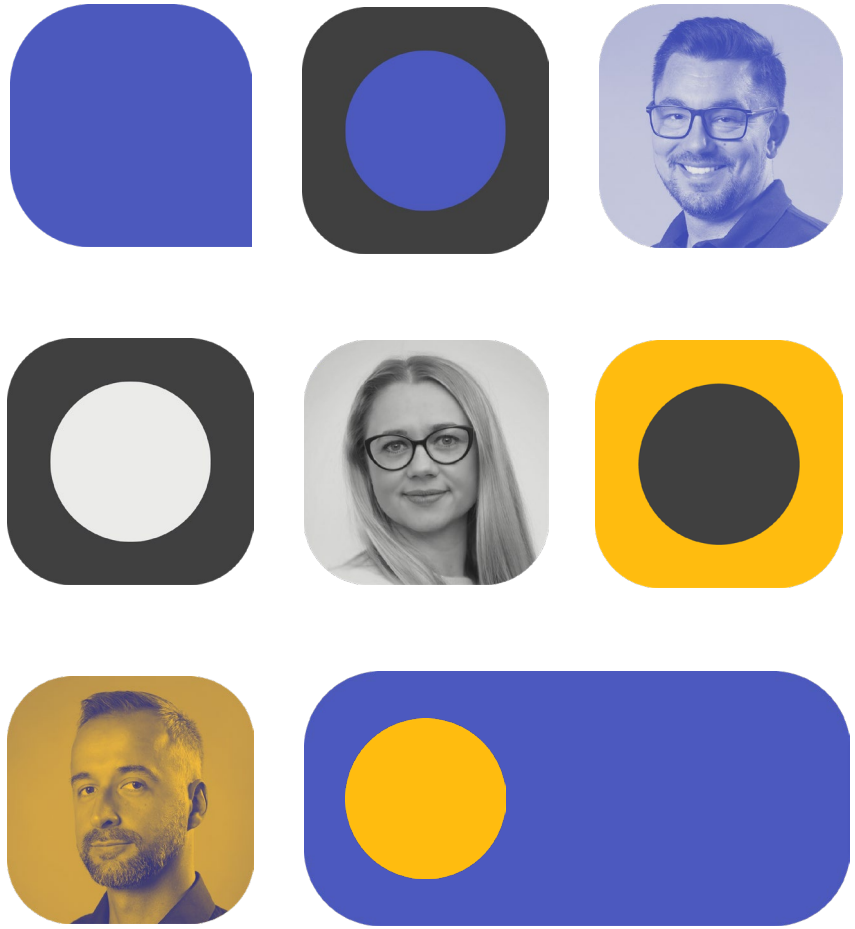
Jak AI korzysta z wiedzy, którą jej dostarczasz?

O czym będziemy dziś rozmawiać?

1. Dlaczego powstał RAG?
2. Jak przekształcić dokument w wiedzę dla modelu.
3. Czym jest chunking.
4. Co to embedding i na czym polega wyszukiwanie semantyczne.
5. Bazy wektorowe i ANN.
6. Retrieval engineering + zaawansowane techniki
7. Generowanie odpowiedzi w systemie z RAG.
8. Use-case.



ecom

o prowadzącym

AI Engineer z doświadczeniem w insurtechu, pracujący z modelami uczenia maszynowego oraz generatywną AI (LLM).

Na co dzień projektuje i wdraża systemy AI rozwiązujące realne problemy biznesowe, łącząc technologię, dane i ludzką kreatywność.

Jesteśmy zespołem **100 specjalistów IT i biznesu** insurtech.

Na co dzień wspieramy klientów w zakresie **kompleksowej transformacji cyfrowej**. Wywodzimy się z branży ubezpieczeniowej.

Naszą misją jest **upraszczanie codziennego życia** użytkowników za pomocą naszych rozwiązań.

