



ibexa

Summit 25

Augumenting your Ibexa DXP Project
with Private AI

MATEUSZ BIENIEK





Private AI



Hugging Face

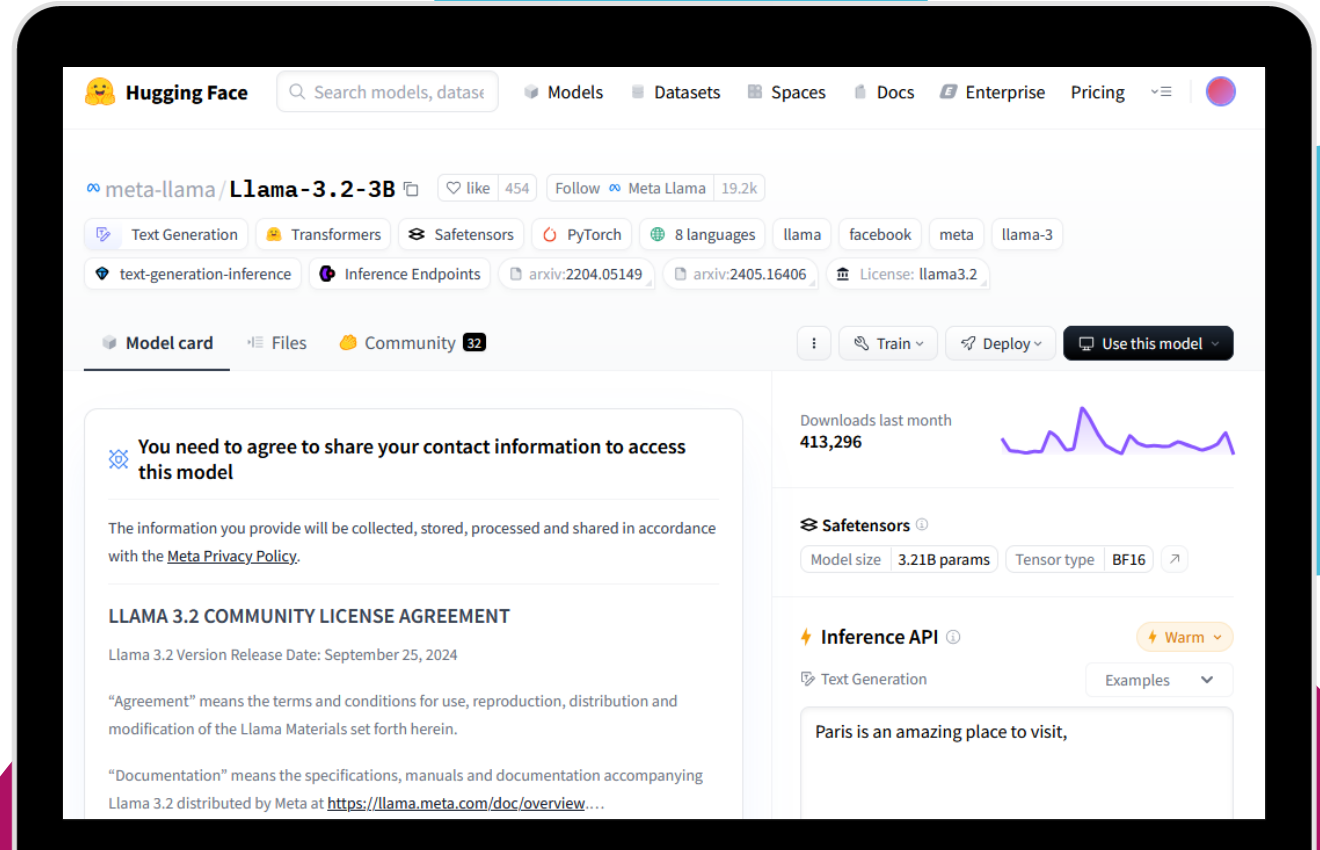
<https://huggingface.co>

The screenshot displays the Hugging Face website interface. On the left, the 'Tasks' section is active, showing a search bar and various task categories: Multimodal (Audio-Text-to-Text, Image-Text-to-Text, Visual Question Answering, Document Question Answering, Video-Text-to-Text, Any-to-Any) and Computer Vision (Depth Estimation, Image Classification, Object Detection, Image Segmentation, Text-to-Image, Image-to-Text, Image-to-Image, Image-to-Video, Unconditional Image Generation, Video Classification, Text-to-Video). On the right, the 'Models' section is visible, showing a list of models with their names, categories, update times, and download/like counts. The models listed are: microsoft/phi-4 (Text Generation, Updated 7 days ago, 72.3k downloads, 1.27k likes), hexgrad/Kokoro-82M (Text-to-Speech, Updated 1 day ago, 15.6k downloads, 1.36k likes), deepseek-ai/DeepSeek-V3 (Updated 16 days ago, 132k downloads, 1.86k likes), NovaSky-AI/Sky-T1-32B-Preview (Text Generation, Updated 2 days ago, 2.88k downloads, 356 likes), openbmb/MiniCPM-o-2_6 (Any-to-Any, Updated about 5 hours ago, 1.46k downloads, 282 likes), and MiniMaxAI/MiniMax-Text-01 (Text Generation, Updated 40 minutes ago, 132 downloads, 212 likes).

