Semantic Search:

From Keywords to Intent

SensioLabs

Stiven Llupa

Solutions Architect @ SensioLabs

Previously worked at very large corporates

And if there's one thing corporate loves in the past 2 years, it's *AI*

Objective

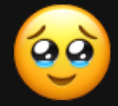
Experiment with adding *semantic*, AI-powered, search into *[product]* to move beyond simple keyword lookup

semantics | *noun*

the study of linguistic *meaning*. It examines what *meaning* is, how words get their *meaning*, and how the *meaning* of a complex expression depends on its parts.

Oskar is *moving*

The "Starry Night" is *moving*



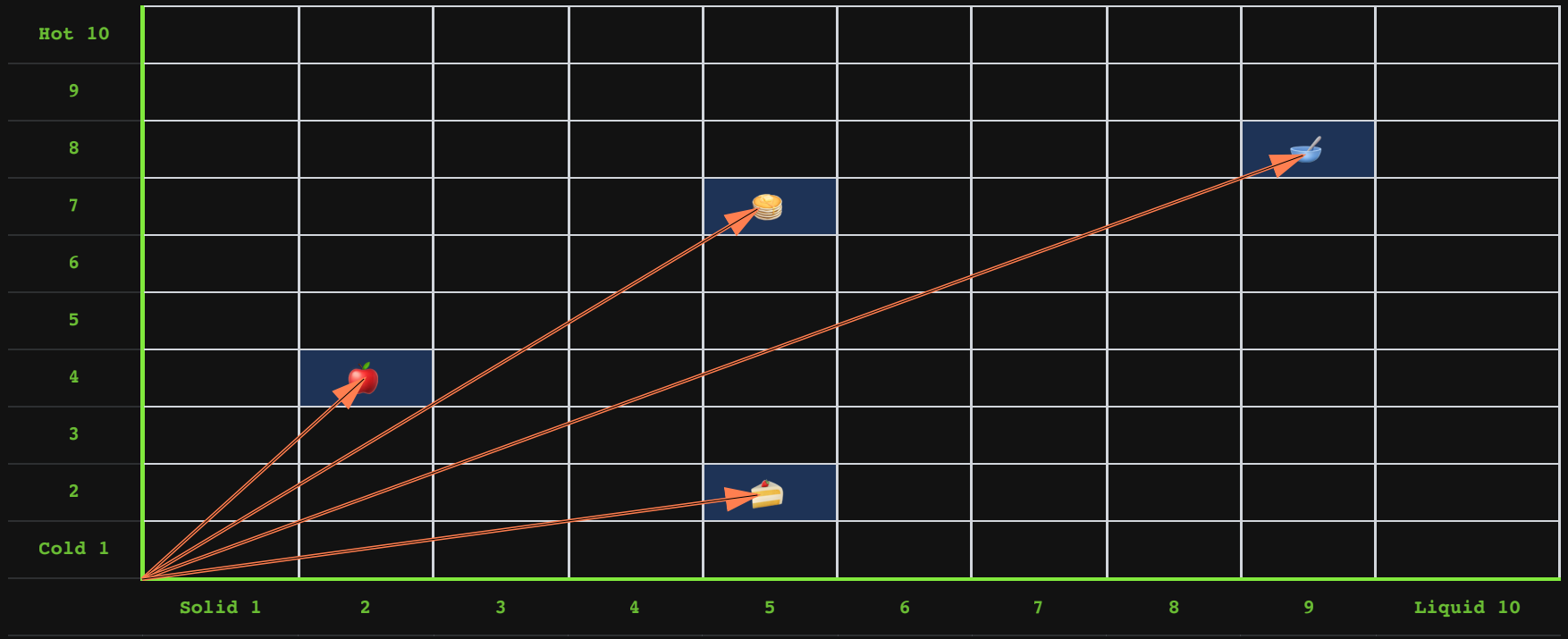
How does AI know what words mean?