CUK UBEZPIECZENIA
nau
ubezpieczenia AZ
esBank BANK SPÓŁDZIELCZY
CMF www.cmf.pl
DIAMOND FINANCE UBEZPIECZENIA
UBEZPIECZENIA CONDITOR MULTIAGENCJA
ANG
ERGO HESTIA
TUZ UBEZPIECZENIA
TOWARZYSTWO UBEZPIECZEŃ WZAJEMNYCH TUW
LINK 4
Balcia insurance
Trasti SIMPLY FAIR
mBank
MJM HOLDINGS
bolttech

1,95 mln

Sprzedanych polis

1,7 mld

Przypisu w rozwiązaniach ecom

8 mln

kalkulacji

83
System usability score

*Rocznie, dane za 2025

ecom

Dlaczego dwa systemy RAG używające tego samego modelu LLM mogą dawać zupełnie inną jakość odpowiedzi?

Paradoks LLM

LLM to nie baza danych

LLM nie przechowuje dokumentów

LLM nie zna prywatnych repozytoriów/PDF/maili

LLM może brzmieć pewnie nawet wtedy kiedy nie ma danych

Wiedza

parametryczna vs nieparametryczna

- Wiedza zapisana w wagach modelu
- Trudna do aktualizacji
- Dobra do wiedzy ogólnej
- Nie ma zawężonej specjalizacji pod dane środowisko specjalistyczne

- Zewnętrzne dokumenty
- Baza wektorowa
- Metadane
- Możliwość aktualizacji pamięci poza modelem

https://arxiv.org/abs/2005.11401

Co było przed RAG?

- **Bazowy LLM**

  prosty w użyciu, ale nie zna prywatnych danych.