Llama-3.2-3B

<https://huggingface.co/meta-llama/Llama-3.2-3B>

- ▶ Very popular LLM by META
- ▶ Free for (almost all) Commercial Use
- ▶ Model Sizes (1B, 3B, 11B, etc)
- ▶ Text Generation
- ▶ Intended use case: commercial and research use in multiple languages
- ▶ Static model trained on an offline dataset (Sep, 2024)



Ollama

- <https://ollama.com/>
<https://github.com/ollama/ollama>
- Platform for running LLMs locally
- Support for many popular models
- Easy to install and run
- Runs on „everything” (Windows, Linux, MacOS)
- Support for multiple hardware (M-series CPUs from apple, GPUs, etc)
- OpenSource and MIT license
- Pros and Cons of running AI locally



```
curl -fsSL https://ollama.com/install.sh | sh
```



```
ollama pull llama3.2:3b
```



```
ollama run llama3.2:3b
```

```
>>> What are the top 3 things to see when you are in Barcelona?
```

```
Barcelona, a vibrant and culturally rich city! There are countless amazing things to see and experience in Barcelona. Here are my top 3 recommendations for your visit:
```

```
1. La Sagrada Família: Antoni Gaudí's iconic cathedral is a must-visit attraction in Barcelona. This UNESCO World Heritage Site is an engineering marvel and a masterpiece of Gothic-Art Nouveau architecture. Take a guided tour to learn more about the history and symbolism behind this incredible structure.
```

```
...
```



Private* AI
and
Ibexa DXP



AI actions

List (4)

<input type="checkbox"/>	Name	Identifier
<input type="checkbox"/>	Answer using context from Blog Post	answer_using_context_from_blog_post
<input type="checkbox"/>	Add Content to embedding	add_text_to_embedding
<input type="checkbox"/>	Current Weather	current_weather
<input type="checkbox"/>	English Spell Checker	spellchecker

Requirement: AI Actions!

AI Framework for Ibexa DXP!

- ▶ LTS Update (Opt-in feature)
- ▶ Integrating Features provided by AI services in your project
- ▶ Backend UI
- ▶ Support for multiple connection
- ▶ Very customizable and extendable
- ▶ Actions, Action Types, Action Configurations, Handlers and more!



```
composer require ibexa/connector-ai
composer require ibexa/connector-openai // <- optional OpenAI
connector, we won't be needing that in this presentation ;)
```



01

Create Ollama Connector

02

AI Functions

03

Semantic Search

04

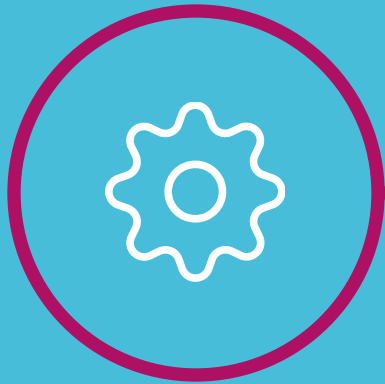
Context Aware Answers

The slide features a solid orange background. At the top and bottom, there are decorative borders made of a mosaic of irregular, colorful shapes in shades of blue, purple, red, and yellow, resembling torn paper or a mosaic pattern.

Custom Connector

Simple Custom AI Connector in 3 steps

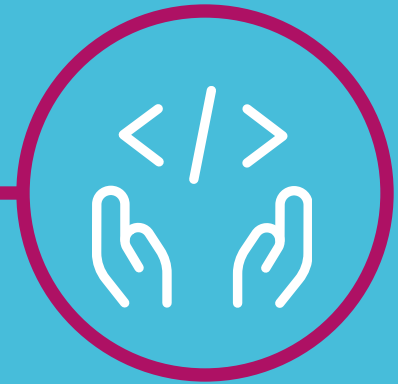
Generate Text with Ollama



Action



Action Type



Handler

Lights Camera ACTION



```
<?php
```

```
// src/AI/Action/GenerateText/Action.php
```

```
declare(strict_types=1);
```

```
namespace App\AI\Action\GenerateText;
```

```
use Ibexa\Contracts\ConnectorAi\Action\TextToText\Action as BaseAction;
```

```
final class Action extends BaseAction
```

```
{
```

```
    public function getActionTypeIdentifier(): string
```

```
    {
```

```
        return 'generate_text';
```

```
    }
```

```
}
```

