# ***ILaaS Embeddings***

Si vous utilisez un logiciel exploitant l'IA générative et qu'il permet de choisir l'URL du serveur d'IA, vous pouvez substituer les services d'OpenAI par les services d'ILaaS très simplement.
Il vous faudra pour cela obtenir du consortium une clé d'API pour vous authentifier sur le service d'inférence d'ILaaS.

Concrètement, il suffit de :

1. demander une clé d'API ILaaS en tant que consommateur d'inférence,
2. paramétrer l'URL d'accès au service avec l'URL qui vous sera fournie,
3. saisir la clé d'API ILaaS dans l'interface de configuration.

## Embedding en quelques mots :

Un embedding est un vecteur de nombres réels compris entre -1 et 1 qui représente le sens sémantique d'un texte. Cette représentation est utilisée pour faire de la recherche d'information au sein d'un document. Il faut transformer le contenu à analyser en embedding, ainsi que l'élément que l'on recherche. Par la suite il faut comparer ces deux vecteurs.
Pour comparer deux embeddings, on utilise généralement la similarité cosinus, qui mesure l'angle entre deux vecteurs dans l'espace traduit par la formule suivante

$$cos(\theta) = \frac{a \,.\, b}{||a|| \times ||b||}$$ avec $cos(\theta)$ la "différence de direction"

Avec

$$a.b = \sum_i a_i \times b_i \Rightarrow \text{produit scalaire de}$$

$$||a|| = \sqrt{\sum_i a_i^{\,2}} \Rightarrow \text{norme de a}$$

$$||b|| = \sqrt{\sum_i b_i^{\,2}} \Rightarrow \text{norme de b}$$

Le résultat est compris entre **-1 et 1** :

- proche de **1** → les textes sont très similaires sémantiquement
- proche de **0** → les textes n'ont pas de lien sémantique
- proche de **-1** → sens opposés (rare en pratique)

## Récupérer du contenu :

Pour pouvoir utiliser les embeddings, nous avons besoin d'un élément textuel à transformer. Mais nous devons déjà extraire les éléments textuels. Prenons l'exemple d'un document PDF. Il va falloir transformer le document PDF en texte intelligible pour l'IA. Pour ce faire, on utilise un parser qui permet d'extraire du texte d'un document.

```php
public function extractText(): string {
    $parser = new \Smalot\PdfParser\Parser();
    $pdf = $parser->parseFile($this->chemin) ;
    return $pdf->getText();
}
```

## Regroupement du texte :

Une fois le texte extrait du document PDF, il est impossible et inefficace de le transformer dans son intégralité en embedding, la précision sémantique serait noyée. Il faut donc procéder à du "chunking". Segmenter en petites parties afin de pouvoir détecter les éléments les plus pertinents. L'enjeu ici est de trouver la bonne taille afin d'avoir un équilibre entre un chunk suffisamment grand pour avoir du contexte , et un contexte suffisamment petit pour ne pas que l'information principale du passage soit diluée.

```php
public function decoupePDF(string $text): array {
    $chunks = [];
    $limit = strlen($text) ;
    for ($i = 0; $i < $limit; $i += 750) {
        $chunks[] = substr($text, $i, 800) ;
    }
    return $chunks;
}
```

## Intervention de iLaaS dans le processus :

Après avoir obtenu nos chunks, nous allons interroger notre API. Pour ce faire, vous aurez besoin de votre clé API fournie par ILaaS, et de pointer vers le lien suivant : https://rag-api.ilaas.fr/v1/models et interroger le modèle “bge-m3” qui est pour faire du “test-embeddings-inference”.

```
public function embedding(array $chunks): array {
        $API_KEY =  "Votre clé API fournie par iLaaS" ;
  $link = "https://rag-api.ilaas.fr/v1/models" ;
  $result = [];

  foreach ($chunks as $c) {
    $data = [
      "model" => "bge-m3",
      "input" => $c
    ];

    $ch = curl_init($link);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_HTTPHEADER, [
      "Content-Type: application/json",
      "Authorization: Bearer $apiKey"
    ]);
    curl_setopt($ch, CURLOPT_POST, true);
    curl_setopt($ch, CURLOPT_POSTFIELDS, json_encode($data));

    $response = curl_exec($ch);
    curl_close($ch);

    $output = json_decode($response, true);
    $embedding = $output['data'][0]['embedding'] ?? null;
    $results[] = [
      'chunk' => $c,
      'embedding' => $embedding
    ];
  }
  return $results;
}
```

## Comparaison des embeddings:

Une fois que toutes les données ont été transformées en embeddings (question utilisateur et la source documentaire), l'étape suivante consiste à mesurer leur proximité sémantique. En s'appuyant sur les mesures de similarité (cf page 1), le système calcule la distance entre le vecteur de la question de l'utilisateur et les vecteurs stockés en base de connaissance. Cela permet d'extraire les segments de textes dont la représentation vectorielle est la plus proche de la question utilisateur. De cette manière, les résultats sont garantis d'être le plus proches contextuellement de la question utilisateur.

```php
public function embeddingCompare(array $question , array $embedding): float {
    $cos0 = 0.0;
    $normeA = 0.0;
    $normeB = 0.0;

    for ($i = 0 ; $i<count($question) ; $i++){
        $cos0 += $question[$i] * $embedding[$i];
        $normeA += $question[$i] ** 2;
        $normeB += $embedding[$i] ** 2;
    }

    if ($normeA == 0 || $normeB == 0) return 0.0;

    return $cos0 / (sqrt($normeA) * sqrt($normeB));

}
```