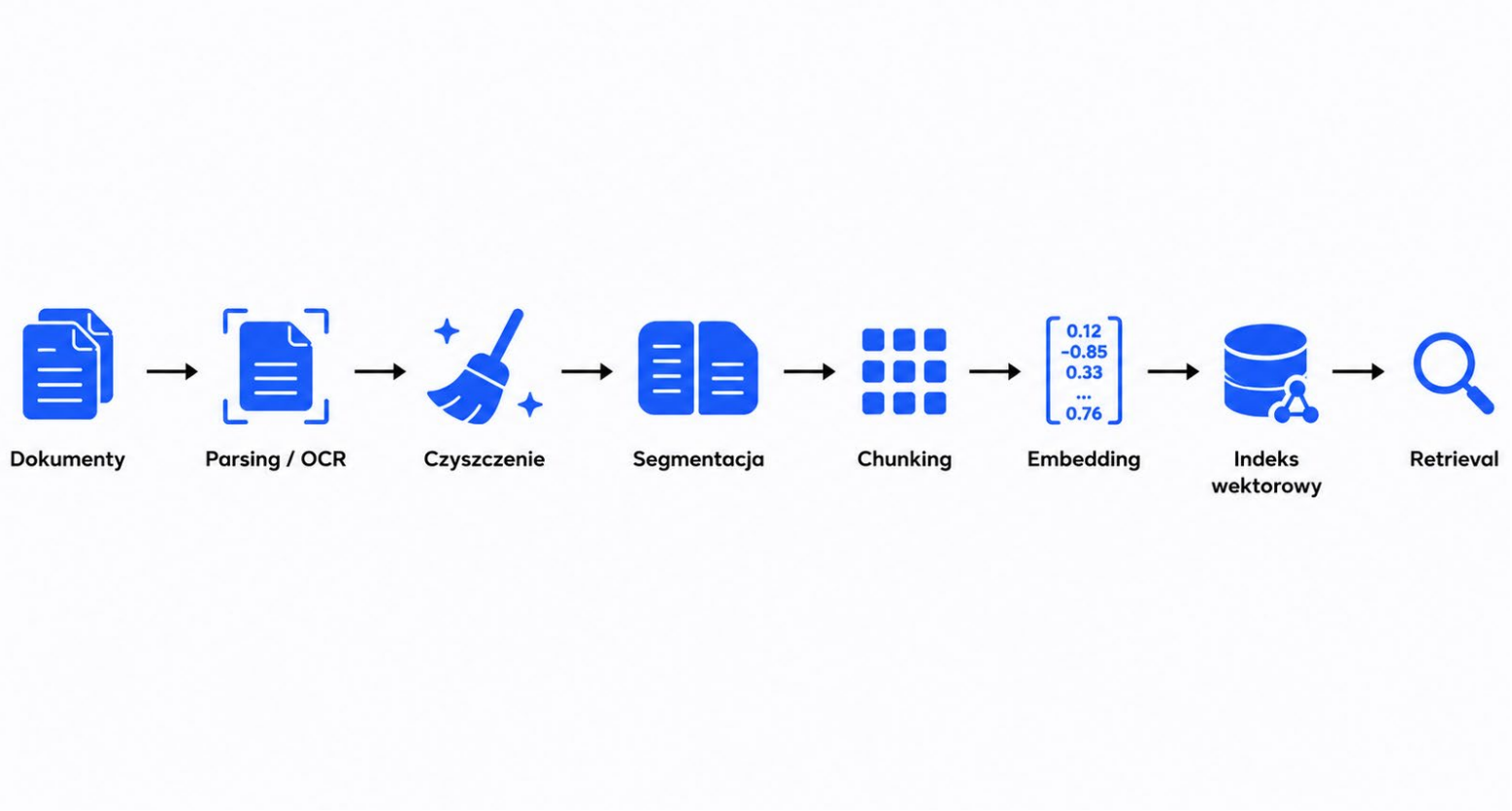
- **Fine-tuning**

  uczy stylu lub zachowania, ale słabe rozwiązanie przy często zmieniającej się wiedzy.

- **Wrzucanie dokumentów do promptu**

  szybkie rozwiązanie, ale szybko dobijamy limitu kontekstu, wyższe koszty i chaos w odpowiedziach modelu.

Jak przekształcić dokument w wiedzę dla modelu?



Dokument nie jest wiedzą.

Dla człowieka PDF jest dokumentem. Dla systemu RAG PDF jest problemem przetwarzania.

Najczęstsze błędy przy danych wejściowych:

- wrzucanie surowych PDF-ów bez czyszczenia,
- indeksowanie stopki i nagłówków jako treści,
- brak informacji o wersji dokumentu,
- duplikaty dokumentów,
- stare i nowe procedury w tej samej bazie,
- brak metadanych,
- brak kontroli uprawnień,
- źle przetworzone tabele.

Czym jest chunking?

Chunk to jednostka wiedzy, którą system może odzyskać w retrievalu.

To nie jest tylko „kawałek tekstu”.

Chunk określa:

- co trafi do embeddingu,
- co będzie porównywane z pytaniem,
- co zobaczy LLM,
- czy odpowiedź będzie miała kontekst.

Kompromisy w chunkingu

Chunk	Zaleta	Problem
Mały	precyzyjny retrieval	może brakować kontekstu
Duży	więcej kontekstu	gorsza trafność, więcej szumu
Z overlapem	zachowuje ciągłość	więcej danych i kosztów
Semantyczny	lepsza spójność	bardziej złożony
Strukturalny	respektuje dokument	wymaga dobrego parsera

Embedding i wyszukiwanie semantyczne - problem wyszukiwania słów

BAZA WIEDZY

Wiedza ogólna

Dokument 1
Stolica Francji to Paryż. Paryż leży nad rzeką Sekwaną i jest znany z wieży Eiffla.

Dokument 2
Woda wrze w temperaturze 100°C na poziomie morza. Zamarza w 0°C.

Dokument 3
Mount Everest to najwyższy szczyt świata, mający 8 848 m n.p.m., położony w Himalajach.

Dokument 4
Ziemia jest trzecią planetą od Słońca i jedyną znaną planetą zamieszkaną przez ludzi.

Dokumenty z bazy wiedzy są dzielone na fragmenty (chunki), a następnie zamieniane na wektory (embeddingi).

PYTANIE UŻYTKOWNIKA

Jaka jest stolica Francji?