Lights Camera ACTION

```
// vendor/ibexa/connector-ai/src/contracts/Action/TextToText/Action.php
// ...

abstract class Action extends BaseAction
{
    private Text $text;

    public function __construct(Text $text)
    {
        $this->text = $text;
    }

    public function getInput(): Text
    {
        return $this->text;
    }

    public function getParameters(): array
    {
        return [];
    }
}
```

```

1 <?php
2
3 namespace App\AI\Action\GenerateText;
4
5 use Ibexa\Contracts\ConnectorAi\Action\DataType\Text;
6 use Ibexa\Contracts\ConnectorAi\ActionInterface;
7 use Ibexa\Contracts\ConnectorAi\ActionType\ActionTypeInterface;
8 use Ibexa\Contracts\ConnectorAi\DataType;
9 use Ibexa\Contracts\Core\Exception\InvalidArgumentException;
10
11 final class ActionType implements ActionTypeInterface
12 {
13     public const IDENTIFIER = 'generate_text';
14
15     public function __construct(
16         private readonly iterable $actionHandlers
17     ) {
18     }
19
20     public function getIdentifier(): string
21     {
22         return self::IDENTIFIER;
23     }
24
25     public function getName(): string
26     {
27         return 'Generate text';
28     }
29
30     public function getInputIdentifier(): string
31     {
32         return Text::getIdentifier();
33     }
34

```

```

35     public function getOutputIdentifier(): string
36     {
37         return Text::getIdentifier();
38     }
39
40     public function getOptions(): array
41     {
42         return [];
43     }
44
45     public function createAction(DataType $input, array $parameters = []):
ActionInterface
46     {
47         if (!$input instanceof Text) {
48             throw new InvalidArgumentException(
49                 'input',
50                 'expected \Ibexa\Contracts\ConnectorAi\Action\DataType\Text type, ' .
get_debug_type($input) . ' given.'
51             );
52         }
53
54         return new Action($input);
55     }
56
57     public function getActionHandlers(): iterable
58     {
59         return $this->actionHandlers;
60     }
61 }

```

Lights Camera

ACTION (Type)

```

1 <?php
2
3 declare(strict_types=1);
4
5 namespace App\AI\Handler;
6
7 use Ibexa\Contracts\ConnectorAi\Action\ActionHandlerInterface;
8 use Ibexa\Contracts\ConnectorAi\Action\DataType\Text;
9 use Ibexa\Contracts\ConnectorAi\Action\Response\TextResponse;
10 use Ibexa\Contracts\ConnectorAi\ActionInterface;
11 use Ibexa\Contracts\ConnectorAi\Action\ResponseInterface;
12 use Ibexa\Contracts\ConnectorAi\Action\TextToText\Action as TextToTextAction;
13 use Symfony\Component\HttpFoundation\Request;
14 use Symfony\Contracts\HttpClient\HttpClientInterface;
15
16 class OllamaTextToTextActionHandler implements ActionHandlerInterface
17 {
18     public const IDENTIFIER = 'OllamaTextToText';
19
20     public function __construct(
21         private readonly HttpClientInterface $client,
22         private readonly string $model = 'llama3.2:3b',
23         private readonly string $host = 'http://localhost:11434/api'
24     ) {
25     }
26
27     public function supports(ActionInterface $action): bool
28     {
29         return $action instanceof TextToTextAction;
30     }
31
32     public function handle(ActionInterface $action, array $context = []):
33     ActionResponseInterface
34     {
35         $input = $action->getInput();

```

```

35         $text = $this->sanitizeInput($input->getText());
36
37         $systemMessage = $action->hasActionContext()
38             ? $action->getActionContext()->getActionHandlerOptions()-
39             >get('system_prompt', '')
40             : '';
41
42         $response = $this->client->request(
43             Request::METHOD_POST,
44             sprintf('%s/generate', $this->host),
45             [
46                 'json' => [
47                     'model' => $this->model,
48                     'system' => $systemMessage,
49                     'prompt' => $text,
50                     'stream' => false,
51                 ]
52             ]
53         );
54
55         $output = strip_tags(json_decode($response->getContent(), true)['response']);
56
57         return new TextResponse(new Text([$output]));
58     }
59
60     public static function getIdentifer(): string
61     {
62         return self::IDENTIFIER;
63     }
64
65     private function sanitizeInput(string $text): string
66     {
67         return str_replace(["\n", "\r"], ' ', $text);
68     }
69 }

```

Lights Camera
Handler?

Services!

```
App\AI\Action\GenerateText\ActionType:  
  arguments:  
    $actionHandlers: !tagged_iterator  
    tag: app.connector_ai.action.handler.text_to_text  
    default_index_method: getIdentifier  
    index_by: key  
  tags:  
    - { name: ibexa.ai.action.type, identifier: !php/const  
\App\AI\Action\GenerateText\ActionType::IDENTIFIER }
```

```
App\AI\Handler\OllamaTextToTextActionHandler:
```

```
  tags:  
    - { name: ibexa.ai.action.handler, priority: 0 }  
    - { name: app.connector_ai.action.handler.text_to_text, priority: 0 }
```

```
app.connector_ai.action_configuration.handler.ollama_text_to_text.form_mapper.options:  
  class:
```

```
Ibexa\Bundle\ConnectorAi\Form\FormMapper\ActionConfiguration\ActionHandlerOptionsFormMapper
```

```
  arguments:  
    $formType: 'App\Form\Type\TextToTextOptionsType'
```

```
  tags:  
    - name: ibexa.connector_ai.action_configuration.form_mapper.options  
      type: !php/const \App\AI\Handler\OllamaTextToTextActionHandler::IDENTIFIER
```

```
public function buildForm(FormBuilderInterface $builder, array $options): void  
{  
    $builder->add('system_prompt', TextareaType::class, [  
        'required' => true,  
        'disabled' => $options['translation_mode'],  
        'label' => 'System message',  
    ]);  
}
```

Demo 1!



