**

We’ve effectively resolved most of our document retrieval hurdles. In our offline tests, 40% of the results favoured our new solution over the existing Google search. We’re now integrating this technology into our front-end UX, gearing up for an A/B test deployment to truly gauge the efficiency of this revamped search compared to the existing version. As previously noted, we also grappled with the quality of direct answers. While we’ve found a solution for this, I’ll delve into those details in a forthcoming blog post.

A key insight I’d like to impart is that while Retrieval Augmented Generation (RAG) holds immense potential, it isn’t the panacea that a lot of articles and posts might lead you to believe especially with the most common setup. Hybrid search and a few other technique we have covered stand out as a potent method for enhancing relevant document retrieval. The ever-evolving landscape of development and research promises further refinements to this approach, making it even more robust in the future.

Please reach out to me via [linkedin](https://www.linkedin.com/in/agustinus-nalwan/) for connection.

All images, unless otherwise noted, are by the author.

Editorial content used in this article is publically available online and owned by Carsales.