EMBEDDING (WEKTORY)

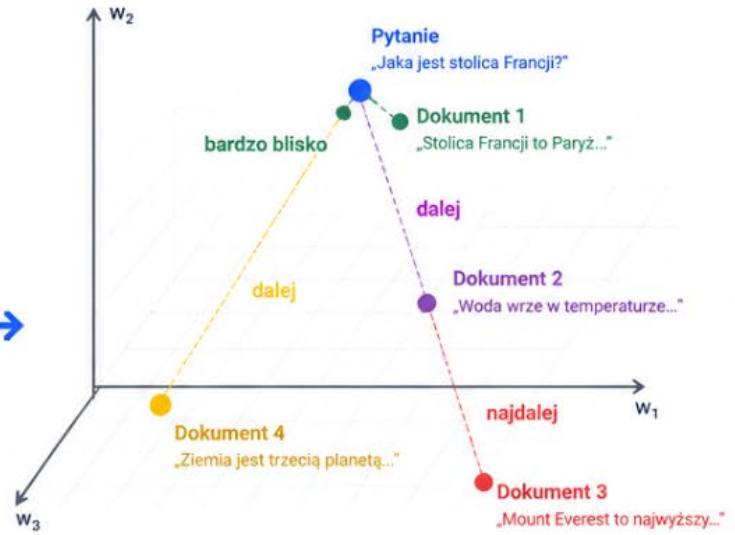
Każdy tekst zamieniany jest na wektor liczbowy (tu 3-wymiarowy dla uproszczenia)

		w_1	w_2	w_3
Pytanie "Jaka jest stolica Francji?"	→	0.82	0.11	0.41
Dokument 1 Stolica Francji to Paryż..."	→	0.80	0.10	0.42
Dokument 2 Woda wrze w temperaturze...	→	-0.20	0.70	0.10
Dokument 3 Mount Everest to najwyższy...	→	-0.60	-0.20	0.75
Dokument 4 Ziemia jest trzecią planetą...	→	-0.30	-0.40	0.60

Embedding odwzorowuje znaczenie tekstu. Semantycznie podobne teksty mają wektory blisko siebie w przestrzeni wektorowej.

PRZESTRZEŃ WEKTOROWA

Teksty reprezentowane jako punkty w przestrzeni.
Odległość = różnica znaczenia.



Bazy wektorowe i ANN:

Dlaczego nie liczyć podobieństwa do wszystkiego?

ANN, czyli **Approximate Nearest Neighbor**, polega na tym, że nie zawsze szukamy matematycznie najbliższego wektora metodą brute force. Szukamy bardzo dobrego przybliżenia dużo szybciej. To jest konieczne, bo dokładne porównywanie query z milionami wektorów jest kosztowne.

Algorytm	Największa zaleta	Największa wada	Kiedy wybrać
Flat / brute force search	100% dokładności	Bardzo wolny przy skali	benchmark, małe zbiory
HNSW	wysoki recall i niska latencja	dużo RAM	domyślny wybór dla RAG
IVF	szybki na dużych zbiorach	wymaga treningu i strojenia	duże dane, dobra klasteryzacja
PQ	mało pamięci	spadek dokładności	bardzo duża skala
IVF-PQ	skalowalność + kompresja	więcej kompromisów	miliony+ wektorów
LSH	prosta idea	często słabszy recall	specyficzne przypadki
ScaNN	bardzo szybki search	tuning/ekosystem	duża skala, niska latencja
DiskANN	duża skala poza RAM	złożoność	ogromne indeksy na SSD

Sparse Retrieval - wyszukiwanie po słowach kluczowych

Jak to działa?

System buduje indeks słów: urlop, pracownik, procedura, wniosek, a następnie dla zapytania „urlop pracownika” szuka dokumentów zawierających te słowa.

Algorytm:

BM25 (Najpopularniejszy algorytm sparse retrieval)

- uwzględnia frequency – im rzadziej występuje słowo/fraza tym ma większą wartość – jest charakterystycznym tagiem.