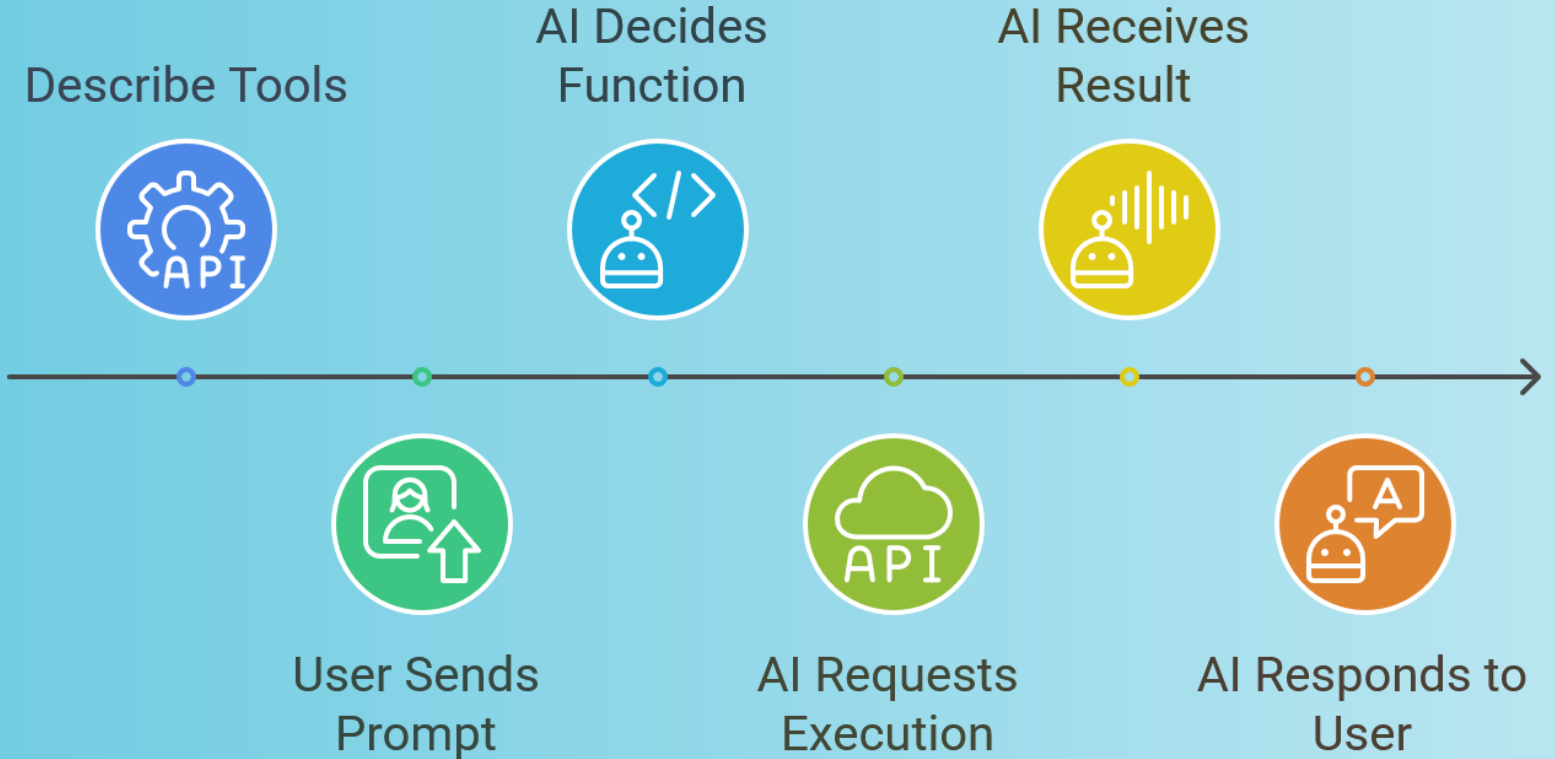


AI Functions

AI Funtions

Allow AI execute funtions in your code!

