How Perplexity Beat Google on AI Search

4 min read · 1 day ago

Civil Learning Follow

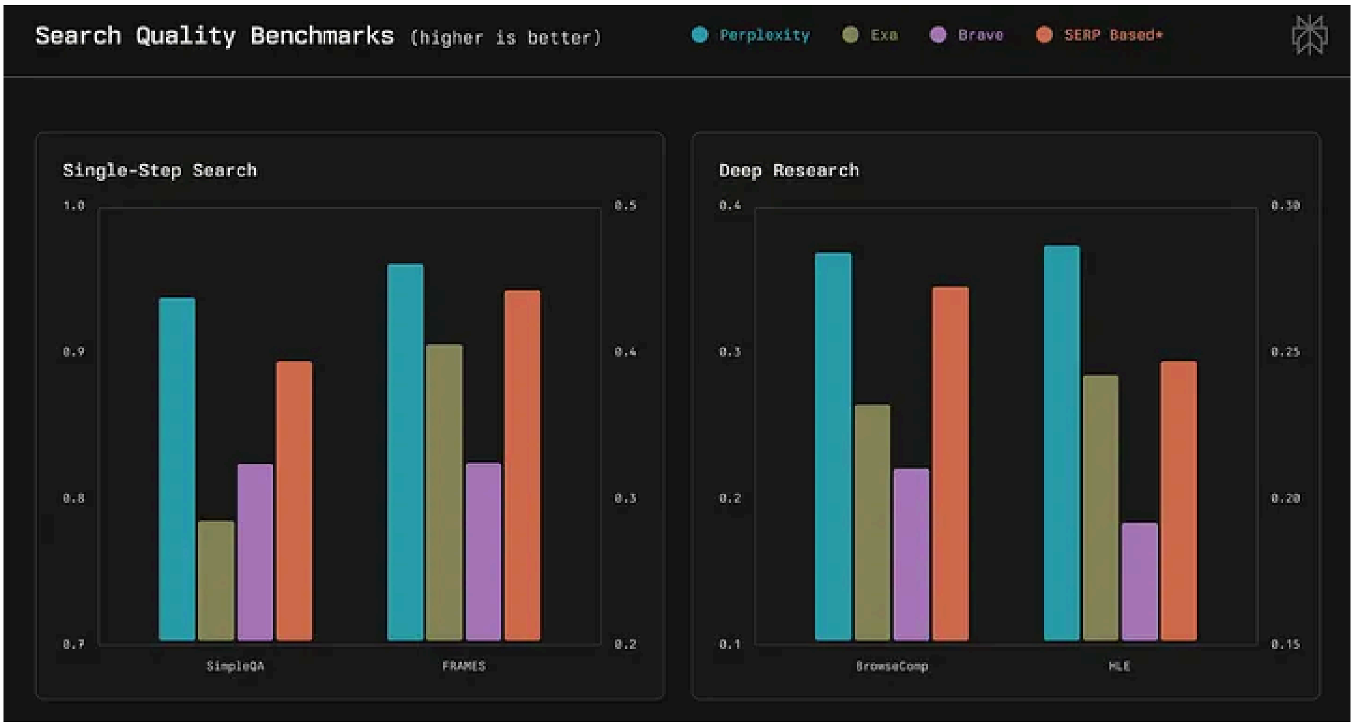
If you've tried **Perplexity** recently, you've likely noticed something strange — it seems smarter than search engines usually are. It's fast, confident, and the responses sound like they were written by someone who *actually read* the web just a few minutes ago.

That's not magic. It's **Vespa.ai**.

Recently, Perplexity revealed that the system behind its search — the one powering all those sleek, AI-generated answers — runs on Vespa. They even made their search service accessible to others and released an open-source evaluation tool so people could test its quality themselves.

And guess what? It's really good — good enough that Perplexity's results often look cleaner and faster than Google's.

Let's analyze why that is and how Vespa helps them achieve it. I'll include a few code snippets so you can see what's happening behind the scene.



. . .

Why Regular Search Engines Don't Cut It for AI

Traditional search engines are designed for people — you enter a few words, and they return links.

But AI requires more than that. When a language model responds to your question, it doesn't seek *a web page* — it wants the **correct piece of information** with the most recent context.

Perplexity found that to make this work, a retrieval system needs to hit three marks:

- Be complete, up to date, and lightning-fast.
- Understand content deeply, not just by keywords.
- Blend text and semantic signals (that's the "hybrid" bit).

The problem? Most tools could only excel at one of these. Search engines handled text perfectly but didn't understand meaning. Vector databases understood semantics but struggled with filtering or metadata.

So, they chose Vespa — a search engine designed to manage *both worlds* on a large scale.