Dokument:

Procedura urlopowa pracownika

Query:

urlop pracownika

To idealne dopasowanie

Retrieval engineering jest ważniejszy niż model

Sparse Retrieval

Zalety

- Szybki,
- Tani,
- Działa bez embeddingów,
- Świetny dla:
  - kodów produktów,
  - numerów procedur,
  - nazw własnych,
  - Identyfikatorów.

Wady

- Nie rozumie znaczenia,
- Nie radzi sobie z parafrazami.

Przykład: „Jak mogę wziąć wolne?” ->
Dokument PROCEDURA URLOPOWA ->
brak wspólnych słów i BM25 może nie znaleźć wyniku

https://arxiv.org/abs/2005.11401

Dense Retrieval - wyszukiwanie po znaczeniu

Jak to działa?

Nie porównujemy słów – porównujemy znaczenie.

Dlaczego to działa?

Model embeddingowy nauczył się relacji

„urlop” jest podobne do ”dzień wolny, wakacje,

leave request”.

Architektura:

Powstają dwa embeddingi

(Q i D), a później similarity (Q,D)

Proces:

Query: „Jak mogę wziąć wolne? -> Embedding -> Wektor

Dokument: PROCEDURA URLOPOWA -> Embedding -> Wektor

Liczymy podobieństwo (najczęściej cosine similarity/dot product)

Dense Retrieval

Zalety

- Rozumie znaczenie,
- Działa do parafraz,
- Świetny do pytań naturalnych.

Wady

Słabszy dla dokładnych identyfikatorów

- Może pomylić podobne znaczeniowo dokumenty,
- Wymaga embeddingów.

Przykład: „Procedura BHP-17” Dense Search może uznać „Procedura BHP-15” za podobną.

Hybrid Search - najlepsze z dwóch światów

Jak to działa?

- BM25 rozumie słowa.
- Dense Search rozumie znaczenie.
- Żaden nie jest idealny.

Rozwiązanie: Uruchamiamy oba równolegle.

Dlaczego obecnie dominuje?

Większość systemów produkcyjnych:

- Azure AI Search,
- Elastic,
- OpenSearch,
- Pinecone,
- Weaviate,

wspiera hybrid search.

Proces:

Question -> BM25 -> TopK

Question -> Dense Search -> TopK

Później merge -> Ranking

Hybrid Search

Zalety

- Wyższy recall,
- Lepsza precyzja,
- Mniej błędów wyszukiwania.

Wady

- Bardziej skomplikowany,
- Dwa indeksy,
- Większy koszt.

Przykład:
Pytanie: „Procedura BHP-17 dotycząca pracy z chemikaliami”

- BM25 znajdzie BHP-17.
- Danse Search znajdzie: chemikalia, substancje niebezpieczne, laboratorium.

Hybrid Search znajdzie oba aspekty.

Reranking - drugi etap wyszukiwania

Jak to działa?

Reranker analizuje pytanie + dokument razem, co Poprawia ilość i jakość zwracanych wyników oraz pozwala zrozumieć relacje pomiędzy dokumentem, a treścią.

Proces:

Question -> Retriever -> Top50 -> Cross Encoder -> Top5 -> LLM

Reranking

Zalety

- Ogromny wzrost jakości,
- Mniejszy szum,
- Bardziej trafne dokumenty.

Wady

- Wolniejszy,
- Droższy,
- Nie nadaje się do przeszukiwania milionów dokumentów.

Cecha	BM25	Dense	Hybrid	Reranker
Rozumie znaczenie	❌	✅	✅	✅
Rozumie dokładne słowa	✅	⚠️	✅	✅
Bardzo szybki	✅	✅	⚠️	❌
Najlepszy dla kodów i ID	✅	❌	✅	✅
Najlepszy dla parafraz	❌	✅	✅	✅
Koszt	niski	średni	średni	wysoki

Zaawansowane techniki retrievalu

Query rewriting

Query: „Czy mogę **to** zrobić po terminie?”

Zastosowania:

- pytania wieloznaczne,
- follow-up questions,
- chat history,
- skróty myślowe użytkownika.