- ▶ Tools
- ▶ Provide AI Agents with Dynamic Data
- ▶ Automations
- ▶ Not supported by all Models
- ▶ „tool_calls” response



New „Tool” Handler

```
$baseJsonRequestContent = [  
  'model' => $this->model,  
  'system' => $options['system_prompt'],  
  'messages' => [  
    [  
      'role' => 'user',  
      'content' => $text,  
    ],  
  ],  
  'stream' => false,  
  'tools' => [  
    [  
      'type' => 'function',  
      'function' => [  
        'name' => $options['name'],  
        'description' => $options['description'],  
        'properties' => [  
          $options['parameter_name'] => [  
            'type' => 'string',  
            'description' => $options['parameter_description'],  
          ],  
        ],  
        'required' => [$options['parameter_name']],  
      ],  
    ],  
  ],  
];
```

```
$response = $this->client->request(  
  Request::METHOD_POST,  
  sprintf('%s/chat', $this->host),  
  [  
    'json' => $requestJson,  
  ],  
);
```

```
[  
  'role' => 'user',  
  'content' => $text,  
],  
[  
  'role' => 'tool',  
  'content' => '<result_of_running_the_function>',  
],
```

Demo 2!





Semantic Search



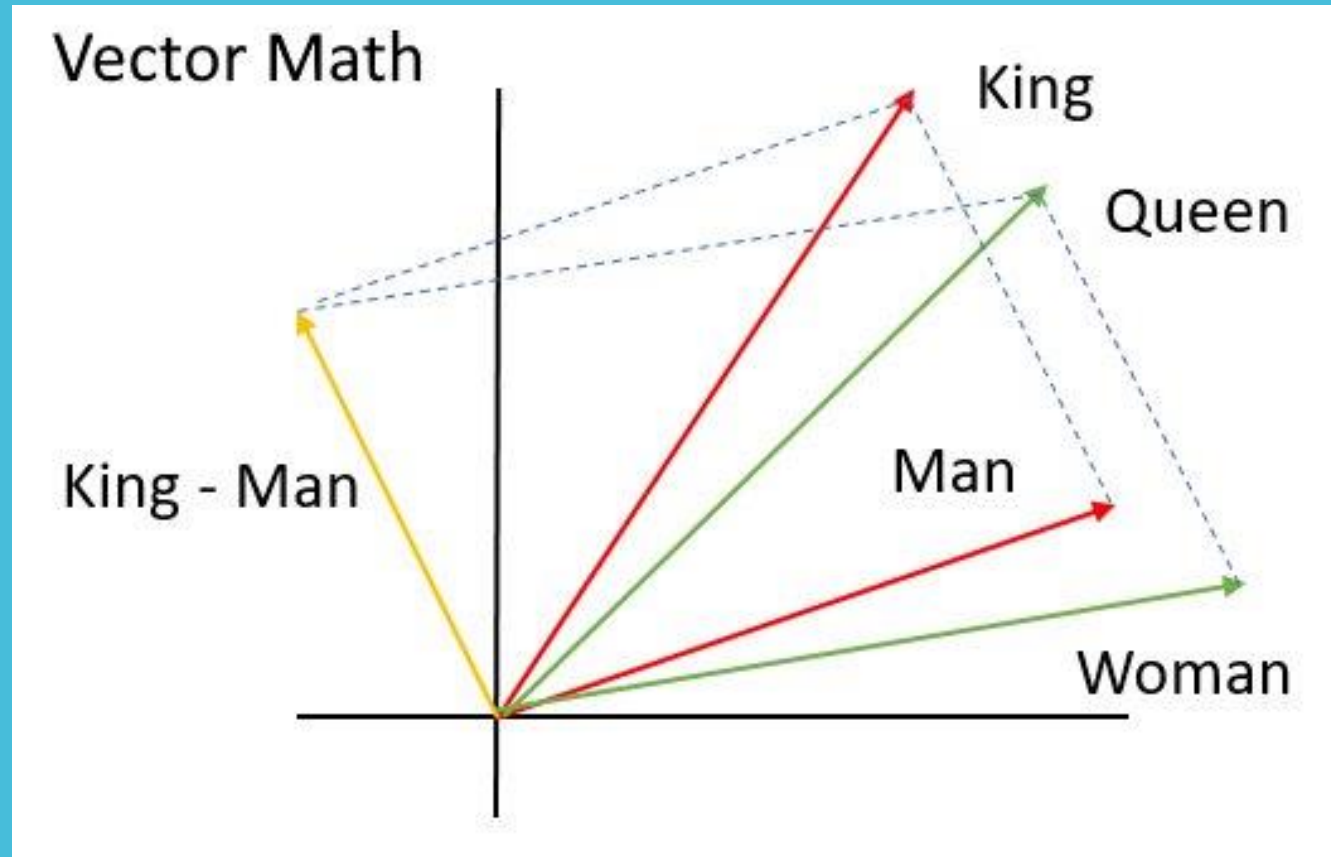


Semantic Search

Semantic search is a search engine technology that interprets the meaning of words and phrases. [/Elastic.co](https://elastic.co)