. . .

Keeping It Fast and Fresh

When you're crawling billions of pages, keeping them current is a nightmare. Vespa made that easy for Perplexity by automatically distributing data across nodes and updating indexes in real time.

Here's a Python example demonstrating data feeding:

```
from vespa.application import Vespa

app = Vespa(url="http://localhost", port=8080)
# Add documents to the index
docs = [
    {"id": "doc1", "fields": {"title": "AI Search Revolution", "content": "Vesp
    {"id": "doc2", "fields": {"title": "Vector Search Basics", "content": "Sema
]
for d in docs:
    app.feed_data_point(schema="articles", data_id=d["id"], fields=d["fields"])
# Quick updates to keep things fresh
app.update_data(schema="articles", data_id="doc1", fields={"popularity": 9.8})
```

That last line — the `update_data` — is where Vespa shines.
You can update ranking signals or metadata instantly without reindexing the whole document.

That's how Perplexity stays "fresh." It can react instantly to new information that appears online.

. . .

Understanding Content the Way AI Needs It

AI models don't read entire web pages. They skim quickly, focusing only on the relevant paragraphs or sections that directly answer your question.

Vespa lets you rank *chunks* of documents, not just entire files. It's like slicing your data into meaningful bites before feeding it to your model.

Here's a small taste of that idea:

```
query = "What makes hybrid search better than traditional search?"

response = app.query(
    schema="articles",
    query=query,
    hits=5,
    body={
        "yql": "select title, content from sources * where userQuery();",
        "presentation": {"summary": "chunk"}
    }
)
for hit in response.hits:
    print(hit["fields"]["content"])
```

Instead of full pages, you get *snippets* — brief, relevant sections that work well for RAG systems.
That's why Perplexity's answers are clear and focused instead of rambling.

. . .

The Magic of Hybrid Search

Here's the part that makes engineers smile.

Vespa can merge **keyword-based (lexical)** and **embedding-based (semantic)** search into a single process. This enables it to find exact matches *and* grasp intent.

In simple terms, it knows both *what you said* and *what you meant.*

Let's see a simplified example:

```
query_vector = model.encode("real-time AI search with Vespa")

response = app.query(
    schema="articles",
    ranking="hybrid",
    hits=5,
    body={
        "yql": "select * from sources * where ([{\"targetHits\":10}]nearestNeig
        "ranking.features.query(query_vector)": query_vector.tolist(),
        "ranking.profile": "hybrid"
    }
)
for hit in response.hits:
    print(hit["fields"]["title"], "-", hit["relevance"])
```

This combines semantic closeness (through vectors) with keyword accuracy. That's the true advantage — the AI doesn't overlook anything relevant, and you still receive sharp, context-aware results.

. . .

Ranking Like a Pro

Even after finding the right candidates, Vespa doesn't stop. It goes through a **multi-stage ranking**, where early filters are fast and rough, and later ones use more powerful ML models for greater precision.

Here's a rough sketch of that process:

```
    # Stage 1: quick retrieval
    stage1 = app.query(schema="articles", query="AI ranking systems", ranking="fast

    # Stage 2: refine with a cross-encoder
    from sentence_transformers import CrossEncoder
    reranker = CrossEncoder("cross-encoder/ms-marco-MiniLM-L-6-v2")
    pairs = [( "AI ranking systems", hit["fields"]["content"]) for hit in stage1.hi
    scores = reranker.predict(pairs)
    # Combine both scores
    final = sorted(
        zip(stage1.hits, scores),
        key=lambda x: (x[0]["relevance"] * 0.5 + x[1] * 0.5),
        reverse=True
    )
    for hit, score in final[:5]:
        print(f"{hit['fields']['title']} - final score: {score:.3f}")
```

That mix of lightweight ranking and deep re-ranking is what gives Perplexity its trademark *"it just knows"* feeling.

. . .

Why It All Matters

Perplexity didn't just create a cool chatbot. They revolutionized **the way AI searches for information**.

With Vespa, they get:

- Billions of indexed documents.
- Real-time updates.
- Chunk-level precision.
- Hybrid retrieval that understands both words and meaning.
- Machine-learned ranking for final polish.

That's how you beat Google — not by building a bigger model, but by feeding your AI better data.

If you want to play with the same foundation, Vespa offers an open-source platform and a ready-to-go RAG Blueprint. Spin it up, tweak it, and see what happens when *search actually helps AI think*.

Perplexity Perplexity Ai Google AI Ai Agent

Follow

Written by Civil Learning

2.8K followers · 6 following

We share what you need to know. Shared only for information.

No responses yet

Bgerby

What are your thoughts?