Problem:

Nie wiadomo, czym jest „to”.

Query rewriting może przekształcić pytanie na: „Czy mogę **złożyć wniosek urlopowy** po wymaganym terminie?”

Zaawansowane techniki retrievalu

Multi-query retrieval - Jedno pytanie generuje kilka zapytań

Query: „Jak wygląda proces zatrudnienia?”

Możliwe warianty:

- etapy rekrutacji,
- procedura zatrudnienia,
- onboarding pracownika,
- podpisanie umowy.

Potem wyniki są scalane i rerankowane.

To zwiększa recall, ale może pogorszyć precision, jeśli system pobierze za dużo luźno powiązanych fragmentów.

LLM jest ostatnim etapem

Dopiero teraz używamy modelu generatywnego.

Prompt RAG:

Masz odpowiedzieć wyłącznie na podstawie kontekstu.

Kontekst:

[chunk 1]

[chunk 2]

[chunk 3]

Pytanie:

...

Odpowiedź:

...

LLM jest ostatnim etapem

Odpowiedź powinna zawierać:

- źródło,
- dokument,
- sekcję,
- fragment,
- ewentualnie numer strony.

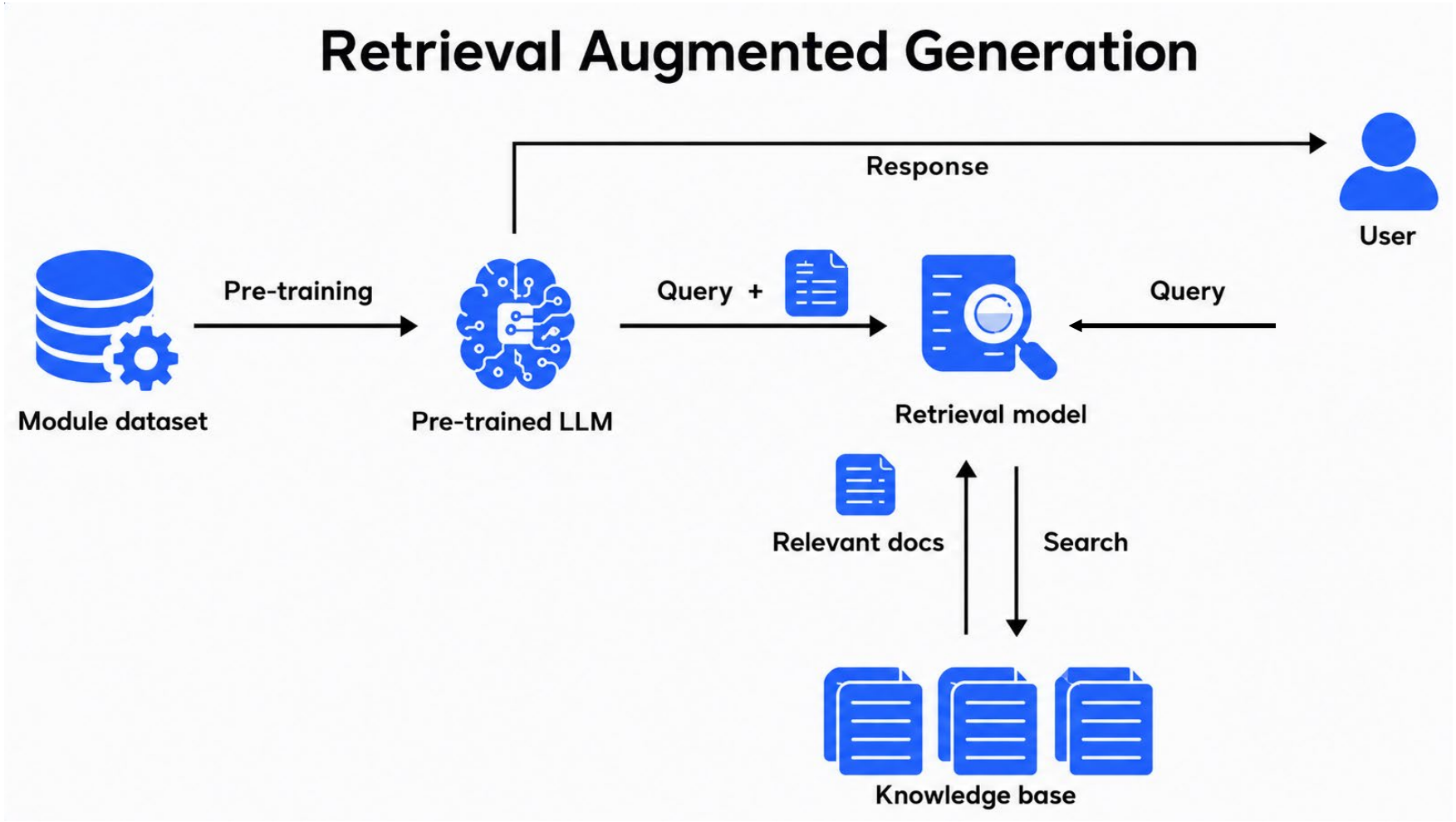
To ważne szczególnie w:

- prawie,
- medycynie,
- edukacji,
- finansach,
- dokumentacji technicznej,
- procedurach firmowych.

RAG był od początku motywowany m.in. problemem aktualizacji wiedzy i dostarczania provenance, czyli pochodzenia informacji.

W dobrym RAG model nie ma zgadywać.

Model ma syntetyzować odpowiedź z dostarczonych źródeł.


Retrieval Augmented Generation
Response
User
Pre-training
Query +
Query
Module dataset
Pre-trained LLM
Retrieval model
Relevant docs
Search
Knowledge base

Gdzie RAG się psuje?

Najczęstsze awarie:

1. parser źle odczytał dokument,
2. chunking przeciął istotny kontekst,
3. embedding nie złapał domenowego znaczenia,
4. retriever znalazł podobne, ale nieodpowiednie fragmenty,
5. reranker źle przestawił kolejność,
6. prompt pozwolił modelowi zgadywać,
7. baza zawiera stare lub sprzeczne dokumenty,
8. użytkownik nie powinien mieć dostępu do danego źródła.

ecom

Use case

Dyskusja

Use case - Uczelniana baza wiedzy

Dane:

- regulaminy,
- sylabusy,
- procedury,
- harmonogramy,
- FAQ dziekanatu.

Ryzyka:

- nieaktualne dokumenty,
- różne wersje regulaminów,
- pytania wymagające interpretacji prawnej.

Dlaczego RAG pasuje:

- dane są tekstowe,
- często aktualizowane,
- użytkownicy pytają naturalnym językiem,
- potrzebne są źródła.

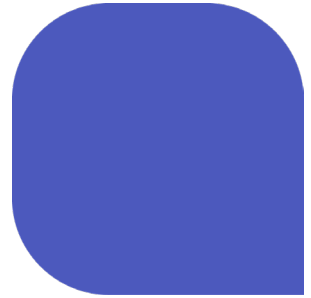
Asystent dokumentacji technicznej

Dane:

- dokumentacja API,
- README,
- changelogi,
- issues,
- wiki projektowe.

Dlaczego RAG pasuje:

- wiedza zmienia się szybciej niż model,
- developerzy pytają kontekstowo,
- można cytować źródła,
- można filtrować po repozytorium, wersji, module.



Wsparcie prawne lub compliance

Dane:

- regulacje,
- interpretacje,
- procedury,
- audyty,
- polityki.

Dlaczego RAG pasuje:

- potrzebne źródła,
- ważna aktualność,
- odpowiedź musi być uzasadniona.

Uwaga:

To raczej system wspomagający eksperta, nie automatyczny decydent.

Podsumowanie

W systemach RAG problemem rzadko jest tylko model.
Najczęściej problemem są dane, retrieval, chunking, metadane i ewaluacja.

RAG to nie „chat z PDF-em”.
RAG to architektura dostarczania właściwego kontekstu we właściwym momencie.

From user stories to success stories

www.ecom.software