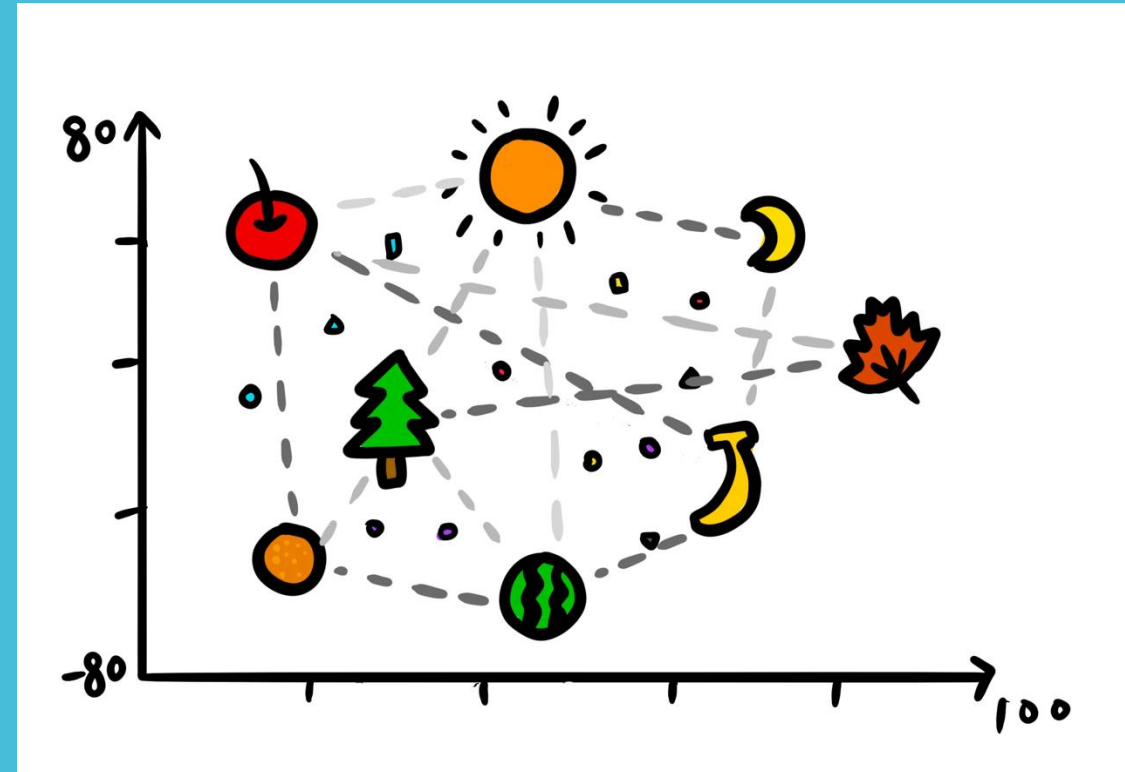
- ▶ Semantic Search VS Keyword Search
- ▶ Vectors
- ▶ Embeddings
- ▶ Vector Store

Vectors

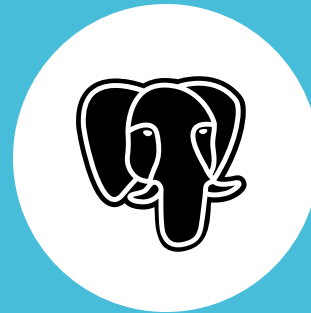


Embeddings

- Numerical representation of documents
- Captures semantics of the text
- Embedding of Question is highly similar to the Answer
- Embedding models
 - Generate vectors for text
 - Embedding size (dimensions)
 - Text splitted to chunks (limits)



Vector Stores



...

Embeddings DataType

TextToEmbeddings ActionType

```
namespace App\AI\Action\DataType;

use Ibexa\Contracts\ConnectorAi\DataType;

class Embeddings implements DataType
{
    public const IDENTIFIER = 'embeddings';

    /**
     * @param array{text: string, embeddings: array} $data
     */
    public function __construct(
        private array $data
    ) {
    }

    /**
     * @return array{text: string, embeddings: array}
     */
    public function getList(): array
    {
        return $this->data;
    }

    public static function getIdentifer(): string
    {
        return self::IDENTIFIER;
    }
}
```

```
final class ActionType implements ActionTypeInterface
{
    public const IDENTIFIER = 'text_to_embeddings';

    public function __construct(
        private readonly iterable $actionHandlers
    ) {
    }

    // ...

    public function getName(): string
    {
        return 'Text to Embeddings';
    }

    public function getInputIdentifier(): string
    {
        return Text::getIdentifer();
    }

    public function getOutputIdentifier(): string
    {
        return Embeddings::getIdentifer();
    }

    // ...
}
```

OllamaTextToEmbeddingsActionHandler

```
public function handle(ActionInterface $action, array $context = []): ActionResponseInterface
{
    $input = $action->getInput();

    $embeddings = [];
    /** @var string $text */
    foreach ($input->getList() as $text) {
        $response = $this->client->request(
            Request::METHOD_POST,
            sprintf('%s/embeddings', $this->host),
            [
                'json' => [
                    'model' => $this->model,
                    'prompt' => $text,
                ]
            ]
        );

        $content = json_decode($response->getContent(), true);
        $embeddings[] = [
            'text' => $text,
            'embeddings' => $content['embedding'],
        ];
    }

    return new ActionResponse(new Embeddings($embeddings));
}
```