It does not … but it can use math to compare *math stuff*

the *math stuff* we are mostly interested in are

Vectors and Dimensions

Vectors and Dimensions



Similarity between vectors

Dot product

$$A \cdot B = \sum_{i=1}^{n} A_i B_i$$

Cosine distance

$$1 - \frac{A \cdot B}{||A|| \cdot ||B||}$$

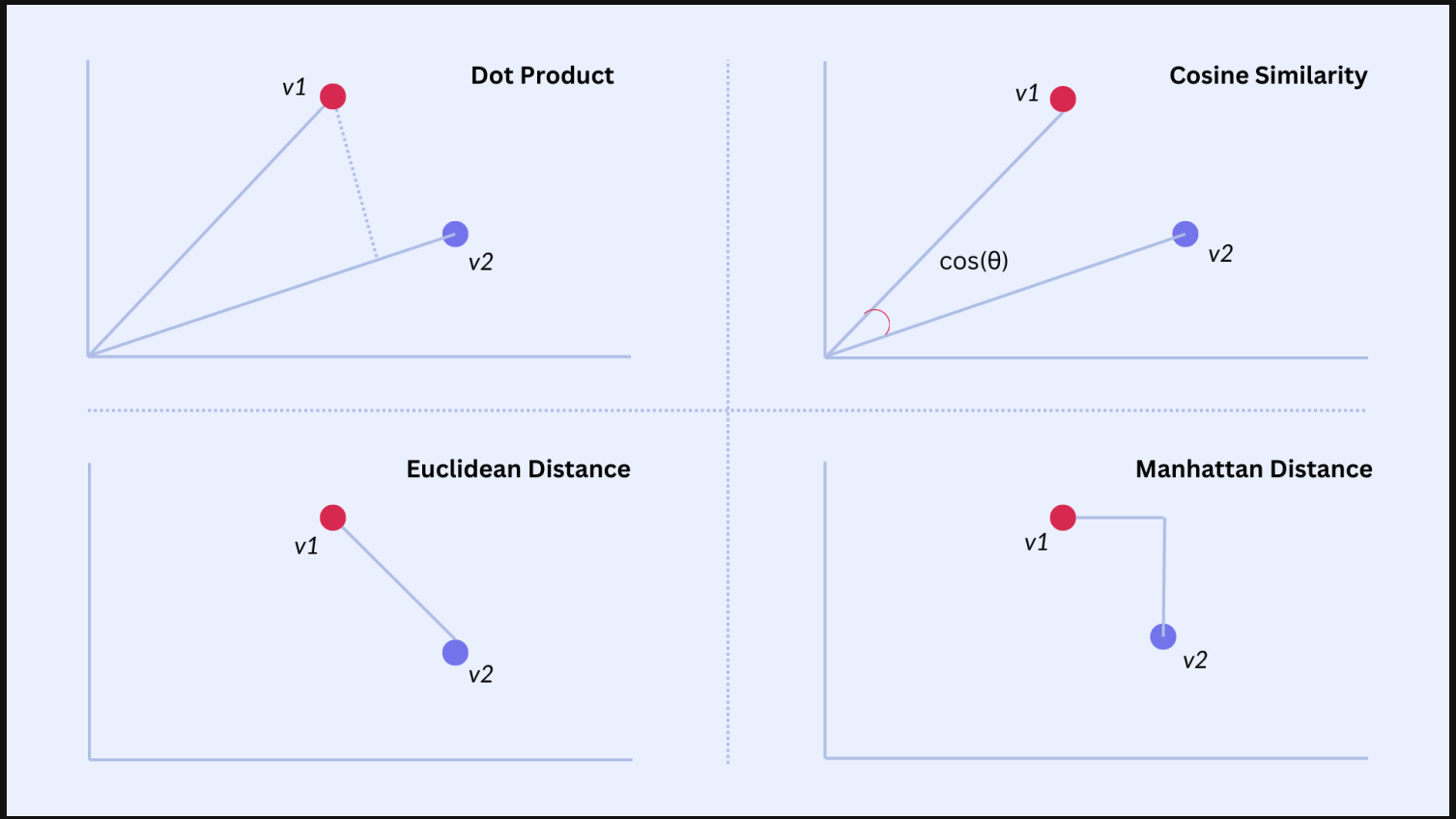
Squared Euclidean
(L2 squared)

$$\sum_{i=1}^{n} (x_i - y_i)^2$$

Manhattan
(L1)

$$\sum_{i=1}^{n} |x_i - y_i|$$

Similarity between vectors



Vector Store

A vector database, vector store or vector search engine is a database that uses the vector space model to store vectors.