Semantic Search

Postgresql Vector Store

<https://github.com/pgvector/pgvector-php>

Insert a vector

```
use Pgvector\Vector;

$embedding = new Vector([1, 2, 3]);
pg_query_params($db, 'INSERT INTO items (embedding) VALUES ($1)', [$embedding]);
```

Get the nearest neighbors to a vector

```
$embedding = new Vector([1, 2, 3]);
$result = pg_query_params($db, 'SELECT * FROM items ORDER BY embedding <-> $1 LIMIT 5', [$embedd.
```

Add an approximate index

```
pg_query($db, 'CREATE INDEX ON items USING hnsw (embedding vector_l2_ops)');
// or
pg_query($db, 'CREATE INDEX ON items USING ivfflat (embedding vector_l2_ops) WITH (lists = 100)'
```

Demo 3!



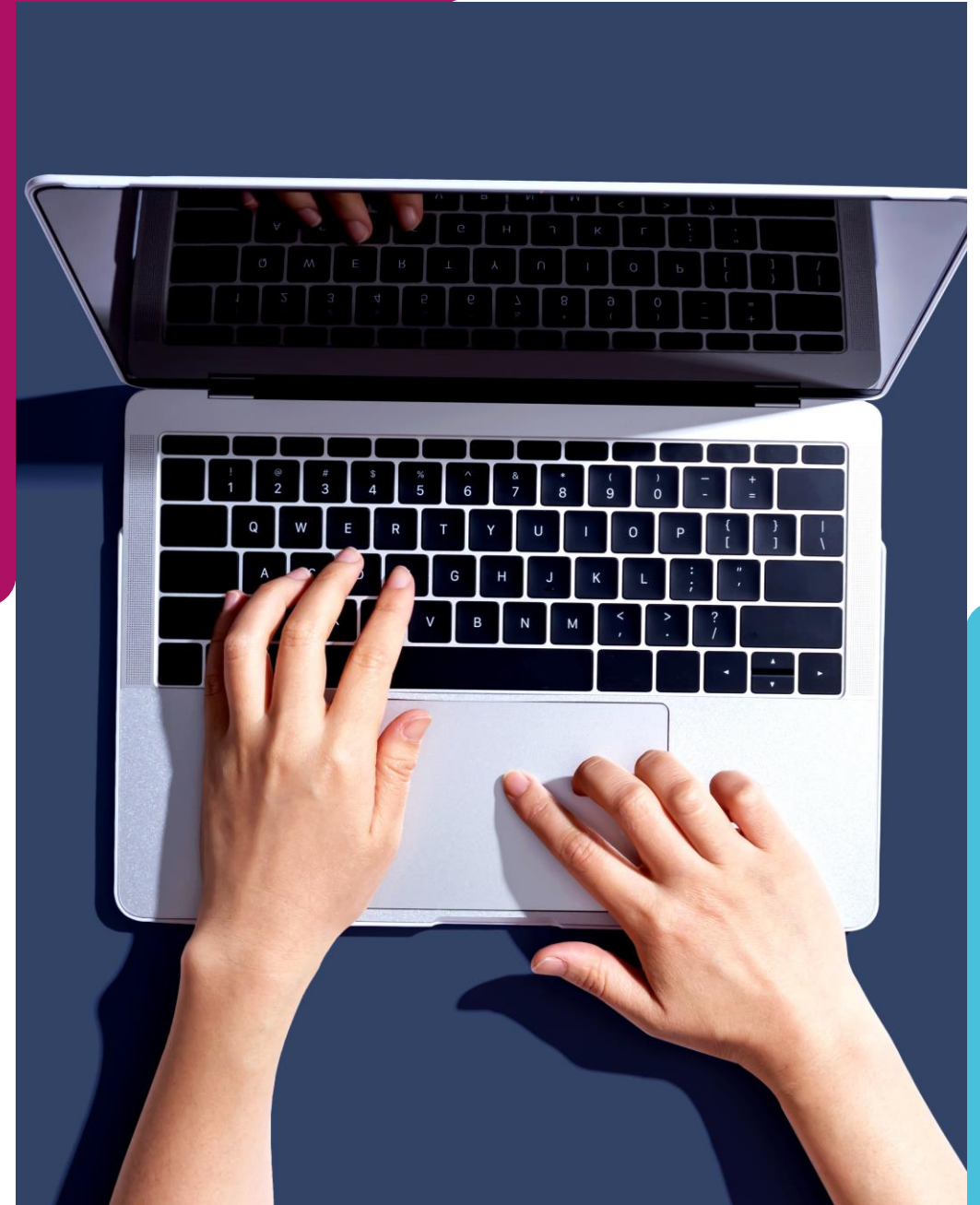


Context-Aware Answers

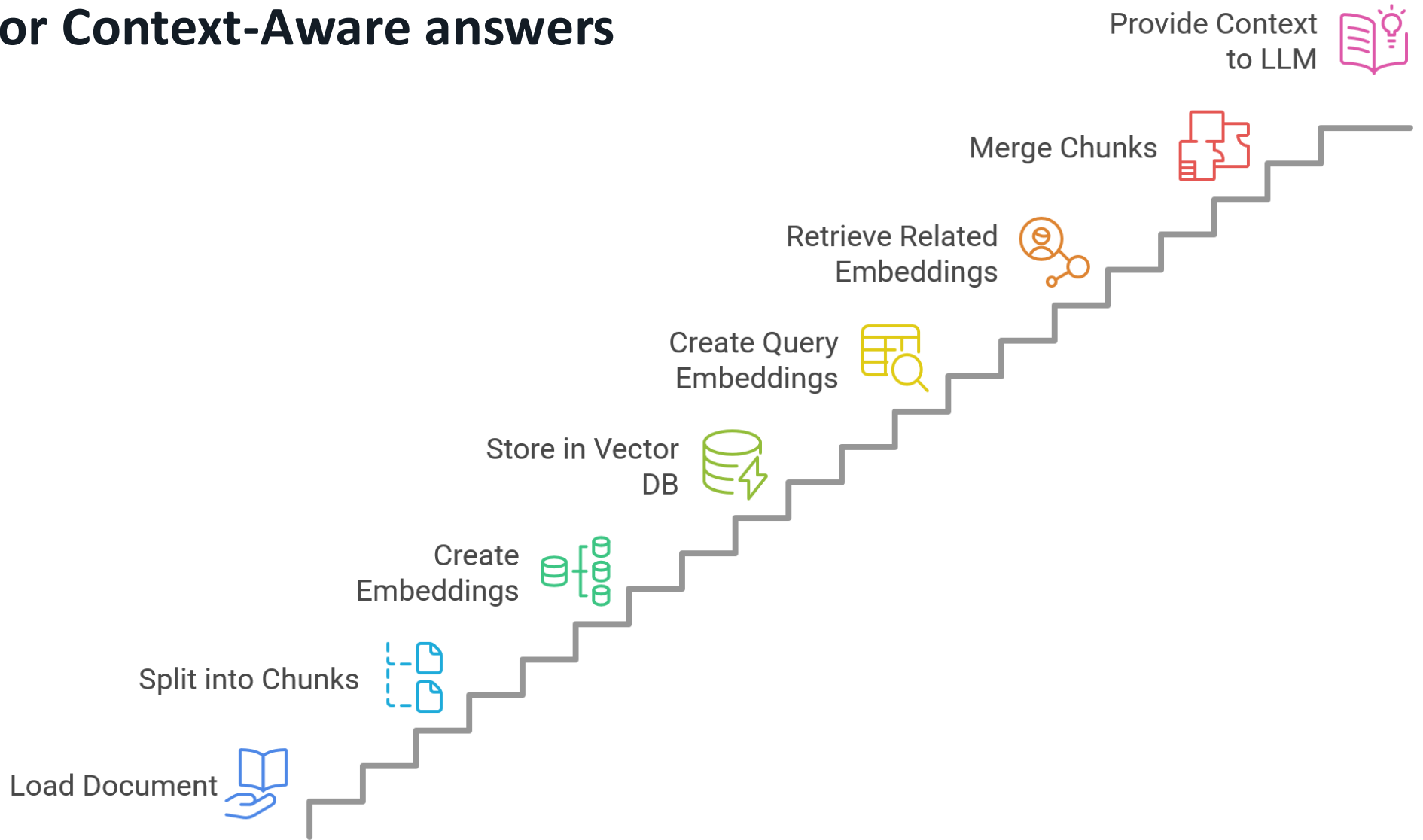


Context-Aware Answers

- Make AI aware of your context!
- AI knowledge is limited to most popular „public stuff“
- Training AI with your own data is complicated and expensive...
- ... but you don't have to do it
- RAG for the rescue! (Retrieval augmented generation)



Steps for Context-Aware answers



Let's look at the code

```
$actionConfiguration = $this->actionConfigurationService->getActionConfiguration(
    'add_text_to_embedding'
);

$action = new TextToEmbeddingsAction(new Text([$text]));
$responseObject = $this->actionService->execute($action, $actionConfiguration);

$embeddingsResponse = $responseObject->getOutput()->getList();
$embeddings = [];
foreach ($embeddingsResponse as $embeddingsData) {
    $embeddings[] = new Embeddings($embeddingsData['text'], $embeddingsData['embeddings']);
}

$hits = $this->vectorStore->search($embeddings, null);

$context = '';
foreach ($hits as $hit) {
    $context .= $hit->getText() . ' ';
}

$answerQuestionActionConfiguration = $this->actionConfigurationService->getActionConfiguration(
    'answer_using_context_from_blog_post'
);
$answerQuestionAction = new AnswerQuestionWithContextAction(
    new TextWithContext([
        [
            'text' => $text, 'context' => $context
        ],
    ]),
);
$responseObject = $this->actionService->execute($answerQuestionAction, $answerQuestionActionConfiguration);
```

Demo 4!





Q&A

Thank you!



mateusz.bieniek@ibexa.co



[@Mateusz Bieniek](#)



<https://github.com/mateuszbieniek/barcelona>