Vector databases typically implement one or more approximate nearest neighbor algorithms, so that one can search the database with a query vector to retrieve the closest matching database records.

- MongoDB
- Redis
- Chroma
- MariaDB
- ElasticSearch
- Qdrant
- …

Objective

Environment, Stack and Tools

Hardware

- MacBook Pro
- Chip: M1 Pro
- Memory: 16 GB

Model Runner

- Ollama

RAG Model

- Llama

Project

- Symfony
- Doctrine with PgSQL
- ElasticSearch for search

Embeddings (vectorizing)

- all-minilm sentence transformers

Packages

Existing packages that were used

- `friendsofsymfony/elastica-bundle`

Specifically added for our goal

- `symfony/ai`

Architecture and Solution diagram

Refreshing the vector store

Semantic search experience

RAG experience

Code Snippets

Listeners

```php
#[AsEventListener(event: PostTransformEvent::class)]
class PostElasticaTransformEventListener
{
    private Vectorizer $vectorizer;
    private StoreInterface $vectorStore;

    // ...

    public function __invoke(PostTransformEvent $event)
    {
        $elasticaDoc = $event->getDocument();

        // populate Document as needed ...
        $document = new TextDocument(id: new Uuid(), content: $elasticaDoc->getData());

        $splitter = new TextSplitTransformer(chunkSize: 999, overlap: 333);
        $documents = $splitter->transform($document);

        $this->vectorStore->add($this->vectorizer->vectorizeTextDocuments($documents));
    }
}
```

Finder

```php
class SemanticFinder
{
    private Vectorizer $vectorizer;
    private StoreInterface $store;

    // FosElastica specific dependencies ...

    public function find(string $query, array $options = []): array
    {
        $vector = $this->vectorizer->vectorize($query);
        $results = $this->store->query($vector, options: $options);

        return $this->elasticaToModelTransformer->transform($results);
    }
}
```

Document

```php
use Symfony\AI\Store\Document\TextDocument;
use Symfony\AI\Store\Document\Metadata;

$document = new TextDocument(
    id: new Uuid(),
    content: 'The quick brown fox jumps over the lazy dog.',
    metadata: new Metadata(['source' => 'guide', 'type' => 'asset']),
);
```

Vectorizer

```php
use Symfony\AI\Platform\Bridge\Ollama\Ollama;
use Symfony\AI\Platform\Bridge\Ollama\PlatformFactory;
use Symfony\AI\Store\Document\Vectorizer;

$platform = PlatformFactory::create($_ENV['OLLAMA_HOST_URL'], HttpClient::create());

$vectorizer = new Vectorizer(
    platform: $platform,
    model: new Ollama(name: Ollama::ALL_MINILM),
);

$documents = [/* array of TextDocument-s */];
$vectorized = $vectorizer->vectorizeTextDocuments($documents);
```

Vector Store

```php
use App\AI\Store\Bridge\Elastic\ElasticStore;
use Elastic\Elasticsearch\ClientBuilder;

$client = ClientBuilder::create()
    ->setHosts(['localhost:9200'])
    ->build();

$store = new ElasticStore(
    client: $client,
    indexName: 'ai_vectors',
    dimension: 384, // all-minilm dimensions
);

$store->add($documents);
$store->query($vector);
```

Feedback & Findings

Data quality issues

- High repetition of business stop words resulted in a lot of vectors looking very similar to each other.
- Mixed-language content results in LLMs scoring vectors poorly, in turn yielding poor similarity search.

Use-case performance

- When content is rich, the setup is able to successfully select good search candidates that match semantic meaning behind users' query.
- Content that either has high pollution or lack of normal language makes their way up to search results, without fully understanding why they are there.
- For R.A.G. trials, local Llama was set to use four neighbors (vectors), but Llama used 1 or 2 in its final answer.

Hallway tests

- Ui is not intuitive
- It is the same search, no?
- What is the benefit of doing this over our current search?

Thank you